MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

K. Drage, Ed.
M. Makaraju
J. Stoetzer-Bradler
Alcatel-Lucent
R. Ejzak
J. Marcon
Unaffiliated

October 27, 2014

SDP-based "SCTP over DTLS" data channel negotiation
draft-ejzak-mmusic-data-channel-sdpneg-02

Abstract

The Real-Time Communication in WEB-browsers (RTCWeb) working group is charged to provide protocols to support direct interactive rich communications using audio, video, and data between two peers' web-browsers. For the support of data communication, the RTCWeb working group has in particular defined the concept of bi-directional data channels over SCTP, where each data channel might be used to transport other protocols, called sub-protocols. Data channel setup can be done using either the internal in-band band (also referred to as 'internal' for the rest of the document) WebRTC Data Channel Establishment Protocol or some external out-of-band simply referred to as 'external negotiation' in the rest of the document . This document specifies how the SDP offer/answer exchange can be used to achieve such an external negotiation. Even though data channels are designed for RTCWeb use initially they may be used by other protocols like, but not limited to, the CLUE protocol. This document is intended to be used wherever data channels are used.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 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 3
2. Conventions 3
3. Terminology 3
4. Data Channels 4
4.1. Stream identifier numbering 5
4.2. Generic external negotiation 6
4.2.1. Overview 6
4.2.2. Opening a data channel 6
4.2.3. Closing a data channel 7
5. SDP-based external negotiation 7
5.1. SDP syntax 8
5.1.1. SDP attribute for data channel parameter negotiation 8
5.1.1.1. dcmap attribute 9
5.1.1.2. label parameter 10
5.1.1.3. subprotocol parameter 11
5.1.1.4. max-retr parameter 11
5.1.1.5. max-time parameter 11
5.1.1.6. ordered parameter 11
5.1.2. Sub-protocol specific attributes 11
5.2. Procedures 13
5.2.1. Managing stream identifiers 13
5.2.2. Opening a data channel 13
5.2.3. Closing a data channel 15
5.2.4. Various SDP offer/answer scenarios and considerations 16
6. Examples 17
7. Security Considerations 19
8. IANA Considerations 19
9. Acknowledgments 20
10. CHANGE LOG 20
10.1. Changes against '-01' 20
10.2. Changes against '-00' 20

11. References 20
11.1. Normative References 20
11.2. Informative References 21
Authors' Addresses 22

1. Introduction

The RTCWeb working group has defined the concept of bi-directional data channels running on top of SCTP/DTLS. RTCWeb leaves it open for other applications to use data channels and its in-band or out-of-band protocol for creating them. Each data channel consists of paired SCTP streams sharing the same SCTP Stream Identifier. Data channels are created by endpoint applications through the WebRTC API, or other users of data channel like CLUE, and can be used to transport proprietary or well-defined protocols, which in the latter case can be signaled by the data channel "sub-protocol" parameter, conceptually similar to the WebSocket "sub-protocol". However, apart from the "sub-protocol" value transmitted to the peer, RTCWeb leaves it open how endpoint applications can agree on how to instantiate a given sub-protocol on a data channel, and whether it is signaled in-band or out-of-band (or both). In particular, the SDP offer generated by the application includes no channel-specific information.

This document defines SDP-based out-of-band negotiation procedures to establish data channels for transport of well-defined sub-protocols.

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 [RFC2119].

3. Terminology

This document uses the following terms:

Data channel: A bidirectional channel consisting of paired SCTP outbound and inbound streams.

Data channel stack: An entity which, upon application request, runs data channel protocol to keep track of states, sending and receive data. If the application is browser based Javascript application then this stack resides in the browser. If the application is a native application then this stack resides in application and accessible to it via some sort of APIs.

Data channel properties: fixed properties assigned to a data channel at the time of its creation. Some of these properties determine the way the data channel stack transmits data on this channel (e.g., stream identifier, reliability, order of delivery...)

DCEP - Data Channel Establishment Protocol defined in [I-D.ietf-rtcweb-data-protocol].

External negotiation: Data channel negotiation based on SDP offer/ answer outlined in this specification.

Internal negotiation: Data channel negotiation based on Data Channel Establishment Protocol defined in [I-D.ietf-rtcweb-data-protocol].

In-band: transmission through the peer-to-peer SCTP association.

In-band negotiation: data channel negotiation based Data Channel Establishment Protocol defined in [I-D.ietf-rtcweb-data-protocol].

Out-of-band: transmission through the application signaling path.

Peer: From the perspective of one of the agents in a session, its peer is the other agent. Specifically, from the perspective of the SDP offerer, the peer is the SDP answerer. From the perspective of the SDP answerer, the peer is the SDP offerer.

Stream identifier: the identifier of the outbound and inbound SCTP streams composing a data channel.

4. Data Channels

This section summarizes how data channels work in general. Note that the references to ’browser’ here is intentional as in this specific example the data channel user is a webrtc enabled browser.

A WebRTC application creates a data channel via the Data Channel API, by providing a number of setup parameters (sub-protocol, label, reliability, order of delivery, priority). The application also specifies if it wants to make use of the in-band negotiation using the DCEP [I-D.ietf-rtcweb-data-protocol], or if the application intends to perform an "external negotiation" using some other in-band or out-of-band mechanism.

In any case, the SDP offer generated by the browser is per [I-D.ietf-mmusic-sctp-sdp]. In brief, it contains one m-line for the

SCTP association on top of which data channels will run, and one attribute per protocol assigned to the SCTP ports:

OPEN ISSUE: The syntax in [I-D.ietf-mmusic-sctp-sdp] may change as that document progresses. In particular we expect "webrtc-datachannel" to become a more general term.

```
m=application 54111 DTLS/SCTP webrtc-datachannel
c=IN IP4 79.97.215.79
a=fmtp:webrtc-datachannel max-message-size=100000
a=sctp-port 5000
a=setup:actpass
a=connection:new
a=fingerprint:SHA-1 \
    4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
```

Note: A WebRTC browser will only use m-line format "webrtc-datachannel", and will not use other formats in the m-line for other protocols such as t38. [I-D.ietf-mmusic-sctp-sdp] supports only one SCTP association to be established on top of a DTLS session.

Note: This SDP syntax does not contain any channel-specific information.

4.1. Stream identifier numbering

Independently from the requested type of negotiation, the application creating a data channel can either pass to the browser the stream identifier to assign to the data channel or else let the browser pick one identifier from the ones unused.

To avoid glare situations, each endpoint can moreover own an exclusive set of stream identifiers, in which case an endpoint can only create a data channel with a stream identifier it owns.

Which set of stream identifiers is owned by which endpoint is determined by convention or other means.

For data channels negotiated in-band, one endpoint owns by convention the even stream identifiers, whereas the other owns the odd stream identifiers, as defined in [I-D.ietf-rtcweb-data-protocol].

For data channels externally negotiated, no convention is defined by default.

4.2. Generic external negotiation

4.2.1. Overview

In-band negotiation only provides for negotiation of data channel transport parameters and does not provide for negotiation of sub-protocol specific parameters. External negotiation can be defined to allow negotiation of parameters beyond those handled by in-band negotiation, e.g., parameters specific to the sub-protocol instantiated on a particular data channel. See Section 5.1.2 for an example of such a parameter.

The following procedures are common to all methods of external negotiation, whether in-band (communicated using proprietary means on an already established data channel) or out-of-band (using SDP or some other protocol associated with the signaling channel).

4.2.2. Opening a data channel

In the case of external negotiation, the endpoint application has the option to fully control the stream identifier assignments. However these assignments have to coexist with the assignments controlled by the data channel stack for the in-band negotiated data channels (if any). It is the responsibility of the application to ensure consistent assignment of stream identifiers.

When the application requests the creation of a new data channel to be set up via external negotiation, the data channel stack creates the data channel locally without sending any DATA CHANNEL OPEN message in-band, and sets the data channel state to Connecting if the SCTP association is not yet established, or sets the data channel state to Open if the SCTP association is already established. The side which starts external negotiation creates data channel using underlying data channel stack API and the data channel is put into open state immediately (assuming ICE, SCTP procedures were already done). However, the application can’t send data on this data channel until external negotiation is complete with the peer. This is because peer needs to be aware and accept the data channel via external negotiation. The peer after accepting the data channel offer can start sending data immediately. This implies that offerer may get data channel message before external negotiation is complete and the application should be ready to handle it.

If the peer rejects the data channel part of the offer then it doesn’t have to do anything as the data channel was not created using the stack. The offerer on the other hand needs to close the data channel that was opened by invoking relevant data channel stack API procedures.

It is also worth noting that a data channel stack implementation may not provide any API to create and close data channels; instead the data channels are used on the fly as needed just by communicating via external means or by even having some local configuration/assumptions on both the peers.

The application then externally negotiates the data channel properties and sub-protocol properties with the peer's application.

[ASSUMPTION] The peer must then symmetrically create a data channel with these negotiated data channel properties. This is the only way for the peer's data channel stack to know which properties to apply when transmitting data on this channel. The data channel stack must allow data channel creation with any non-conflicting stream identifier so that both peers can create the data channel with the same stream identifier.

In case the external negotiation is correlated with an SDP offer/answer exchange that establishes the SCTP association, the SCTP initialization completion triggers a callback from the data channel stack to an application on both the ends to change the data channel state from Connecting to Open. The details of this interface is specific to the data channel user application. Browser based applications (could include hybrid apps) will use [WebRtcAPI], while native applications use a compatible API, which is yet to be specified. See Section 5.2.2 for details on when the data channel stack can assume the data channel is open, and on when the application can assume the data channel is open.

4.2.3. Closing a data channel

When the application requests the closing of an externally negotiated data channel, the data channel stack always performs an in-band SSN reset for this channel.

Depending upon the method used for external negotiation and the sub-protocol associated with the data channel, the closing might in addition be signaled to the peer via external negotiation.

5. SDP-based external negotiation

This section defines a method of external negotiation by which two clients can negotiate data channel-specific and sub-protocol-specific parameters, using the out-of-band SDP offer/answer exchange. This SDP extension can only be used with SDP offer/answer model.

5.1. SDP syntax

Two new SDP attributes are defined to support external negotiation of data channels. The first attribute provides for negotiation of channel-specific parameters. The second attribute provides for negotiation of sub-protocol-specific parameters.

5.1.1. SDP attribute for data channel parameter negotiation

Associated with the SDP "m" line that defines the SCTP association for data channels (defined in Section 4), each SDP offer and answer includes an attribute line that defines the data channel parameters for each data channel to be negotiated. Each attribute line specifies the following parameters for a data channel: Stream Identifier, sub-protocol, label, reliability, order of delivery, and priority. Conveying a reliable data channel is achieved by including neither 'max-retr' nor 'max-time'. Conveying an unreliable data channel is achieved by including only one of 'max-retr' or 'max-time'. By definition max-retr and max-time are mutually exclusive, so only one of them can be present in a=dcmap. If an SDP offer contains both of these parameters then such an SDP offer will be rejected. If an SDP answer contains both of these parameters then the offerer may treat it as an error and may assume the associated SDP offer/answer failed and may take appropriate recovery actions. These recovery options are outside the scope of this specification. Following is an example of the attribute line for sub-protocol "BFCP" and stream id "2":

a=dcmap:2 subprotocol="BFCP";label="channel 2"

The SDP answer shall echo the same subprotocol, max-retr, max-time, ordered parameters, if those were present in the offer, and may include a label parameter. They may appear in any order, which could be different from the SDP offer, in the SDP answer.

The same information MUST be replicated without changes in any subsequent offer or answer, as long as the data channel is still opened at the time of offer or answer generation.

> Note: This attribute is derived from attribute "webrtc-DataChannel", which was defined in old version 03 of the following draft, but which was removed along with any support for SDP external negotiation in subsequent versions: [I-D.ietf-mmusic-sctp-sdp].
>
> Note: This document does not provide a complete specification of how to negotiate the use of a data channel to transport BFCP. Procedures specific to each sub-protocol such as BFCP will be

documented elsewhere. The use of BFCP is only an example of how the generic procedures described herein might apply to a specific sub-protocol.

The intention of exchanging these attributes is to create data channels on both the peers with the same set of attributes without actually using [I-D.ietf-rtcweb-data-protocol]. It is assumed that the data channel properties (reliable/unreliable, ordered/unordered) are suitable per the sub-protocol transport requirements. Data channel types defined in [I-D.ietf-rtcweb-data-protocol] are mapped to SDP in the following manner:

```
DATA_CHANNEL_RELIABLE
      a=dcmap:2 subprotocol="BFCP";label="channel 2"

DATA_CHANNEL_RELIABLE_UNORDERED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
                ordered=0

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
                max-retr=3

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT_UNORDERED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
                max-retr=3;ordered=0;

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
                max-time=10000;

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
                max-time=10000; ordered=0
```

5.1.1.1. dcmap attribute

The 'stream' parameter indicates the actual stream identifier within the association used to form the channel. Stream is a mandatory parameter and is noted directly after the "a=dcmap:" attribute's colon.

Formal Syntax:
TBD: Should this be moved to SDP grammar section?

Name: dcmap

Value: dcmap-value

Usage Level: media

Charset Dependent: no

Syntax:

```
dcmap-value     = dcmap-stream-id
                  [ SP dcmap-opt *(";" dcmap-opt) ]
dcmap-opt       = ordering-opt / subprotocol-opt / label-opt
                  / maxretr-opt / maxtime-opt
                  ; Either only maxretr-opt or maxtime-opt
                  ; is present.
                  ; Both MUST not be present.

dcmap-stream-id = 1*DIGIT
ordering-opt    = "ordered=" ordering-value
ordering-value  = "0"/"1"
subprotocol-opt = "subprotocol=" quoted-string
label-opt       = "label=" quoted-string
maxretr-opt     = "max-retr=" maxretr-value
maxretr-value   = <from-Reliability-Parameter of
                   I-D.ietf-rtcweb-data-protocol>
                  ; number of retransmissions
maxtime-opt     = "max-time=" maxtime-value
maxtime-value   = <from-Reliability-Parameter of
                  I-D.ietf-rtcweb-data-protocol>
                  ; milliseconds

quoted-string   = DQUOTE *(quoted-char / escaped-char) DQUOTE
quoted-char     = SP / quoted-visible
quoted-visible  = %21 / %23-24 / %26-7E ; VCHAR without " or %
escaped         = "%" HEXDIG HEXDIG
DQUOTE          = <from-RFC5234>
integer         = <from-RFC5234>
```

Examples:

```
a=dcmap:0
a=dcmap:1 subprotocol="BFCP";max-time=60000
a=dcmap:2 subprotocol="MSRP";ordered;label="MSRP"
a=dcmap:3 label="Label 1";unordered;max-retr=5
a=dcmap:4 label="foo%09bar";ordered;max-time=15000;max-retr=3
```

5.1.1.2. label parameter

The optional ’label’ parameter indicates the name of the channel. It represents a label that can be used to distinguish, in the context of the WebRTC API, an RTCDataChannel object from other RTCDataChannel

objects. Label is a mandatory parameter. This parameter maps to the 'label' parameter defined in [I-D.ietf-rtcweb-data-protocol]

5.1.1.3. subprotocol parameter

The 'subprotocol' parameter indicates which protocol the client expects to exchange via the channel. Subprotocol is a mandatory parameter.

[ACTION ITEM] The IANA registry to be used for the subprotocol parameter is still to be determined. It also needs to be determined what the relationship is to existing registries and how to reference already-existing protocols.

5.1.1.4. max-retr parameter

This parameter indicates that the data channel is unreliable. The 'max-retr' parameter indicates the max times a user message will be retransmitted. The max-retr parameter is optional with default value unbounded. This parameter maps to the 'Number of RTX' parameter defined in [I-D.ietf-rtcweb-data-protocol]

5.1.1.5. max-time parameter

This parameter indicates that the data channel is unreliable. A user messages will no longer be transmitted or retransmitted after a specified life-time given in milliseconds in the 'max-time' parameter. The max-time parameter is optional with default value unbounded. This parameter maps to the 'Lifetime in ms' parameter defined in [I-D.ietf-rtcweb-data-protocol]

5.1.1.6. ordered parameter

The ordered' parameter indicates that DATA chunks in the channel MUST be dispatched to the upper layer by the receiver while preserving the order. The ordered parameter is optional and takes two values: "0" for ordered and "1" for ordered delivery with "1" as the default value. Any other value is ignored and default ordered is assumed. If the ordered parameter is absent, the receiver is required to deliver DATA chunks to the upper layer in proper order. This parameter maps to the ordered or unorderd data channel types as defined in [I-D.ietf-rtcweb-data-protocol]

5.1.2. Sub-protocol specific attributes

In the SDP, each data channel declaration MAY also be followed by other SDP attributes specific to the sub-protocol in use. Each of these attributes is represented by one new attribute line, and it

includes the contents of a media-level SDP attribute already defined for use with this (sub)protocol in another IETF specification. Sub-protocol-specific attributes might also be defined for exclusive use with data channel transport, but should use the same syntax described here for other sub-protocol-specific attributes.

Each sub-protocol specific SDP attribute that would normally be used to negotiate the subprotocol using SDP is replaced with an attribute of the form "a=dcsa: stream-id original-attribute", where dcsa stands for "data channel sub-protocol attribute", stream-id is the sctp stream identifier assigned to this sub-protocol instance, and original-attribute represents the contents of the sub-protocol related attribute to be included.

Formal Syntax:

Name: dcsa

Value: dcsa-value

Usage Level: media

Charset Dependent: no

Syntax:

```
dcsa-value      = stream-id SP attribute
attribute       = <from-RFC4566>
```

Examples:

a=dcsa:2 accept-types:text/plain

Thus in the example above, the original attribute line "a=accept-types:text/plain" is represented by the attribute line "a=dcsa:2 accept-types:text/plain", which specifies that this instance of MSRP being transported on the sctp association using the data channel with stream id 2 accepts plain text files. The above example creates a reliable, ordered data channel.

As opposed to the data channel setup parameters, these parameters are subject to offer/answer negotiation following the procedures defined in the sub-protocol specific documents.

The same syntax applies to any other SDP attribute required for negotiation of this instance of the sub-protocol.

Note: This document does not provide a complete specification of how to negotiate the use of a data channel to transport MSRP. Procedures specific to each sub-protocol such as MSRP will be documented elsewhere. The use of MSRP is only an example of how the generic procedures described herein might apply to a specific sub-protocol.

5.2. Procedures

5.2.1. Managing stream identifiers

For the SDP-based external negotiation described in this document, the initial offerer based "SCTP over DTLS" owns by convention the even stream identifiers whereas the initial answerer owns the odd stream identifiers. This ownership is invariant for the whole lifetime of the signaling session, e.g. it does not change if the initial answerer sends a new offer to the initial offerer.

This specification allows simultaneous use of external and internal negotiation. However, a single stream is managed using one method at a time. Stream ids that are not currently used in SDP can be used for internal negotiation. Stream id allocation per SDP based external negotiation may not align with DTLS role based allocation. This could cause glare conditions when one side trying to do external negotiation on a stream id while the other end trying to open data channel on the same stream id using internal negotiation. To avoid these glare conditions this specification recommends that the data channel stack user always selects stream ids per SDP offer/answer rule even when internal negotiation is used. To avoid glare conditions, it is possible to come up with a different stream id allocation scheme, but such schemes are outside the scope of this specification.

5.2.2. Opening a data channel

The procedure for opening a data channel using external negotiation starts with the agent preparing to send an SDP offer. If a peer receives an SDP offer before getting to send a new SDP offer with data channels that are to be externally negotiated, or loses an SDP offer glare resolution procedure in this case, it must wait until the ongoing SDP offer/answer completes before resuming the external negotiation procedure.

The agent that intends to send an SDP offer to create data channels through SDP-based external negotiation performs the following:

- Creates data channels using stream identifiers from the owned set (see Section 5.2.1).

- As described in Section 4.2.2, if the SCTP association is not yet established, then the newly created data channels are in the Connecting state, else if the SCTP association is already established, then the newly created data channels are in the Open state.

- Generates a new SDP offer. In the case of the browser based applications the browser generates the offer via the createOffer() API call [I-D.ietf-rtcweb-jsep].

- Determines the list of stream identifiers assigned to data channels opened through external negotiation.

- Completes the SDP offer with the dcmap and dcsa attributes needed, if any, for each externally-negotiated data channel, as described in Section 5.1.

- Sends the SDP offer.

The peer receiving such an SDP offer performs the following:

- Applies the SDP offer. Note that the browser ignores data channel specific attributes in the SDP.

- Analyzes the channel parameters and sub-protocol attributes to determine whether to accept each offered data channel.

- For accepted data channels, creates peer instances for the data channels with the browser using the channel parameters described in the SDP offer. Note that the browser is asked to create data channels with stream identifiers not "owned" by the agent.

- As described in Section 4.2.2, if the SCTP association is not yet established, then the newly created data channels are in the Connecting state, else if the SCTP association is already established, then the newly created data channels are in the Open state.

- Generates an SDP answer.

- Completes the SDP answer with the dcmap and optional dcsa attributes needed for each externally-negotiated data channel, as described in Section 5.1.

- Sends the SDP answer.

The agent receiving such an SDP answer performs the following:

- Closes any created data channels (whether in Connecting or Open state) for which the expected dcmap and dcsa attributes are not present in the SDP answer.

- Applies the SDP answer.

Any data channels in Connecting state are transitioned to the Open state when the SCTP association is established.

Each agent application MUST wait to send data until it has confirmation that the data channel at the peer is in the Open state. For webrtc, this is when both data channel stacks have channel parameters instantiated. This occurs:

- At both peers when a data channel is created without an established SCTP association, as soon as the data channel stacks report that the data channel transitions to the Open state from the Connecting state.

- At the agent receiving an SDP offer for which there is an established SCTP association, as soon as it creates an externally negotiated data channel in the Open state based on information signaled in the SDP offer.

- At the agent sending an SDP offer to create a new externally negotiated data channel for which there is an established SCTP association, when it receives the SDP answer confirming acceptance of the data channel or when it begins to receive data on the data channel from the peer, whichever occurs first.

5.2.3. Closing a data channel

When the application requests the closing of a data channel that was externally negotiated, the data channel stack always performs an in-band SSN reset for this channel.

It is specific to the sub-protocol whether this closing must in addition be signaled to the peer via a new SDP offer/answer exchange.

A data channel can be closed by sending a new SDP offer which excludes the dcmap and dcsa attributes lines for the data channel. The port value for the m line should not be changed (e.g., to zero) when closing a data channel (unless all data channels are being closed and the SCTP association is no longer needed), since this would close the SCTP association and impact all of the data channels. If answerer accepts the SDP offer then it MUST also exclude the corresponding attribute lines in the answer. In addition to that, SDP answerer may exclude other data channels which were closed but

not yet communicated to the peer. So, offerer MUST inspect the answer to see if it has to close other data channels which are now not included in the answer

If a new SDP offer/answer is used to close data channels then the data channel(s) should only be closed by the answerer/offerer after successful SDP answer is sent/received.

This delayed close is to handle cases where a successful SDP answer is not received, in which case the state of session should be kept per the last successful SDP offer/answer.

If a client receives a data channel close indication (due to inband SSN reset or some other reason) without associated SDP offer then an SDP offer which excludes this closed data channel SHOULD be generated.

The application must also close any data channel that was externally negotiated, for which the stream identifiers are not listed in an incoming SDP offer.

A closed data channel using local close (SCTP reset), without an additional SDP offer/answer to close it, may be reused for a new data channel. This can only be done via new SDP offer/answer, describing the new sub-protocol and its attributes, only after the corresponding data channel close acknowledgement is received from the peer (i.e. SCTP reset of both incoming and outgoing streams is completed). This restriction is to avoid the race conditions between arrival of "SDP offer which reuses stream" with "SCTP reset which closes outgoing stream" at the peer

5.2.4. Various SDP offer/answer scenarios and considerations

SDP offer has no a=dcmap attributes

* Initial SDP offer: No data channel negotiated yet.

* Subsequent SDP offer: All the externally negotiated data channels must be closed now. The DTLS/SCTP association remains open for external or internal negotiation of data channels.

SDP answer has no a=dcmap attributes

* Initial SDP answer: Either the peer does not support dcmap attributes or it rejected all the data channels. In either case offerer closes all the externally negotiated data channels that were open at the time of initial offer. The DTLS/SCTP association will still be setup.

* Sub-sequent SDP answer: All the externally negotiated data channels must be closed now. The DTLS/SCTP association remains open for future external or internal negotiation of data channels.

SDP offer has no a=dcsa attributes for a data channel.

* This is allowed and indicates there are no sub-protocol parameters to convey.

SDP answer has no a=dcsa attributes for a data channel.

* This is allowed and indicates there are no sub-protocol parameters to convey in the SDP answer. The number of dcsa attributes in the SDP answer does not have to match the number of dcsa attributes in the SDP offer.

6. Examples

```
SDP offer:
  m=application 10001 DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.1
  a=fmtp:webrtc-datachannel max-message-size=100000
  a=sctp-port 5000
  a=setup:actpass
  a=connection:new
  a=fingerprint:SHA-1 \
      4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
  a=dcmap:0 subprotocol="BFCP";label="BGCP"

SDP answer:
  m=application 10002 DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.2
  a=fmtp:webrtc-datachannel max-message-size=100000
  a=sctp-port 5002
  a=setup:passive
  a=connection:new
  a=fingerprint:SHA-1 \
      5B:AD:67:B1:3E:82:AC:3B:90:02:B1:DF:12:5D:CA:6B:3F:E5:54:FA
```

Figure 1: Example 1

In the above example the SDP answerer rejected the data channel with stream id 0 either for explicit reasons or because it does not understand the a=dcmap attribute. As a result the offerer will close the data channel created with the external negotiation option. The SCTP association will still be setup over DTLS. At this point

offerer or answerer may use internal negotiation to open data channels.

```
SDP offer:
  m=application 10001 DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.1
  a=fmtp:webrtc-datachannel max-message-size=100000
  a=sctp-port 5000
  a=setup:actpass
  a=connection:new
  a=fingerprint:SHA-1 \
      4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
  a=dcmap:0 subprotocol="BFCP";label="BGCP"
  a=dcmap:2 subprotocol="MSRP";label="MSRP"
  a=dcsa:2 accept-types:message/cpim text/plain text/
  a=dcsa:2 path:msrp://alice.example.com:10001/2s93i93idj;dc

SDP answer:
  m=application 10002 DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.2
  a=fmtp:webrtc-datachannel max-message-size=100000
  a=sctp-port 5002
  a=setup:passive
  a=connection:new
  a=fingerprint:SHA-1 \
      5B:AD:67:B1:3E:82:AC:3B:90:02:B1:DF:12:5D:CA:6B:3F:E5:54:FA
  a=dcmap:2 subprotocol="MSRP";label="MSRP"
  a=dcsa:2 accept-types:message/cpim text/plain
  a=dcsa:2 path:msrp://bob.example.com:10002/si438dsaodes;dc
```

Figure 2: Example 2

In the above example SDP offer contains data channels for BFCP and MSRP sub-protocols. SDP answer rejected BFCP and accepted MSRP. So, the offerer should close the data channel for BFCP and both offerer and answerer may start using MSRP data channel (after SCTP/DTLS association is setup). The data channel with stream id 0 is free and can be used for future internal or external negotiation.

Continuing on the earlier example in Figure 1.

```
Subsequent SDP offer:
  m=application 10001 DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.1
  a=fmtp:webrtc-datachannel max-message-size=100000
  a=sctp-port 5000
  a=setup:actpass
  a=connection:existing
  a=fingerprint:SHA-1 \
      4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
  a=dcmap:4 subprotocol="MSRP";label="MSRP"
  a=dcsa:4 accept-types:message/cpim text/plain
  a=dcsa:4 path:msrp://alice.example.com:10001/2s93i93idj;dc

Subsequent SDP answer:
  m=application 10002 DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.2
  a=fmtp:webrtc-datachannel max-message-size=100000
  a=sctp-port 5002
  a=setup:passive
  a=connection:existing
  a=fingerprint:SHA-1 \
      5B:AD:67:B1:3E:82:AC:3B:90:02:B1:DF:12:5D:CA:6B:3F:E5:54:FA
  a=dcmap:4 subprotocol="MSRP";label="MSRP"
  a=dcsa:4 accept-types:message/cpim text/plain
  a=dcsa:4 path:msrp://bob.example.com:10002/si438dsaodes;dc
```

Figure 3: Example 3

The above example is a continuation of the example in Figure 1. The SDP offer now removes the MSRP data channel with stream id 2, but opens a new MSRP data channel with stream id 4. The answerer accepted the entire offer. As a result the offerer closes the earlier negotiated MSRP related data channel and both offerer and answerer may start using new the MSRP related data channel.

7. Security Considerations

No security considerations are envisaged beyond those already documented in [RFC4566]

8. IANA Considerations

To be completed.

9. Acknowledgments

The authors wish to acknowledge the borrowing of ideas from other internet drafts by Salvatore Loreto, Gonzalo Camarillo, Peter Dunkley and Gavin Llewellyn, and to thank Paul Kyzivat, Jonathan Lennox, and Uwe Rauschenbach for their invaluable comments.

10. CHANGE LOG

10.1. Changes against '-01'

- Formal syntax for dcmap and dcsa attribute lines.
- Making subprotocol as an optional parameter in dcmap.
- Specifying disallowed parameter combinations for max-time and max-retr.
- Clarifications on data channel close procedures.

10.2. Changes against '-00'

- Revisions to identify difference between internal and external negotiation and their usage.
- Introduction of more generic terminology, e.g. "application" instead of "browser".
- Clarification of how "max-retr and max-time affect the usage of unreliable and reliable data channels.
- Updates of examples to take into account the SDP syntax changes introduced with draft-ietf-mmusic-sctp-sdp-07.
- Removal of the SCTP port number from the a=dcmap and a=dcsa attributes as this is now contained in the a=sctp-port attribute, and as draft-ietf-mmusic-sctp-sdp-07 supports only one SCTP association on top of the DTLS connection.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

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

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[I-D.ietf-rtcweb-jsep]
Uberti, J., Jennings, C., and E. Rescorla, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-07 (work in progress), July 2014.

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-12 (work in progress), September 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-07 (work in progress), July 2014.

[WebRtcAPI]
Bergkvist, A., Burnett, D., Jennings, C., and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD-webrtc-20130910, September 2013, <http://www.w3.org/TR/2013/WD-webrtc-20130910/>.

11.2. Informative References

[I-D.ietf-rtcweb-data-protocol]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Establishment Protocol", draft-ietf-rtcweb-data-protocol-08 (work in progress), September 2014.

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

[RFC4976] Jennings, C., Mahy, R., and A. Roach, "Relay Extensions for the Message Sessions Relay Protocol (MSRP)", RFC 4976, September 2007.

[RFC5547] Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S., and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", RFC 5547, May 2009.

[RFC6135] Holmberg, C. and S. Blau, "An Alternative Connection Model for the Message Session Relay Protocol (MSRP)", RFC 6135, February 2011.

[RFC6714] Holmberg, C., Blau, S., and E. Burger, "Connection Establishment for Media Anchoring (CEMA) for the Message Session Relay Protocol (MSRP)", RFC 6714, August 2012.

Authors' Addresses

Keith Drage (editor)
Alcatel-Lucent
Quadrant, Stonehill Green, Westlea
Swindon
UK

Email: keith.drage@alcatel-lucent.com

Raju Makaraju
Alcatel-Lucent
2000 Lucent Lane
Naperville, Illinois
US

Email: Raju.Makaraju@alcatel-lucent.com

Juergen Stoetzer-Bradler
Alcatel-Lucent
Lorenzstrasse 10
D-70435 Stuttgart
Germany

Email: Juergen.Stoetzer-Bradler@alcatel-lucent.com

Richard Ejzak
Unaffiliated

Email: richard.ejzak@gmail.com

Jerome Marcon
Unaffiliated

CLUE
Internet-Draft
Intended status: Informational
Expires: August 22, 2013

C. Groves, Ed.
W. Yang
R. Even
Huawei

February 18, 2013

CLUE media capture description
draft-groves-clue-capture-attr-01

Abstract

This memo discusses how media captures are described and in particular the content attribute in the current CLUE framework document and proposes several alternatives.

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 August 22, 2013.

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.

1. Introduction

One of the fundamental aspects of the CLUE framework is the concept of media captures. The media captures are sent from a provider to a consumer. This consumer then selects which captures it is interested in and replies back to the consumer. The question is how does the consumer choose between what may be many different media captures?

In order to be able to choose between the different media captures the consumer must have enough information regarding what the media capture represents and to distinguish between the media captures.

The CLUE framework draft currently defines several media capture attributes which provide information regarding the capture. The draft indicates that Media Capture Attributes describe static information about the captures. A provider uses the media capture attributes to describe the media captures to the consumer. The consumer will select the captures it wants to receive. Attributes are defined by a variable and its value.

One of the media capture attributes is the content attribute. As indicated in the draft it is a field with enumerated values which describes the role of the media capture and can be applied to any media type. The enumerated values are defined by RFC 4796 [RFC4796]. The values for this attribute are the same as the mediacnt values for the content attribute in RFC 4796 [RFC4796]. This attribute can have multiple values, for example content={main, speaker}.

RFC 4796 [RFC4796] defines the values as:

> slides: the media stream includes presentation slides. The media type can be, for example, a video stream or a number of instant messages with pictures. Typical use cases for this are online seminars and courses. This is similar to the 'presentation' role in H.239.
>
> speaker: the media stream contains the image of the speaker. The media can be, for example, a video stream or a still image. Typical use cases for this are online seminars and courses.
>
> sl: the media stream contains sign language. A typical use case for this is an audio stream that is translated into sign language, which is sent over a video stream.

Whilst the above values appear to be a simple way of conveying the content of a stream the Contributors believe that there are multiple issues that make the use of the existing "Content" tag insufficient for CLUE and multi-stream telepresence systems. These issues are

described in section 3. Section 4 proposes new capture description attributes.

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].

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

3. Issues with Content attribute

3.1. Ambiguous definition

*There is ambiguity in the definitions that may cause problems for interoperability. A clear example is "slides" which could be any form of presentation media. Another example is the difference between "main" and "alt". In a telepresence scenario the room would be captured by the "main cameras" and a speaker would be captured by an alternative "camera". This runs counter with the definition of "alt".

Another example is a university use case where:

The main site is a university auditorium which is equipped with three cameras. One camera is focused on the professor at the podium. A second camera is mounted on the wall behind the professor and captures the class in its entirety. The third camera is co-located with the second, and is designed to capture a close up view of a questioner in the audience. It automatically zooms in on that student using sound localization.

For the first camera, it’s not clear whether to use "main" or "speaker". According to the definition and example of "speaker" in RFC 4796 [RFC4796], maybe it’s more proper to use "speaker" here? For the third camera it could fit the definition of "main" or "alt" or"speaker".

3.2. Multiple functions

It appears that the definitions cover disparate functions. "Main" and "alt" appear to describe the source from which media is sent. "Speaker" indicates a role associated with the media stream.

"Slides" and "Sign Language" indicates the actual content. Also indirectly some prioritization is applied to these parameters. For example: the IMTC document on best practices for H.239 indicates a display priority between "main" and "alt". This mixing of functions per code point can lead to ambiguous behavior and interoperability problems. It also is an issue when extending the values.

3.3. Limited Stream Support

The values above appear to be defined based on a small number of video streams that are typically supported by legacy video conferencing. E.g. a main video stream (main), a secondary one (alt) and perhaps a presentation stream (slides). It is not clear how this value scales when many media streams are present. For example if you have several main streams and several presentation streams how would an endpoint distinguish between them?

3.4. Insufficient information for individual parameters

Related to the above point is that some individual values do not provide sufficient information for an endpoint to make an educated decision on the content. For example: Sign language (sl) - If a conference provides multiple streams each one containing a sign interpretation in a different sign language how does an endpoint distinguish between the languages if "sl" is the only label? Also for accessible services other functions such a real time captioning and video description where an additional audio channel is used to describe the conference for vision impaired people should be supported.

Note: SDP provide a language attribute.

3.5. Insufficient information for negotiation

CLUE negotiation is likely to be at the start of a session initiation. At this point of time only a very simple set of SDP (i.e. limited media description) may be available (depending on call flow). In most cases the supported media captures may be agreed upon before the full SDP information for each media stream. The effect of this is that detailed information would not be available for the initial decision about which capture to choose. The obvious solution is to provide "enough" data in the CLUE provider messages so that a consumer can choose the appropriate media captures. The current CLUE framework already partly addresses this through the "Content" attribute however based on the current "Content" values it appears that the information is not sufficient to fully describe the content of the captures.

The purpose of the CLUE work is to supply enough information for negotiating multiple streams. CLUE framework [I-D.ietf-clue-framework] addresses the spatial relation between the streams but it looks like it does not provide enough information about the semantic content of the stream to allow interoperability.

Some information is available in SDP and may be available before the CLUE exchange but there are still some information missing.

4. Capture description attributes

As indicated above it is proposed to introduce a new attribute/s that allows the definition of various pieces of information that provide metadata about a particular media capture. This information should be described in a way that it only supplies one atomic function. It should also be applicable in a multi-stream environment. It should also be extensible to allow new information elements to be introduced in the future.

As an initial list the following attributes are proposed for use as metadata associated with media captures. Further attributes may be identified in the future.

This document propose to remove the "Content" attribute. Rather than describing the "source device" in this way it may be better to describe its characteristics. i.e.

An attribute to indicate "Presentation" rather than the value "Slides".

An attribute to describe the "Role" of a capture rather than the value "Speaker".

An attribute to indicate the actual language used rather than a value "Sign Language". This is also applicable to multiple audio streams.

With respect to "main" and "alt" in a multiple stream environment it’s not clear these values are needed if the characteristics of the capture are described. An assumption may be that a capture is "main" unless described otherwise.

Note: CLUE may have missed a media type "text". How about a real time captioning or a real time text conversation associated with a video meeting? It’s a text based service. It’s not necessarily a presentation stream. It’s not audio or visual but a valid component of a conference.

The sections below contain an initial list of attributes.

4.1. Presentation

This 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.

4.2. View

The Area of capture attribute provides a physical indication of a region that the media capture captures. However the consumer does not know what this physical region relates to. In discussions on the IETF mailing list it is apparent that some people propose to use the "Description" attribute to describe a scene. This is a free text field and as such can be used to signal any piece of information. This leads to problems with interoperability if this field is automatically processed. For interoperability purposes it is proposed to introduce a set of keywords that could be used as a basis for the selection of captures. It is envisaged that this list would be extendable to allow for future uses not covered by the initial specification. Therefore it is proposed to introduce a number of keywords (that may be expanded) indicating what the spatial region relates to? I.e. Room, table, etc. this is an initial description of an attribute introducing these keywords.

This attribute provides a textual description of the area that a media capture captures. This provides supplementary information in addition to the spatial information (i.e. area of capture) regarding the region that is captured.

Room - Captures the entire scene.

Table - Captures the conference table with seated participants

Individual - Captures an individual participant

Lectern - Captures the region of the lectern including the presenter in classroom style conference

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

Others - TBD

4.3. Language

Captures may be offered in different languages in case of multi-lingual and/or accessible conferences. It is important to allow the remote end to distinguish between them. It is noted that SDP already contains a language attribute however this may not be available at the time that an initial CLUE message is sent. Therefore a language attribute is needed in CLUE to indicate the language used by the capture.

This indicates which language is associated with the capture. For example: it may provide a language associated with an audio capture or a language associated with a video capture when sign interpretation or text is used.

An example where multiple languages may be used is where a capture includes multiple conference participants who use different languages.

The possible values for the language tag are the values of the 'Subtag' column for the "Type: language" entries in the "Language Subtag Registry" defined in RFC 5646 [RFC5646].

4.4. Role

The original definition of "Content" allows the indication that a particular media stream is related to the speaker. CLUE should also allow this identification for captures. In addition with the advent of XCON there may be other formal roles that may be associated with media/captures. For instance: a remote end may like to always view the floor controller. It is envisaged that a remote end may also chose captures depending on the role of the person/s captured. For example: the people at the remote end may wish to always view the chairman. This indicates that the capture is associated with an entity that has a particular role in the conference. It is possible for the attribute to have multiple values where the capture has multiple roles.

The values are grouped into two types: Person roles and Conference Roles

4.4.1. Person Roles

The roles are related to the titles of the person/s associated with the capture.

Manager - Indicates that the capture is assigned to a person with a senior position.

Chairman- indicates who the chairman of the meeting is.

Secretary - indicates that the capture is associated with the conference secretary.

Lecturer - indicates that the capture is associated with the conference lecturer.

Audience - indicates that the capture is associated with the conference audience.

Others

4.4.2. Conference Roles

These roles are related to the establishment and maintenance of the multimedia conference and is related to the conference system.

Speaker - indicates that the capture relates to the current speaker.

Controller - indicates that the capture relates to the current floor controller of the conference.

Others

An example is:

```
AC1 [Role=Speaker]
VC1 [Role=Lecturer,Speaker]
```

4.5. Priority

As has been highlighted in discussions on the CLUE mailing list there appears to be some desire to provide some relative priority between captures when multiple alternatives are supplied. This priority can be used to determine which captures contain the most important information (according to the provider). This may be important in case where the consumer has limited resources and can on render a subset of captures. Priority may also be advantageous in congestion scenarios where media from one capture may be favoured over other captures in any control algorithms. This could be supplied via "ordering" in a CLUE data structure however this may be problematic if people assume some spatial meaning behind ordering, i.e. given three captures VC1, VC2, VC3: it would be natural to send VC1,VC2,VC3

if the images are composed this way. However if your boss sits in the middle view the priority may be VC2,VC1,VC3. Explicit signalling is better.

Additionally currently there are no hints to relative priority among captures from different capture scenes. In order to prevent any misunderstanding with implicit ordering a numeric number that may be assigned to each capture.

The "priority" attribute 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 as important. If no priority is assigned no assumptions regarding relative important of the capture can be assumed.

4.6. Others

4.6.1. Dynamic

In the framework it has been assumed that the capture point is a fixed point within a telepresence session. However depending on the conference scenario this may not be the case. In tele-medical or tele-education cases a conference may include cameras that move during the conference. For example: a camera may be placed at different positions in order to provide the best angle to capture a work task, or may include a camera worn by a participant. This would have an effect of changing the capture point, capture axis and area of capture. In order that the remote endpoint can chose to layout/ render the capture appropriately an indication of if the camera is dynamic should be indicated in the initial capture description.

This indicates that the spatial information related to the capture may be dynamic and change through the conference. Thus captures may be characterised as static, dynamic or highly dynamic. The capture point of a static capture does not move for the life of the conference. The capture point of dynamic captures is categorised by a change in position followed by a reasonable period of stability. High dynamic captures are categorised by a capture point that is constantly moving. This may assist an endpoint in determining the correct display layout. 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 a CLUE message is sent. No information regarding future spatial information

should be assumed.

4.6.2. Embedded Text

In accessible conferences textual information may be added to a capture before it is transmitted to the remote end. In the case where multiple video captures are presented the remote end may benefit from the ability to choose a video stream containing text over one that does not.

This 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.

The EmbeddedText attribute contains a language value according to RFC 5646 [RFC5646] and may use a script sub-tag. For example:

EmbeddedText=zh-Hans

Which indicates embedded text in Chinese written using the simplified Chinese script.

4.6.3. Complementary Feed

Some conferences utilise translators or facilitators that provide an additional audio stream (i.e. a translation or description of the conference). These persons may not be pictured in a video capture. Where multiple audio captures are presented it may be advantageous for an endpoint to select a complementary stream instead of or additional to an audio feed associated with the participants from a main video capture.

This indicates that a capture provides additional description of the conference. For example an additional audio stream that provides a commentary of a conference that provides complementary information (e.g. a translation) or extra information to participants in accessible conferences.

An example is where an additional capture provides a translation of another capture:

AC1 [Language = English]
AC2 [ComplementaryFeed = AC1, Language=Chinese]

The complementary feed attribute indicates the capture to which it is providing additional information.

5. Summary

The main proposal is a to remove the Content Attribute in favour of describing the characteristics of captures in a more functional(atomic) way using the above attributes as the attributes to describe metadata regarding a capture.

6. Acknowledgements

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

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

TBD

9. Changes and Status Since Last Version

Changes from 00 to 01:

1. Changed source to XML.

2. 4.1 Presentation : No comments or concerns. No changes.

3. 4.2 View : No comments or concerns. No changes.

4. 4.3 Language: There were comments that multiple languages need to be supported e.g. audio in one, embedded text in another. The text need to be clear whether it is supported or preferred language however it was clarified it is neither. Its the language of the content/capture. It was also noted that different speakers using different languages could talk on the main speakers capture therefore language should be a list. Seemed to be support for this. Text was adapted accordingly.

5. 4.4 Role: There were a couple of responses for support for this attribute. The actual values still need some work. It was noted that there were two possible sets of roles: One group related to the titles of the person: i.e. Boss,

Chairman, Secretary, Lecturer, Audience. Another group related to conference functions: i.e. Conference initiator, controller, speaker. Text was adapted accordingly.

6. 4.5 Priority: No direct comment on the proposal. There appeared to be some interest in a prioritisation scheme during discussions on the framework. No changes.

7. 4.6.1 Dynamic : No comments or concerns. No changes.

8. 4.6.2 Embedded text: There was a comment that "text" media capture was needed. It was also indicated that it should be possible to associate a language with embedded text. It should be possible to also specify language and script. e.g. Embedded text could have its own language. Text adapted accordingly.

9. 4.6.3 Supplementary Description: There were comments that it could be interpreted as a free text field. The intention is that its more of a flag. A better name could be "Complementary feed"? There was also a comment that perhaps a specific "translator flag" is needed. It was noted the usage was like: AC1 Language=English or AC2 Supplementary Description = TRUE, Language=Chinese. Text updated accordingly.

10. 4.6.4 Telepresence There were a couple of comments questioning the need for this parameter. Attribute removed.

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.

10.2. Informative References

[I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-08 (work in progress).

[RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.

[RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.

Authors' Addresses

Christian Groves (editor)
Huawei
Melbourne,
Australia

Email: Christian.Groves@nteczone.com

Weiwei Yang
Huawei
P.R.China

Email: tommy@huawei.com

Roni Even
Huawei
Tel Aviv,
Isreal

Email: roni.even@mail01.huawei.com

CLUE
Internet-Draft
Intended status: Informational
Expires: July 30, 2014

C. Groves, Ed.
W. Yang
R. Even
Huawei

January 26, 2014

CLUE and latent configurations
draft-groves-clue-latent-config-00

Abstract

This document proposes to use Latent Configurations as described by the SDP media capability negotiation framework [RFC6871] for the description and negotiation of CLUE encodings.

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 July 30, 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. Latent Configurations and CLUE 3
2.1. Semantics 3
2.2. Messaging 3
2.3. Correlation 4
2.4. Returning an Answer 5
2.5. Interworking 5
2.6. Relation to BUNDLE 6
2.7. Open Issues 6
3. Example 7
4. Summary 11
5. Acknowledgements 11
6. IANA Considerations 11
7. Security Considerations 11
8. References 11
8.1. Normative References 11
8.2. Informative References 11
Authors' Addresses 12

1. Introduction

One of the issues faced in CLUE is how to describe the encodings associated with the Advertised Captures. It was recently decided that this encoding information would not be described in CLUE itself. This means that other methods such as the use of SDP are required to transmit this encoding information. When considering the use of SDP (and in particular the use of SDP Offer/Answer) it should be recognized that there is a semantic difference between a CLUE encoding and an SDP media stream description. Given the nature of a CLUE exchange the encoding represents a Capture/Stream that may occur in the future (i.e. no resources are reserved) whereas a SDP Offer typically relates to resources that are available at that point in time. This does lead to complications when trying to describe CLUE encodings using standard SDP O/A mechanisms.

An alternate approach to using the standard SDP O/A mechanisms for describing CLUE encodings is to use the "SDP Capability Negotiation framework" [RFC5939] and in particular to use the SDP Media Capabilities Negotiation [RFC6871]. [RFC6871] defines "Latent Configurations" as a means to describe media streams that may be used in the future.

2. Latent Configurations and CLUE

The sections below discuss different aspects and benefits of using Latent Configurations to describe CLUE encodings.

2.1. Semantics

[RFC6501] defines a Latent Configuration as:

> A latent configuration indicates which combinations of capabilities could be used in a future negotiation for the session and its associated media stream components. Latent configurations are neither ready for use nor offered for actual or potential use in the current offer/answer exchange. Latent configurations merely inform the other side of possible configurations supported by the entity. Those latent configurations may be used to guide subsequent offer/answer exchanges, but they are not offered for use as part of the current offer/answer exchange.

From the above description it can be seen that the semantic of a Latent Configurations closely matches a CLUE message flow. I.e. A set of possible Captures/Encodings (e.g. configurations) are Advertised, the receiver can choose particular Captures/Encodings and then the actual media stream is subsequently established. Therefore the authors believe that use of latent configurations is a good semantic fit with CLUE to describe the encodings.

2.2. Messaging

It has been recognized that CLUE Advertisements may contain a large number of Captures and as there may be multiple encodings per capture potentially a larger number of encodings.

Section 4.2 of CLUE signaling draft [I-D.kyzivat-clue-signaling] indicates the CLUE Provider uses an "m" line for each separate encoding and utilizes the "a=inactive", "a=sendonly" and "a=recvonly" to manage when the media flows are instantiated.

This means that ports must be allocated for these "m" lines and the SDP Offer/Answer [RFC3264] rules regarding maintaining these "m" lines must be followed.

This results in potentially very large SDP descriptions containing superfluous information that must be maintained for the life of the session.

Latent Configurations allow a Provider to advertise potential media without allocating multiple "m" lines or allocating ports for the configurations. The SDP O/A model doesn't apply to Latent Configurations which means that less data is sent over the life of a session.

This allows a Provider to offer a basic media stream for immediate use (i.e. an audio "m" line) and a list of latent configurations for later use without the need for additional m-lines. This is described by [RFC6871]:

> A new attribute ("a=lcfg") specifies latent media stream configurations when no corresponding media line ("m=") is offered. An example is the offer of latent configurations for video even though no video is currently offered. If the peer indicates support for one or more offered latent configurations, the corresponding media stream(s) may be added via a new offer/ answer exchange.
>
> AND
>
> The "lcfg" attribute is defined as a media-level attribute since it specifies a possible future media stream. However, the "lcfg" attribute is not necessarily related to the media description within which it is provided. Each media line in an SDP description represents an offered simultaneous media stream, whereas each latent configuration represents an additional stream that may be negotiated in a future offer/answer exchange.

The use of Latent Configurations also means that resources aren't tied up and can be allocated when they are needed. i.e. from [RFC6871]:

> A latent configuration represents a future capability; hence, the "pt=" parameter is not directly meaningful in the "lcfg" attribute because no actual media session is being offered or accepted. It is permitted in order to tie any payload type number parameters within attributes to the proper media format.

Therefore the authors believe that Latent Configurations provide a clear benefit in terms of messaging size and complexity over normal SDP Offer/Answer mechanism for Advertising CLUE encodings.

2.3. Correlation

One of the issues recognized in the CLUE work it that there needs to be a correlation between the Captures signaled in CLUE and the encodings defined in SDP. The encoding identity (EncodeID) is used

as an identity in CLUE. This same identity is then assigned to the encoding in SDP. Currently [I-D.kyzivat-clue-signaling] indicates that the label attribute [RFC4574] is used to identify the encoding and that it is set per "m" line.

This same method can be used with Latent configurations as they allow the use of SDP attributes in the configurations' description. Section 3.3.8/[RFC6871] shows an example of the use of "label" with a latent configuration, e.g.

```
a=lcfg:4 mt=video t=1 m=1 a=41,42
a=acap:41 label:13
a=acap:42 content:slides
```

The use of Latent Configurations does not require any new SDP attributes to be defined in order for it to be used for CLUE encodings.

2.4. Returning an Answer

A consumer upon receiving an SDP Offer containing CLUE related latent configurations from the Provider could immediately send an SDP answer with the configurations that it could support, i.e. section 3.3.6.1/ [RFC6871]:

> Potential and/or latent configuration attributes may be returned within an answer SDP to indicate the ability of the answerer to support alternative configurations of the corresponding stream(s).

The consumer would then send a CLUE Configure indicating the Capture Encodings it wants. The Provider can then subsequently offer actual media streams for the encodings.

2.5. Interworking

One aspect to be considered is the level of adoption of the SDP media capabilities negotiation framework in network entities. Whilst the framework is not widely deployed it is supported by 3GPP (e.g. [SDO-3GPP.24.292]) and is supported by the ITU-T (e.g. [ITU.H248.80.2014] and [ITU.T38.2010].

It must also be recognized that CLUE itself is something completely new and clients and network equipment must be upgraded to support CLUE signaling. Thus this equipment could also be upgraded to support Latent Configurations at the same time.

In cases where a CLUEfull client sends SDP requesting a CLUE channel and a number of latent configurations to a client that doesn't support CLUE or the media capability framework, the receiving client will ignore the attributes associated with the latent configuration as per normal SDP behavior. Thus there are no interworking issues in this case.

In cases where a CLUEfull client sends SDP requesting a CLUE channel and a number of latent configurations to a client that doesn't support CLUE but DOES support the media capability framework, the receiving client would ignore the CLUE related attributes but could respond with what latent configurations it could support. This would allow the sender to decide if it wanted to offer subsequent media streams. Again there are no interworking issues in this case.

In either of the above cases the session between the clients wouldn't be forced to maintain "m" lines for media that would never be used.

2.6. Relation to BUNDLE

At its core BUNDLE is about using SDP to describe that several "m" lines use the same IP Address/Port for the transport of RTP media. If SDP O/A is being used to describe CLUE encodings then there is a potential interaction in that the CLUE encoding "m" lines would all be subject the BUNDLE procedures whether or not they were actually used for media.

The use of Latent Configurations would simplify this interaction because Latent Configurations do not allocate or specify ports. They would not be subject to BUNDLE procedures. Once the use of BUNDLE is established (i.e. for the base media streams), then only the media streams (Capture Encodings) that have been chosen by the Consumer can be added to the BUNDLE.

2.7. Open Issues

There are several issues that need to be clarified in [RFC6871] with respect to latent configurations.

1. Latent configurations are specified as media level attributes and thus may be associated with any m-line in an SDP O/A as they don't really pertain to a particular media. There appears to be no guidance as to with m-line they should be associated with in the case of multiple m-line in a SDP Offer.

2. It's not clear from [RFC6871] what happens to latent configurations when an endpoint decides to instantiate the latent configuration as an m-line through an updated SDP Offer.

Section 3.4.4/[RFC6871] discusses modifying the session but has minimal information. It is assumed by the authors that the latent configuration is removed once instantiated.

3. It needs to be clarified whether [RFC6871] indicates that the SDP Answerer SHOULD reply with the latent configurations it supports or whether this is optional. If it's optional what does it mean?

4. [RFC6871] allows an SDP Answerer to reply with additional latent configurations. However it doesn't specify what action the SDP Offerer should take. This needs to be clarified.

3. Example

This section describes an example session establishment utilizing CLUE and latent configurations. The Provider offers a base audio stream, a CLUE channel (according to [I-D.ietf-mmusic-sctp-sdp] and several latent configurations related to video captures.

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.anywhere.com
s=
c=IN IP4 host.anywhere.com
t=0 0
m=audio 49170 RTP/AVP 0 ; Base audio stream
a=rtpmap:0 PCMU/8000

; CLUE channel establishment request
m=application 49172 SCTP 49172
a=setup:actpass
a=connection:new
a=sctpmap:49172 CLUE 1

;Offered configurations
a=rmcap:1 H264/90000 ; this is needed for the m=
a=rmcap:2 VP8/90000
a=tcap:1 RTP/AVPF    ; for the t=
a=acap:5 sendonly
a=acap:1 label:encoding1
a=lcfg:1 mt=video t=1 m=1|2 a=1,5 pt=1:100,2:101
a=acap:2 label:encoding2
a=lcfg:2 mt=video t=1 m=1|2 a=2,5 pt=1:102,2:103
a=acap:3 label:encoding3
a=lcfg:3 mt=video t=1 m=1|2 a=3,5 pt=1:104,2:105
a=acap:4 label:encoding4
a=lcfg:4 mt=video t=1 m=1|2 a=4,5 pt=1:106,2:107
```

Figure 1: Initial Offer

The receiver is CLUE capable and responds indicating support of the CLUE channel and indicates the IP Address/Port where the Provider should establish a connection to. It also indicates that it only supports H264 encoding.

```
v=0
o=bob 2890844730 2890844730 IN IP4 host.example.com
s=
c=IN IP4 host.example.com
t=0 0
m=audio 49920 RTP/AVP 0
a=rtpmap:0 PCMU/8000

; Accepted SCTP CLUE connection
m=application 54321 SCTP 54321
c=IN IP4 192.0.2.1
a=setup:passive
a=connection:new
a=sctpmap:54321 CLUE 1

;Accepted configurations
a=rmcap:1 H264/90000 ; this is needed for the m=
a=tcap:1 RTP/AVPF    ; for the t=
a=acap:5 sendonly
a=acap:1 label:encoding1
a=lcfg:1 mt=video t=1 m=1|2 a=1,5 pt=1:100
a=acap:2 label:encoding2
a=lcfg:2 mt=video t=1 m=1|2 a=2,5 pt=1:102
a=acap:3 label:encoding3
a=lcfg:3 mt=video t=1 m=1|2 a=3,5 pt=1:104
a=acap:4 label:encoding4
a=lcfg:4 mt=video t=1 m=1|2 a=4,5 pt=1:106
```

Figure 2: Answer

Author’s note: In the signaling document the grouping framework [RFC5888] is used to indicate the "m" lines that are under CLUE control. It’s not clear whether a=group:CLUE and a=mid is needed at this stage or during the updated-Offer. It could be assumed that the above latent configurations are related to CLUE due to the fact they appear under the m=application SCTP/CLUE line.

On receipt of the SDP Answer the Provider establishes the SCTP connection, performs CLUE version and capability negotiation (not shown) and then sends the initial CLUE Advertisement. In it the Provider advertises a single Capture Scene described by three video captures (i.e. Left,Centre,Right) or a video capture of the entire scene.

Author's note: According to the current CLUE protocol work [I-D.presta-clue-protocol], it's the consumer that sends the first Advertisement. The author believes that the Provider should send the first Advertisement to align with the Offer.

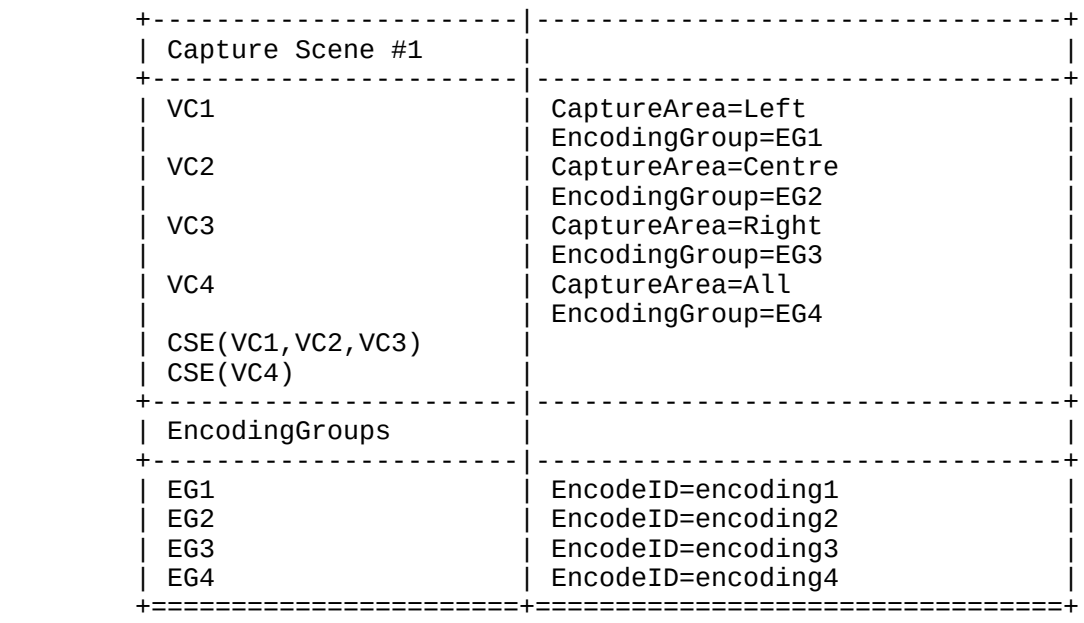


Figure 3: Advertisement

On receipt of the Advertisement the Consumer determines that it wants the three video captures representing left/centre/right. It sends a CLUE Configure to the Provider:

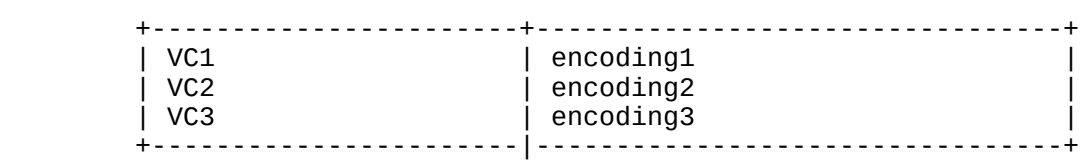


Figure 4: Configure

On receipt of the CLUE Configure the Provider determines that the Consumer wants to see VC1, VC2 and VC3 according to encoding1, encoding2 and encoding3 respectively. The Provider then issues an updated SDP Offer for three additional video streams. Note: The latent configurations have been removed but the latent configuration related to VC4/encoding4 could also be maintained if still available.

The grouping framework [RFC5888] is used to indicate that the additional video streams are under CLUE control.

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.anywhere.com
s=
a=group:CLUE 1 2 3
c=IN IP4 host.anywhere.com
t=0 0
m=audio 49170 RTP/AVP 0 ; Base audio stream
a=rtpmap:0 PCMU/8000

; CLUE channel estab. Req.
m=application 49172 SCTP 49172
a=setup:actpass
a=connection:new
a=sctpmap:49172 CLUE 1

;Additional video streams
m=video 49174 RTP/AVPF 100
a=rtpmap:100 H264/90000
a=label:encoding1
a=mid:1
a=sendonly

m=video 49176 RTP/AVPF 102
a=rtpmap:102 H264/90000
a=label:encoding2
a=mid:2
a=sendonly

m=video 49178 RTP/AVPF 104
a=rtpmap:104 H264/90000
a=label:encoding3
a=mid:3
a=sendonly
```

Figure 5: Updated Offer

The Consumer then receives the Updated Offer and confirms with an updated Answer. Media flow for the 3 video streams then starts to flow.

4. Summary

The authors believe that the use of Latent Configurations is an ideal way to indicate CLUE encoding information. It is proposed that the use of Latent Configuration is the preferred way of describing CLUE encoding information.

5. Acknowledgements

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

6. IANA Considerations

It is not expected that the proposed changes present the need for any IANA registrations.

7. Security Considerations

It is not expected that the proposed changes present any addition security issues to the current framework.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[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-05 (work in progress), October 2013.

[I-D.kyzivat-clue-signaling]
Kyzivat, P., Xiao, L., Groves, C., and R. Hansen, "CLUE Signaling", draft-kyzivat-clue-signaling-05 (work in progress), September 2013.

[I-D.presta-clue-protocol]
Presta, R. and S. Romano, "CLUE protocol", draft-presta-clue-protocol-03 (work in progress), November 2013.

[ITU.H248.80.2014]
International Telecommunications Union, "Usage of the revised SDP offer / answer model with H.248", ITU-T Recommendation H.248.80, 2014.

[ITU.T38.2010]
International Telecommunications Union, "Procedures for real time Group 3 facsimile communication between terminals using IP Networks", ITU-T Recommendation T.38, 2010.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

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

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

[RFC5939] Andreasen, F., "Session Description Protocol (SDP) Capability Negotiation", RFC 5939, September 2010.

[RFC6871] Gilman, R., Even, R., and F. Andreasen, "Session Description Protocol (SDP) Media Capabilities Negotiation", RFC 6871, February 2013.

[SDO-3GPP.24.292]
3GPP, "IP Multimedia (IM) Core Network (CN) subsystem Centralized Services (ICS); Stage 3", 3GPP TS 24.292 10.11.0, June 2013.

Authors' Addresses

Christian Groves (editor)
Huawei
Melbourne
Australia

Email: Christian.Groves@nteczone.com

Weiwei Yang
Huawei
P.R.China

Email: tommy@huawei.com

Roni Even
Huawei
Tel Aviv
Isreal

Email: roni.even@mail01.huawei.com

CLUE
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2013

R. Hansen
Cisco Systems

February 25, 2013

SDP and CLUE message interactions
draft-hansen-clue-sdp-interaction-01

Abstract

This document attempts to help resolve some of the complexities of interaction between SDP and CLUE messages in call flows by providing some strategies and some suggested syntax.

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 August 29, 2013.

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 3
3. Initial Assumptions 3
4. The CLUE framework: dividing the information between SDP and CLUE messaging 4
4.1. CLUE information principally in CLUE channel 4
4.2. Media encoding/decoding information in SDP, media content information in CLUE messaging 4
5. Interdependence of SDP and CLUE negotiation 6
6. Security Considerations 7
7. References 7
7.1. Normative References 7
7.2. Informative References 7
Appendix A. Changes From Draft -00 8
Author's Address 8

1. Introduction

One issue that has repeatedly come up in the development of CLUE is the interconnected nature of many of the issues − making decisions in any one area requires that decisions are made in other areas. One particularly problematic area has been that of producing call flows: many of the decisions that need to be made revolve around how offer/answer exchanges and CLUE messages will interact, but without a good understanding of what will be in SDP and what will be in CLUE these decisions have been difficult to make.

In the hope of resolving some of these issues and allowing us to make more progress on the subject of call flows and CLUE signalling generally this draft addresses two issues that are hopefully not dependent on decisions in other areas, both aspects of the relationship between CLUE signalling and SDP. Hopefully this draft will either provoke discussion, or document decisions that people feel are obvious but aren't currently reflected in writing.

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] and indicate requirement levels for compliant implementations.

3. Initial Assumptions

This section enumerates a few assumptions based on previous discussion which are, at this stage, hopefully uncontroversial.

CLUE information such as capture descriptions are unsuitable for SDP, and as such there will be an alternate method for sending CLUE messages end to end. In a call scenario where both sides wish to send and receive this CLUE negotiation takes the form of two independent, uni−directional exchanges; on each exchange one device provides its send capabilities while the other side determines what it wishes to receive.

This CLUE negotiation will never enable or require a call to exceed boundaries negotiated in SDP. This most obviously applies to bandwidth, both for the total call and for negotiated sessions, but also means that codec−specific limitations such as the maximum number of H.264 macroblocks a second a receiver can process MUST be respected.

4. The CLUE framework: dividing the information between SDP and CLUE messaging

The CLUE Framework [I−D.ietf−clue−framework] defines the information that will be needed to successfully negotiate a CLUE call, but does not define the mechanism by which this information is conveyed. This section provides two options for dividing this information between SDP and CLUE signalling, without proposing explicit signalling for either channel (merely what information needs to be conveyed in each).

4.1. CLUE information principally in CLUE channel

One approach that has been a major part of CLUE discussions has been to make no significant additions to SDP, and continue to use it only for the negotiation of RTP sessions. The sessions are then potentially subdivided into multiple streams using CLUE signalling. In this model standard SDP signalling provides the envelope within which CLUE negotiates the number and content of multiple streams.

This method has a number of advantages − there is no need for additional SDP syntax, making interoperability with existing devices simple and concentrating new signalling in a single location (the CLUE negotiation). There is also clear separation of responsibilities between SDP and CLUE: as normal SDP negotiates the specifics of the RTP sessions: address and ports, supported codecs, receive maxima and so on, while CLUE messaging then specifies how many streams are to be multiplexed on a port, details for demultiplexing, content of those streams, encoding limits and so on. The only necessary addition to the SDP would be a label [RFC4574] attribute per media line to allow CLUE messaging to identify them.

Unfortunately, there are some downsides to this approach. The primary one is that all multiplexing of streams is entirely dependant on the CLUE channel − as such this is not a method applicable to other applications. Since other groups within the IETF have an interest in such multiplexing for reasons other than enabling telepresence scenarios they would have to invent other methods for negotiating similar multiplexing − both inefficient, and likely problematic when CLUE and some other solution involving multistreaming are both used in the call scenario.

4.2. Media encoding/decoding information in SDP, media content information in CLUE messaging

An alternative approach is to divide the information in the CLUE Framework [I−D.ietf−clue−framework] into the information specific to encoding and decoding RTP streams, and the content of those streams.

On the advertising side this split is fairly natural: most of the information in the framework relates to the number, content, physical dimensions and simultaneity of the media captures available, information related to the contents of the media streams rather than the streams themselves. In contrast, the encoder and encoder group information gives the limits on the media streams the sender can provide, with parameters such as bandwidth, max h.264 macroblocks per second and other parameters relevant to SDP. These are defined as sender limitations rather than receiver ones and so are not directly analogous to existing SDP parameters, but are better suited to SDP than CLUE.

When it comes to receiver selection the separation between parameters that logically should be in CLUE and should be in SDP is no longer so clean−cut, as the receiver must specify capture encodings, choosing both which captures they wish to receive and the media limitations on such streams. The latter limitations are obviously suited to SDP, but information about captures is more relevant to the CLUE channel. The CLUE−specific information, however, is limited to simply selecting a capture for the stream.

The ability to describe the sender’s encoder limitations for multiplexed streams along with the receiver’s selection of those streams and the media limitations, SSRCs and other demultiplexing information are all requirements that are not specific to CLUE; having them in SDP means that a consistent mechanism can be used by CLUE as well as by other call scenarios wishing to support additional media streams in this fashion. Capture information, in contrast, is CLUE−specific and as such is sensible to keep in the CLUE channel. The CLUE channel will also reference the SDP, linking captures to encoding capabilities and identifying which capture is desired for each stream. This split of information means that any change in capture information on the part of a sender does not necessitate an offer/answer exchange of SDP if there is no corresponding change to the encoding capabilities of that sender − only a new CLUE advertisement is required.

This approach leads to a number of dependencies between the SDP and CLUE messages − the sender must define which captures and capture scenes are usable with which streams/encodings, while the receiver must define what capture they wish to receive with a particular encoding. These could take the form of references in SDP to CLUE, references in CLUE to SDP or references in each to the other. However, this draft proposes that all such references MUST be from CLUE messages to SDP, not the other way around. By ensuring all depenencies are unidirectional it reduces the complexity of integrating the two signalling methods. There are multiple reasons for having references be CLUE−>SDP and not the other way around: one

is that logically CLUE is providing metadata about the contents of streams that are negotiated in SDP, so it makes sense for CLUE to be dependent on SDP and not visa-versa. Another is that middle boxes wishing to monitor or alter SDP can then do so without necessarily needing to involve themselves in the CLUE channel as SDP remains self-contained.

5. Interdependence of SDP and CLUE negotiation

With separate negotiation of SDP and CLUE there is the question of how to deal with dependencies between these two channels. The number of dependancies depends on how the information defined in the CLUE Framework [I-D.ietf-clue-framework] is split between SDP and CLUE, as discussed in the previous section, but even in the case where all new information is in CLUE there will still be some dependencies as it will be necessary to determine which m-lines the CLUE signalling is referring to. However, because we have two signalling methods changes that require alterations in both CLUE and SDP are no longer atomic: one message will be processed before the other. There has been debate within the working group about how this will be dealt with, as such a decision has significant effect upon call flows.

This draft proposes that CLUE messages and SDP messages should be independent: parameters in CLUE messages MAY exceed values negotiated in SDP, or may make reference to SDP contents not present in the most recent offer/answer exchange. Without this provision, SDP and CLUE messages become part of a single negotiation, and a change on either by either side may necessitate an exchange of the other message type. For instance, removing stream information from SDP might first necessitate sending a new CLUE message removing the references to this stream. The state machine required to ensure validity of negotiation will be complicated, and there will be a number of invalid states which must be avoided. This is further complicated by the fact that, even if both ends of a call obey the constraints to ensure validity, a middle box may choose to rewrite an SDP such that an invalid state is reached.

Making the two message types independent significantly reduces the complexity of the state machines required. And with the message flows independent there is no way for an invalid state to occur when the two negotiations contain contradictory information. A cost of this is that endpoints will now need to deal with the fact that CLUE messages may contain parameters exceeding those negotiated in SDP, or referencing SDP content that does not exist. However, this is analogous to an issue endpoints already deal with in SDP. For instance, the sum of bandwidth parameters for various m-lines can exceed the overall session bandwidth. Not only is this not invalid,

but it can be desirable, as it allows the sender to prioritise streams. What can be sent for any device is simply the intersection of what is permitted by the most recent SDP offer/answer, and the outcome of the CLUE negotiation; implementations should ignore references to entities in the other negotiation that do no exist.

This does not mean that there will be no interaction between SDP and CLUE messaging - a device wishing to add a new stream may well need to update both their SDP and their CLUE negotiations. However, there is no fixed order in which this must be done and no requirement for them to be updated in a particular order or fashion; it is left to the implementation to renegotiate the channels as it sees fit. If updates to both negotiations are required for a new stream to be added, then the new stream will not be available until both renegotiations are complete - the completion of the first renegotiation will have no effect.

6. Security Considerations

This draft only addresses how best to split information between SDP and CLUE signalling and the interdependencies between these two methods of signalling, it does not define the signalling or information itself. As such this draft should require no additional security considerations.

7. References

7.1. Normative References

[I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-09 (work in progress), February 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

7.2. Informative References

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

Appendix A. Changes From Draft −00

- Reordered main sections, as in the discussion about interdependence of SDP and CLUE is is useful to reference the split between CLUE and SDP.
- Added more detail to the argument of why dependencies should be CLUE−>SDP and not the other way around or in both directions.
- Fixed spelling issues and did some minor rewording.

Author's Address

Robert Hansen
Cisco Systems
San Jose, CA 95134
USA

Email: rohanse2@cisco.com

CLUE Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2014

C. Holmberg
Ericsson

February 12, 2014

CLUE Protocol Data Channel
draft-holmberg-clue-datachannel-03

Abstract

This document specifies how to use the Stream Control Transmission Protocol (SCTP) on top of the Datagram Transport Layer Security (DTLS) protocol (SCTPoDTLS) for transporting CLUE protocol messages between CLUE entities.

The document describes the SCTP considerations for CLUE, and the SDP Offer/Answer procedures for negotiating a SCTPoDTLS connection for CLUE.

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 August 16, 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. Usage of SCTPoDTLS in the CLUE Context 3
3.1. General 3
3.2. CLUE Data Channel Definition 3
3.3. RTCWEB Data Channel Protocol Usage 4
3.4. SCTP Considerations 4
3.4.1. SCTP Payload Protocol Identifier (PPID) 4
3.4.2. Reliability 4
3.4.3. Order 5
3.4.4. Stream Reset 5
3.4.5. Interleaving 5
3.4.6. SCTP Multihoming 5
4. SDP Offer/Answer Procedures 5
4.1. SDP-based WebRTC Data Channel Negotiation Usage 5
5. Security Considerations 5
6. IANA Considerations 5
7. Acknowledgments 6
8. Change Log 6
9. References 6
9.1. Normative References 6
9.2. Informative References 7
Author's Address 8

1. Introduction

This document specifies how to use the Stream Control Transmission Protocol (SCTP) on top of the Datagram Transport Layer Security (DTLS) protocol (SCTPoDTLS) for transporting CLUE protocol messages between CLUE entities.

The document describes the SCTP considerations for CLUE, and the SDP Offer/Answer procedures for negotiating a SCTPoDTLS connection for CLUE.

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].

CLUE data channel refers to the SCTPoDTLS association that is used between two CLUE entities in order to transport CLUE messages.

CLUE message refers to a CLUE protocol message that is sent over the CLUE data channel.

CLUE entity refers to a SIP User Agent (UA) device that supports the CLUE mechanism (including the CLUE protocol).

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

3. Usage of SCTPoDTLS in the CLUE Context

3.1. General

This section defines the CLUE data channel, and describes the SCTP features and extensions used to realize it.

The CLUE data channel realization, and set of SCTP features, are based on the the RTCWEB data channel defined in [I-D.ietf-rtcweb-data-channel]. This will allow CLUE entities to be interoperable with entities implementing [I-D.ietf-rtcweb-data-channel].

3.2. CLUE Data Channel Definition

The CLUE data channel is realized by using the Stream Control Transmission Protocol (SCTP) on top of the Datagram Transport Layer Security (DTLS) protocol [I-D.ietf-tsvwg-sctp-dtls-encaps].

The realization of a bidirectional CLUE Data Channel is a pair of one incoming SCTP stream and one outgoing SCTP stream. These streams are then used to transport CLUE messages in both directions.

The SCTP streams MUST belong to the same SCTP association.

Within a given CLUE session, CLUE entities MUST use a single CLUE data channel for all CLUE messages associated with the CLUE session.

The SCTP streams MUST have identical SCTP stream identifier values, unless a specific value is already used for some other purpose.

OPEN ISSUE #1: The requirement to use identical STCP stream identifier values might be modified depending on what mechanism will be used to negotiate the identifier values.

3.3. RTCWEB Data Channel Protocol Usage

OPEN ISSUE #2: It is FFS whether the RTCWEB Data Channel Protocol [I-D.ietf-rtcweb-data-protocol] will be used with the CLUE data channel.

NOTE: If [I-D.ietf-rtcweb-data-protocol] will be used with the CLUE data channel, a new associated 'protocol' value needs to be registered with IANA in the 'Protocol Registry' defined by [I-D.ietf-rtcweb-data-protocol].

3.4. SCTP Considerations

3.4.1. SCTP Payload Protocol Identifier (PPID)

CLUE entities MUST use the PPID value 51 when sending CLUE messages, according to the procedures in [I-D.ietf-rtcweb-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 protocol the SCTP message contains.

NOTE: If the RTCWEB Data Channel Protocol [I-D.ietf-rtcweb-data-protocol] will be used for the CLUE data channel, the PPID value 50 will be used for Data Channel Protocol messages.

3.4.2. Reliability

The usage of SCTP for the data channel ensures reliable transport of CLUE messages.

NOTE: [I-D.ietf-rtcweb-data-channel] requires the support of the partial reliability extension defined in [RFC3758]. This is not needed for CLUE, 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].

3.4.3. Order

CLUE entities MUST use the ordered delivery SCTP service, as described in section 6.6 of [RFC2960].

3.4.4. Stream Reset

CLUE entities 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.

3.4.5. Interleaving

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

3.4.6. SCTP Multihoming

CLUE entities MUST NOT use SCTP multihoming.

NOTE: The SCTPoDTLS mechanism does not support SCTP multihoming.

4. SDP Offer/Answer Procedures

4.1. SDP-based WebRTC Data Channel Negotiation Usage

OPEN ISSUE #3: 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.

NOTE: If [I-D.ejzak-dispatch-webrtc-data-channel-sdpneg] will be used with the CLUE data channel, a new associated ’sub-protocol’ value needs to be registered with IANA.

5. Security Considerations

TBD

6. IANA Considerations

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

7. Acknowledgments

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

8. Change Log

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

Changes from draft-holmberg-clue-datachannel-02

- PPID value for CLUE messages added
- References updated

Changes from draft-holmberg-clue-datachannel-01

- More text added

Changes from draft-holmberg-clue-datachannel-00

- Editorial corrections based on comments from Paul K

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[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.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-02.txt (work in progress), October 2013.

[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.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.

[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.

9.2. 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.

Author's Address

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

CLUE Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2014

C. Holmberg
Ericsson

March 13, 2014

CLUE Protocol Data Channel
draft-holmberg-clue-datachannel-04

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 14, 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 5
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 6
5. SDP Offer/Answer Procedures 7
5.1. General 7
5.2. SDP Media Description Fields 7
5.3. SDP sctpmap Attribute 7
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 11
Author's Address 11

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 use the Data Channel Establishment Protocol (DCEP) [I-D.ietf-rtcweb-data-channel], in order to open a CLUE Data Channel.

The details of the DCEP usage with a CLUE Data Channel are described in section X.X.X.

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.

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 [RFC2960].

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. 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".

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. Instead, 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 X.X.X 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 Salvatore’s draft.

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
"applicationS	DTLS port value	"UDP/TLS/UDPTL"	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.

In an offer, the offerer MUST NOT insert more than one "m=" line describing an SCTPoDTLS association to be used to realize a CLUE Data Channel.

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-03

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

Changes from draft-holmberg-clue-datachannel-02

- PPID value for CLUE messages added

- References updated

Changes from draft-holmberg-clue-datachannel-01

- More text added

Changes from draft-holmberg-clue-datachannel-00

- 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.

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[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-02.txt (work in progress), October 2013.

[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 Working Group R. Presta
Internet-Draft S P. Romano
Intended status: Informational University of Napoli
Expires: August 7, 2014 February 3, 2014

An XML Schema for the CLUE data model
draft-ietf-clue-data-model-schema-03

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 August 7, 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 6
4. <mediaCaptures> 15
5. <encodings> 16
6. <encodingGroups> 16
7. <captureScenes> 16
8. <simultaneousSets> 16
9. <captureEncodings> 16
10. <mediaCapture> 16
10.1. <capturedMedia> 18
10.2. <captureSceneIDREF> 18
10.3. <encGroupIDREF> 18
10.4. <spatialInformation> 18
10.4.1. <capturePoint> 19
10.4.2. <captureArea> 20
10.5. <nonSpatiallyDefinable> 21
10.6. <contentCaptureIDs> 21
10.7. <synchronizationID> 22
10.8. <composed> 22
10.9. <switching> 22
10.10. <policy> 22
10.11. <maxCaptures> 22
10.12. <single> 22
10.13. <description> 23
10.14. <priority> 23
10.15. <lang> 23
10.16. <mobility> 23
10.17. <maxCaptureEncodings> 24
10.18. <relatedTo> 24
10.19. captureID attribute 24
11. Audio captures 24
11.1. <audioChannelFormat> 25
12. Video captures 25
12.1. <embeddedText> 26
13. Text captures 26
14. <captureScene> 27
14.1. <sceneEntries> 27
14.2. sceneID attribute 28
14.3. scale attribute 28
15. <sceneEntry> 28
15.1. <mediaCaptureIDs> 29
15.2. sceneEntryID attribute 29
15.3. mediaType attribute 30
16. <encoding> 30
16.1. <encodingName> 30
16.2. <maxBandwidth> 30
16.3. encodingID attribute 30

17. Audio encodings 31
18. Video encodings 31
19. <encodingGroup> 32
19.1. <maxGroupBandwidth> 32
19.2. <encodingIDList> 32
19.3. encodingGroupID attribute 33
20. <simultaneousSet> 33
20.1. <captureIDREF> 33
20.2. <sceneEntryIDREF> 33
21. <captureEncoding> 33
21.1. <mediaCaptureID> 34
21.2. <encodingID> 34
22. <clueInfo> 34
23. Sample XML file 35
24. MCC example 42
25. Diff with draft-ietf-clue-data-model-schema-01 version . . . 49
26. Diff with draft-ietf-clue-data-model-schema-02 version . . . 50
27. Informative References 50

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 actually represents a proposal aiming at the definition of a coherent structure for all the information associated with the description of a telepresence scenario.

2. Terminology

This document refers to the same terminology used in [I-D.ietf-clue-framework]. We briefly recall herein some of the main terms exploited in the document.

Audio Capture: Media Capture for audio. Denoted as ACn 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: 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.

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 proposed 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 current streams it is transmitting within a telepresence session.

The main groups of information are:

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

<encodings>: the list of individual encodings (Section 5)

<encodingGroups>: the list of encodings groups (Section 6)

<captureScenes>: the list of capture scenes (Section 7)

<simultaneousSets>: the list of simultaneous transmission sets(Section 8)

<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"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">

<!-- ELEMENT DEFINITIONS -->
<xs:element name="mediaCaptures" type="mediaCapturesType"/>
<xs:element name="encodings" type="encodingsType"/>
<xs:element name="encodingGroups" type="encodingGroupsType"/>
<xs:element name="captureScenes" type="captureScenesType"/>
<xs:element name="simultaneousSets" type="simultaneousSetsType"/>
<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"/>
    <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="contentCaptureIDs" type="captureIDListType"
        minOccurs="0"/>
        <xs:element name="composed" type="xs:boolean" minOccurs="0"/>
        <xs:element name="switching" type="xs:boolean" minOccurs="0"/>
        <xs:element name="policy" type="xs:string" minOccurs="0"/>
        <xs:element name="maxCaptures" type="xs:unsignedInt" minOccurs="0"/>
      </xs:sequence>
      <xs:element name="single" 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="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>

<!-- 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="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:attribute name="pointID" type="xs:ID"/>
  </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:attribute name="mediaType" type="xs:string" 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>

<!-- ENCODINGS TYPE -->
<xs:complexType name="encodingsType">
 <xs:sequence>
  <xs:element name="encoding" type="encodingType"
  maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>

<!-- ENCODING TYPE -->
<xs:complexType name="encodingType" abstract="true">
 <xs:sequence>
  <xs:element name="encodingName" type="xs:string"/>
  <xs:element name="maxBandwidth" type="xs:unsignedInt"/>
  <xs:any namespace="##other" processContents="lax" minOccurs="0"
  maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="encodingID" type="xs:ID" use="required"/>
 <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<!-- AUDIO ENCODING TYPE -->
<xs:complexType name="audioEncodingType">
 <xs:complexContent>
  <xs:extension base="tns:encodingType">
   <xs:sequence>
    <xs:element name="encodedMedia" type="xs:string" fixed="audio"
    minOccurs="0"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>

<!-- VIDEO ENCODING TYPE -->
<xs:complexType name="videoEncodingType">
 <xs:complexContent>
  <xs:extension base="tns:encodingType">
   <xs:sequence>
    <xs:element name="encodedMedia" type="xs:string" fixed="video"
    minOccurs="0"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</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="encIDREF" type="xs:IDREF" 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:sequence>
 <xs:attribute name="setID" type="xs:ID" use="required"/>
</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="mediaCaptureID" type="xs:string"/>
  <xs:element name="encodingID" type="xs:string"/>
 </xs:sequence>
 <xs:attribute name="ID" type="xs:ID"/>
</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="encodings"/>
   <xs:element ref="encodingGroups"/>
   <xs:element ref="captureScenes"/>
   <xs:element ref="simultaneousSets" 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: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. <encodings>

<encodings> represents the list of individual encodings available on the media provider's side. Each individual encoding is represented by an <encoding> element (Section 16).

6. <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 19).

7. <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).

8. <simultaneousSets>

<simultaneousSets> contains the simultaneous sets indicated by the media provider. Each simultaneous set is represented by a <simultaneousSet> element. (Section 20).

9. <captureEncodings>

<captureEncodings> is a list of capture encodings. It can represents 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 21).

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 with a set of features that are independent from the specific type of medium, and with a set of feature that are media-specific. We design the media capture type as an abstract type, providing all the features that can be common to all media types. Media-specific captures, such as video captures, audio captures and others, are specialization of that 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"/>
    <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="contentCaptureIDs" type="captureIDListType"
        minOccurs="0"/>
        <xs:element name="composed" type="xs:boolean" minOccurs="0"/>
        <xs:element name="switching" type="xs:boolean" minOccurs="0"/>
        <xs:element name="policy" type="xs:string" minOccurs="0"/>
        <xs:element name="maxCaptures" type="xs:unsignedInt" minOccurs="0"/>
      </xs:sequence>
      <xs:element name="single" 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="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 belongs to. Indeed, each media capture must be associated with 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: non spatially definable captures and spatially definable captures.

Non spatially definable captures are those that do not capture parts of the telepresence room. Capture of this case 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.

Spatially definable captures are those that capture part of the telepresence room. The captured part of the telepresence room is described by means of the <spatialInformation> element.

This is the definition of the spatial information type:

```
<!-- 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 represented 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:

```
<!-- 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:attribute name="pointID" type="xs:ID"/>
  </xs:extension>
 </xs:complexContent>
</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>
```

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.

If the point of capture is not specified, it means the consumer should not assume anything about the spatial location of the capturing 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 axis 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 co-planar.

By comparing the capture area of different media captures within the same capture scene, a consumer can determine the spatial relationships between them and render them correctly. If the area of capture is not specified, it means the Media Capture is not spatially related to any other media capture.

10.5. <nonSpatiallyDefinable>

When media captures are non spatially definable, they are marked with the boolean <nonSpatiallyDefinable> element set to "true".

10.6. <contentCaptureIDs>

A media capture can be alternatively a single media capture or a multiple content capture. A multiple content capture is made by different captures that can be arranged spatially (by a mixing 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 <contentCaptureIDs>. It is the identifiers list of the media captures that can be part of the content of the multiple content capture.

[containedCaptureIDs or contentIDs or contentCaptureIDs?]

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 room endpoint.

10.8. <composed>

<composed> is an optional boolean element that can be used only for multiple content captures. It indicates wheter or not a multiple content capture is a mix (audio) or a composition (video) of streams. This attribute is useful for a media consumer for example to avoid nesting a composed video capture into another composed capture or rendering.

[edt’s note: proposal - discussion needed]

10.9. <switching>

<switching> is an optional boolean element that can be used only for multiple content captures. It indicates wheter or not a multiple content capture switches over the time.

[edt’s note: proposal - discussion needed]

10.10. <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]).

10.11. <maxCaptures>

<maxCaptures> is an optional element that can be used only for multiple content captures. It indicates the maximum number of media captures that can be represented in the multiple content capture at a time.

10.12. <single>

<single> is a mandatory 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.

[edt's note: proposal - discussion needed]

10.13. <description>

<description> is used to provide optionally human-readable textual information. It is used to describe media captures, capture scenes and capture scene entries. A media capture can be described by using multiple <description> elements, each one providing information in a different language. Indeed, 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.

10.14. <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 "important" contribution available 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".

10.15. <lang>

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

10.16. <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: static, dynamic or highly dynamic, defined as in [I-D.ietf-clue-framework].

10.17. <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 my means of the <encGroupIDREF> element.

10.18. <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.

10.19. 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="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 10.13. Besides <description>, there the <sceneEntries> element (Section 14.1), which is the list of the capture scene entries.

14.1. <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>
```

14.2. sceneID attribute

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

14.3. 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:attribute name="mediaType" type="xs:string" 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 10.13.

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. <encoding>

The <encoding> element represents an individual encoding, i.e., a way to encode a media capture. Individual encodings can be characterized with features that are independent from the specific type of medium, and with features that are media-specific. We design the individual encoding type as an abstract type, providing all the features that can be common to all media types. Media-specific individual encodings, such as video encodings, audio encodings and others, are specialization of that type, as in a typical generalization-specialization hierarchy.

```
<!-- ENCODING TYPE -->
<xs:complexType name="encodingType" abstract="true">
 <xs:sequence>
  <xs:element name="encodingName" type="xs:string"/>
  <xs:element name="maxBandwidth" type="xs:integer"/>
  <xs:any namespace="##other" processContents="lax" minOccurs="0"
  maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="encodingID" type="xs:ID" use="required"/>
 <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

16.1. <encodingName>

<encodingName> is a mandatory field containing the name of the encoding (e.g., G711, H264, ...).

16.2. <maxBandwidth>

<maxBandwidth> represent the maximum bitrate the media provider can instantiate for that encoding.

16.3. encodingID attribute

The encodingID attribute is a mandatory attribute containing the identifier of the individual encoding.

17. Audio encodings

Audio encodings inherit all the features of a generic individual encoding and can present further audio-specific encoding characteristics. The XML Schema definition of the audio encoding type is reported below:

```
<!-- AUDIO ENCODING TYPE -->
<xs:complexType name="audioEncodingType">
 <xs:complexContent>
  <xs:extension base="tns:encodingType">
   <xs:sequence>
    <xs:element name="encodedMedia" type="xs:string" fixed="audio"
    minOccurs="0"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Up to now the only audio-specific information is the <encodedMedia> element containing the media type of the media captures that can be encoded with the considered individual encoding. In the case of audio encoding, that element is forced to "audio".

18. Video encodings

Similarly to audio encodings, video encodings can extend the information of a generic individual encoding with video-specific encoding features.

The <encodedMedia> element contains the media type of the media captures that can be encoded with the considered individual encoding. In the case of video encoding, that element is forced to "video".

```
<!-- VIDEO ENCODING TYPE -->
<xs:complexType name="videoEncodingType">
 <xs:complexContent>
  <xs:extension base="tns:encodingType">
   <xs:sequence>
    <xs:element name="encodedMedia" type="xs:string" fixed="video"
    minOccurs="0"/>
   </xs:sequence>
  </xs:extension>
```

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

19. <encodingGroup>

The <encodingGroup> element represents an encoding group, which is a set of one or more individual encodings, and parameters that apply to the group as a whole. 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.

19.1. <maxGroupBandwidth>

<maxGroupBandwidth> is an optional field containing the maximum bitrate supported for all the individual encodings included in the encoding group.

19.2. <encodingIDList>

<maxGroupBandwidth> is the list of the individual encoding grouped together. Each individual encoding is represented through its identifier contained within an <encIDREF> element.

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

19.3. encodingGroupID attribute

The encodingGroupID attribute contains the identifier of the encoding group.

20. <simultaneousSet>

<simultaneousSet> represents a simultaneous set, i.e. a list of capture of the same type that cab 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>
```

[edt note: need to be checked]

20.1. <captureIDREF>

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

20.2. <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.

21. <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]. A media consumer expresses for each capture encoding its preferences about the capture parameters (such as the desired switching policy) and the encoding parameters (such as the bandwidth). A possible solution to model such entity is provided in the following.

```
   <!-- CAPTURE ENCODING TYPE -->
   <xs:complexType name="captureEncodingType">
    <xs:sequence>
     <xs:element name="mediaCaptureID" type="xs:string"/>
     <xs:element name="encodingID" type="xs:string"/>
     <xs:element name="captureParameters" type="captureParametersType"
                 minOccurs="0"/>
     <xs:element name="encodingParameters" type="encodingParametersType"
                 minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:ID"/>
   </xs:complexType>

   <!-- CAPTURE PARAMETERS TYPE -->
   <xs:complexType name="captureParametersType">
    <xs:sequence>
           <xs:element name="switchingOption" type="xs:string"/>
           <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>

   <!-- ENCODING PARAMETERS TYPE -->
   <xs:complexType name="captureParametersType">
    <xs:sequence>
           <xs:element name="bandwidth" type="xs:unsignedInt"/>
           <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
```

21.1. <mediaCaptureID>

<mediaCaptureID> contains the identifier of the media capture that has been encoded to form the capture encoding.

21.2. <encodingID>

<encodingID> contains the identifier of the applied individual encoding.

22. <clueInfo>

The <clueInfo> element has been left within the XML Schema for the sake of convenience when representing a prototype of 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="encodings"/>
   <xs:element ref="encodingGroups"/>
   <xs:element ref="captureScenes"/>
   <xs:element ref="simultaneousSets"/>
   <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>
```

23. Sample XML file

The following XML document represents a schema compliant example of a CLUE telepresence scenario.

In the considered scenario, there are 5 video captures:

VC0: the video from the left camera

VC1: the video from the central camera

VC2: the video from the right camera

VC3: the overall view of the telepresence room taken from the central camera

VC4: the video associated with the slide stream

There are 2 audio captures:

AC0: the overall room audio taken from the central camera

AC1: the audio associated with the slide stream presentation

The captures are organized into two capture scenes:

CS1: this scene contains captures associated with the participants that are in the telepresence room.

CS2: this scene contains captures associated with the slide presentation, which is a pre-registered presentation played within the context of the telepresence session.

Within the capture scene CS1, there are three scene entries available:

CS1_SE1: this entry contains the partipants' video captures taken from the three cameras (VC0, VC1, VC2).

CS1_SE2: this entry contains the zoomed-out view of the overall telepresence room (VC3)

CS1_SE3: this entry contains the overall telepresence room audio (AC0)

On the other hand, capture scene CS2 presents two scene entries:

CS2_SE1: this entry contains the presentation audio stream (AC1)

CS2_SE2: this entry contains the presentation video stream (VC4)

There are two encoding groups:

EG0 This encoding groups involves video encodings ENC0, ENC1, ENC2. All of them are identical video encodings that can consume up to 128K bps. The total amount of available bandwidth for EG0 is 384Kbps. That means that only three video capture encodings can be issued at a time.

EG1 This encoding groups involves audio encodings ENC3, ENC4. Both of them have a maximum bandwidth of 64Kbps. The EG1 maximum bandwidth is 128Kbps. That means that only two audio captures can be issued at a time.

As to the simultaneous sets, only VC1 and VC3 cannot be transmitted simultaneously since they are captured by the same device. i.e. the central camera (VC3 is a zoomed-out view while VC1 is a focused view of the front participants). The simultaneous sets would then be the following:

SS1 made by VC0, VC1, VC2, VC4

SS2 made by VC0, VC3, VC2, VC4

SS3 made by AC0 and AC1

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<clueInfo xmlns="urn:ietf:params:xml:ns:clue-info" clueInfoID="NapoliRoom">
    <mediaCaptures>
        <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>1.0</x>
                    <y>1.0</y>
                    <z>1.0</z>
                    <lineOfCapturePoint>
                        <x>1.0</x>
                        <y>0.0</y>
                        <z>1.0</z>
                    </lineOfCapturePoint>
                </capturePoint>
            </spatialInformation>
            <single>true</single>
            <description lang="en">left-most participants video</description>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </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>0.0</x>
                    <y>1.0</y>
                    <z>1.0</z>
                    <lineOfCapturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>1.0</z>
```

```
                    </lineOfCapturePoint>
                </capturePoint>
            </spatialInformation>
            <single>true</single>
            <description lang="en">central participants video</description>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </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>-1.0</x>
                    <y>1.0</y>
                    <z>1.0</z>
                    <lineOfCapturePoint>
                        <x>-1.0</x>
                        <y>0.0</y>
                        <z>1.0</z>
                    </lineOfCapturePoint>
                </capturePoint>
            </spatialInformation>
            <single>true</single>
            <description lang="en">right-most participants video</description>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </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>
            <spatialInformation>
                <capturePoint>
                    <x>0.0</x>
                    <y>1.0</y>
                    <z>1.0</z>
                    <lineOfCapturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>1.0</z>
                    </lineOfCapturePoint>
                </capturePoint>
```

```
            </spatialInformation>
            <single>true</single>
            <description lang="en">overall room video</description>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>room</view>
        </mediaCapture>
        <mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="vc4">
            <capturedMedia>video</capturedMedia>
            <captureSceneIDREF>cs2</captureSceneIDREF>
            <encGroupIDREF>EG0</encGroupIDREF>
            <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
            <single>true</single>
            <description lang="en">slides video</description>
            <lang>Italian</lang>
            <presentation>slides</presentation>
        </mediaCapture>
        <mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="audioCaptureType" captureID="ac0">
            <capturedMedia>audio</capturedMedia>
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <encGroupIDREF>EG1</encGroupIDREF>
            <spatialInformation>
                <capturePoint>
                    <x>0.0</x>
                    <y>1.0</y>
                    <z>1.0</z>
                    <lineOfCapturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>1.0</z>
                    </lineOfCapturePoint>
                </capturePoint>
            </spatialInformation>
            <single>true</single>
            <description lang="en">room audio</description>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>room</view>
            <audioChannelFormat>stereo</audioChannelFormat>
        </mediaCapture>
        <mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="audioCaptureType" captureID="ac1">
            <capturedMedia>audio</capturedMedia>
            <captureSceneIDREF>cs2</captureSceneIDREF>
            <encGroupIDREF>EG1</encGroupIDREF>
            <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
```

```
            <single>true</single>
            <description lang="en">slide presentation audio</description>
            <lang>Italian</lang>
            <presentation>slides</presentation>
            <audioChannelFormat>mono</audioChannelFormat>
        </mediaCapture>
    </mediaCaptures>
    <encodings>
        <encoding xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoEncodingType" encodingID="ENC0">
            <encodingName>vp8</encodingName>
            <maxBandwidth>128000</maxBandwidth>
            <encodedMedia>video</encodedMedia>
        </encoding>
        <encoding xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoEncodingType" encodingID="ENC1">
            <encodingName>vp8</encodingName>
            <maxBandwidth>128000</maxBandwidth>
            <encodedMedia>video</encodedMedia>
        </encoding>
        <encoding xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoEncodingType" encodingID="ENC2">
            <encodingName>vp8</encodingName>
            <maxBandwidth>128000</maxBandwidth>
            <encodedMedia>video</encodedMedia>
        </encoding>
        <encoding xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="audioEncodingType" encodingID="ENC3">
            <encodingName>g711</encodingName>
            <maxBandwidth>64000</maxBandwidth>
            <encodedMedia>audio</encodedMedia>
        </encoding>
        <encoding xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="audioEncodingType" encodingID="ENC4">
            <encodingName>g711</encodingName>
            <maxBandwidth>32000</maxBandwidth>
            <encodedMedia>audio</encodedMedia>
        </encoding>
    </encodings>
    <encodingGroups>
        <encodingGroup encodingGroupID="EG0">
            <maxGroupBandwidth>0</maxGroupBandwidth>
            <encodingIDList>
                <encIDREF>ENC0</encIDREF>
                <encIDREF>ENC1</encIDREF>
                <encIDREF>ENC2</encIDREF>
            </encodingIDList>
        </encodingGroup>
```

```
        <encodingGroup encodingGroupID="EG1">
            <maxGroupBandwidth>0</maxGroupBandwidth>
            <encodingIDList>
                <encIDREF>ENC3</encIDREF>
                <encIDREF>ENC4</encIDREF>
            </encodingIDList>
        </encodingGroup>
    </encodingGroups>
    <captureScenes>
        <captureScene scale="unknown" sceneID="CS1">
            <description lang="en">main scene</description>
            <sceneEntries>
                <sceneEntry mediaType="audio" sceneEntryID="CS1_SE3">
                    <description lang="en">overall room audio</description>
                    <mediaCaptureIDs>
                        <captureIDREF>ac0</captureIDREF>
                    </mediaCaptureIDs>
                </sceneEntry>
                <sceneEntry mediaType="video" sceneEntryID="CS1_SE2">
                    <description lang="en">overall room video</description>
                    <mediaCaptureIDs>
                        <captureIDREF>vc3</captureIDREF>
                    </mediaCaptureIDs>
                </sceneEntry>
                <sceneEntry mediaType="video" sceneEntryID="CS1_SE1">
                    <description lang="en">participants video</description>
                    <mediaCaptureIDs>
                        <captureIDREF>vc0</captureIDREF>
                        <captureIDREF>vc1</captureIDREF>
                        <captureIDREF>vc2</captureIDREF>
                    </mediaCaptureIDs>
                </sceneEntry>
            </sceneEntries>
        </captureScene>
        <captureScene scale="noscale" sceneID="cs2">
            <description lang="en">presentation scene</description>
            <sceneEntries>
                <sceneEntry mediaType="audio" sceneEntryID="CS2_SE2">
                    <mediaCaptureIDs>
                        <captureIDREF>ac1</captureIDREF>
                    </mediaCaptureIDs>
                </sceneEntry>
                <sceneEntry mediaType="video" sceneEntryID="CS2_SE1">
                    <mediaCaptureIDs>
                        <captureIDREF>vc4</captureIDREF>
                    </mediaCaptureIDs>
                </sceneEntry>
            </sceneEntries>
```

```
        </captureScene>
    </captureScenes>
    <simultaneousSets>
        <simultaneousSet setID="ss1">
            <captureIDREF>vc0</captureIDREF>
            <captureIDREF>vc3</captureIDREF>
            <captureIDREF>vc2</captureIDREF>
            <captureIDREF>vc4</captureIDREF>
        </simultaneousSet>
        <simultaneousSet setID="ss2">
            <captureIDREF>vc0</captureIDREF>
            <captureIDREF>vc1</captureIDREF>
            <captureIDREF>vc2</captureIDREF>
            <captureIDREF>vc4</captureIDREF>
        </simultaneousSet>
        <simultaneousSet setID="ss3">
            <captureIDREF>ac0</captureIDREF>
            <captureIDREF>ac1</captureIDREF>
        </simultaneousSet>
    </simultaneousSets>
</clueInfo>
```

24. MCC example

In this example, the endpoint is equipped with three cameras capturing three individual captures (VC1, VC2, VC3). The central one is able to capture a zoomed-out view of the room (VC4). Moreover, the MP is able to advertise a composed MCC (MCC5) made by a big picture representing the current speaker (MCC0) and two picture-in-picture boxes representing the previous speakers (the previous one -MCC1- and the oldest one - MCC2-).

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" clueInfoID="NapoliRoom">
    <mediaCaptures>
        <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>
                <captureArea>
                    <bottomLeft>
                        <x>0.0</x>
                        <y>3.0</y>
                        <z>0.0</z>
                    </bottomLeft>
                    <bottomRight>
                        <x>1.0</x>
                        <y>3.0</y>
                        <z>0.0</z>
                    </bottomRight>
                    <topLeft>
                        <x>0.0</x>
                        <y>3.0</y>
                        <z>3.0</z>
                    </topLeft>
                    <topRight>
                        <x>1.0</x>
                        <y>3.0</y>
                        <z>3.0</z>
                    </topRight>
                </captureArea>
            </spatialInformation>
            <single>true</single>
            <description lang="en">left-most participants video</description>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </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>0.0</x>
                    <y>1.0</y>
                    <z>1.0</z>
```

```
                    <lineOfCapturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>1.0</z>
                    </lineOfCapturePoint>
                </capturePoint>
            </spatialInformation>
            <single>true</single>
            <description lang="en">central participants video</description>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </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>
            <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>
                <captureArea>
                    <bottomLeft>
                        <x>0.0</x>
                        <y>3.0</y>
                        <z>0.0</z>
                    </bottomLeft>
                    <bottomRight>
                        <x>1.0</x>
                        <y>3.0</y>
                        <z>0.0</z>
                    </bottomRight>
                    <topLeft>
                        <x>0.0</x>
                        <y>3.0</y>
                        <z>3.0</z>
                    </topLeft>
                    <topRight>
                        <x>1.0</x>
                        <y>3.0</y>
```

```
                    <z>3.0</z>
                </topRight>
            </captureArea>
        </spatialInformation>
        <single>true</single>
        <description lang="en">right-most participants video</description>
        <lang>Italian</lang>
        <mobility>static</mobility>
        <view>individual</view>
    </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>
            <captureArea>
                <bottomLeft>
                    <x>0.0</x>
                    <y>3.0</y>
                    <z>0.0</z>
                </bottomLeft>
                <bottomRight>
                    <x>1.0</x>
                    <y>3.0</y>
                    <z>0.0</z>
                </bottomRight>
                <topLeft>
                    <x>0.0</x>
                    <y>3.0</y>
                    <z>3.0</z>
                </topLeft>
                <topRight>
                    <x>1.0</x>
                    <y>3.0</y>
                    <z>3.0</z>
                </topRight>
            </captureArea>
```

```
            </spatialInformation>
            <single>true</single>
            <description lang="en">table view video</description>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>table</view>
        </mediaCapture>
        <mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="mcc0">
            <capturedMedia>video</capturedMedia>
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <encGroupIDREF>EG0</encGroupIDREF>
            <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
            <contentCaptureIDs>
                <captureIDREF>VC1</captureIDREF>
                <captureIDREF>vc2</captureIDREF>
                <captureIDREF>vc3</captureIDREF>
            </contentCaptureIDs>
            <composed>false</composed>
            <switching>true</switching>
            <policy>SoundLevel:0</policy>
            <maxCaptures>1</maxCaptures>
            <description lang="en">video of the current
            loudest speaker</description>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </mediaCapture>
        <mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="mcc1">
            <capturedMedia>video</capturedMedia>
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <encGroupIDREF>EG0</encGroupIDREF>
            <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
            <contentCaptureIDs>
                <captureIDREF>VC1</captureIDREF>
                <captureIDREF>vc2</captureIDREF>
                <captureIDREF>vc3</captureIDREF>
            </contentCaptureIDs>
            <composed>false</composed>
            <switching>true</switching>
            <policy>SoundLevel:1</policy>
            <maxCaptures>1</maxCaptures>
            <description lang="en">video of the previous loudest speaker</descri
ption>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </mediaCapture>
```

```
        <mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="mcc2">
            <capturedMedia>video</capturedMedia>
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <encGroupIDREF>EG0</encGroupIDREF>
            <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
            <contentCaptureIDs>
                <captureIDREF>VC1</captureIDREF>
                <captureIDREF>vc2</captureIDREF>
                <captureIDREF>vc3</captureIDREF>
            </contentCaptureIDs>
            <composed>false</composed>
            <switching>true</switching>
            <policy>SoundLevel:2</policy>
            <maxCaptures>1</maxCaptures>
            <description lang="en">video of the oldest loudest speaker</descript
ion>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </mediaCapture>
        <mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="mcc5">
            <capturedMedia>video</capturedMedia>
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <encGroupIDREF>EG0</encGroupIDREF>
            <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
            <contentCaptureIDs>
                <captureIDREF>mcc0</captureIDREF>
                <captureIDREF>mcc1</captureIDREF>
                <captureIDREF>mcc2</captureIDREF>
            </contentCaptureIDs>
            <composed>true</composed>
            <switching>true</switching>
            <policy>SoundLevel</policy>
            <maxCaptures>1</maxCaptures>
            <description lang="en">big video of the current loudest speaker
            + PiPs of previous speakers</description>
            <lang>Italian</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </mediaCapture>
    </mediaCaptures>
    <encodings>
        <encoding xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoEncodingType" encodingID="ENC0">
            <encodingName>vp8</encodingName>
            <maxBandwidth>128000</maxBandwidth>
            <encodedMedia>video</encodedMedia>
```

```
        </encoding>
        <encoding xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoEncodingType" encodingID="ENC1">
            <encodingName>vp8</encodingName>
            <maxBandwidth>128000</maxBandwidth>
            <encodedMedia>video</encodedMedia>
        </encoding>
        <encoding xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoEncodingType" encodingID="ENC2">
            <encodingName>vp8</encodingName>
            <maxBandwidth>128000</maxBandwidth>
            <encodedMedia>video</encodedMedia>
        </encoding>
    </encodings>
    <encodingGroups>
        <encodingGroup encodingGroupID="EG0">
            <maxGroupBandwidth>384000</maxGroupBandwidth>
            <encodingIDList>
                <encIDREF>ENC0</encIDREF>
                <encIDREF>ENC1</encIDREF>
                <encIDREF>ENC2</encIDREF>
            </encodingIDList>
        </encodingGroup>
    </encodingGroups>
    <captureScenes>
        <captureScene scale="unknown" sceneID="CS1">
            <description lang="en">main scene</description>
            <sceneEntries>
                <sceneEntry mediaType="video" sceneEntryID="CS1_SE1">
                    <description lang="en">participants video</description>
                    <mediaCaptureIDs>
                        <captureIDREF>VC1</captureIDREF>
                        <captureIDREF>vc2</captureIDREF>
                        <captureIDREF>vc3</captureIDREF>
                    </mediaCaptureIDs>
                </sceneEntry>
                <sceneEntry mediaType="video" sceneEntryID="CS1_SE2">
                    <description lang="en">overall room video</description>
                    <mediaCaptureIDs>
                        <captureIDREF>vc4</captureIDREF>
                    </mediaCaptureIDs>
                </sceneEntry>
                <sceneEntry mediaType="video" sceneEntryID="CS1_SE3">
                    <description lang="en">multi-content captures</description>
                    <mediaCaptureIDs>
                        <captureIDREF>mcc0</captureIDREF>
                        <captureIDREF>mcc1</captureIDREF>
                        <captureIDREF>mcc2</captureIDREF>
```

```
                    </mediaCaptureIDs>
                </sceneEntry>
                <sceneEntry mediaType="video" sceneEntryID="CS1_SE4">
                    <description lang="en">composed capture</description>
                    <mediaCaptureIDs>
                        <captureIDREF>mcc5</captureIDREF>
                    </mediaCaptureIDs>
                </sceneEntry>
            </sceneEntries>
        </captureScene>
    </captureScenes>
    <simultaneousSets>
        <simultaneousSet setID="ss1">
            <captureIDREF>VC1</captureIDREF>
            <captureIDREF>vc2</captureIDREF>
            <captureIDREF>vc3</captureIDREF>
            <sceneEntryIDREF>CS1_SE3</sceneEntryIDREF>
            <sceneEntryIDREF>CS1_SE4</sceneEntryIDREF>
        </simultaneousSet>
        <simultaneousSet setID="ss2">
            <captureIDREF>VC1</captureIDREF>
            <captureIDREF>vc4</captureIDREF>
            <captureIDREF>vc3</captureIDREF>
            <sceneEntryIDREF>CS1_SE3</sceneEntryIDREF>
            <sceneEntryIDREF>CS1_SE4</sceneEntryIDREF>
        </simultaneousSet>
    </simultaneousSets>
</clueInfo>
```

25. Diff with draft-ietf-clue-data-model-schema-01 version

Terminology has been added.

Switching policies has been removed from capture scene entry attributes list.

multiple content captures can now be modeled by using the schema. Related attributes (composed, switching, maxCaptures, captureIDs) have been added in the definition of the media capture type.

H26X encoding type has been removed.

maxGroupPps (max pixels per second for encoding groups) removed

Presentation attribute for media captures has been added.

View attribute for media captures has been added.

26. 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.

27. Informative References

[I-D.ietf-clue-framework] Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-13 (work in progress), December 2013.

[RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.

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 R. Presta
Internet-Draft S P. Romano
Intended status: Standards Track University of Napoli
Expires: February 14, 2017 August 13, 2016

An XML Schema for the CLUE data model
draft-ietf-clue-data-model-schema-17

Abstract

This document provides an XML schema file for the definition of CLUE data model types. The term "CLUE" stands for "ControLling mUltiple streams for tElepresence" and is the name of the IETF working group in which this document, as well as other companion documents, has been developed. The document defines a coherent structure for information associated with the description of a telepresence scenario.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 14, 2017.

Copyright Notice

Copyright (c) 2016 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 5
2. Terminology 5
3. Definitions 5
4. XML Schema 7
5. <mediaCaptures> 19
6. <encodingGroups> 19
7. <captureScenes> 19
8. <simultaneousSets> 19
9. <globalViews> 19
10. <captureEncodings> 19
11. <mediaCapture> 19
11.1. captureID attribute 22
11.2. mediaType attribute 22
11.3. <captureSceneIDREF> 22
11.4. <encGroupIDREF> 22
11.5. <spatialInformation> 23
11.5.1. <captureOrigin> 24
11.5.2. <captureArea> 25
11.6. <nonSpatiallyDefinable> 26
11.7. <content> 26
11.8. <synchronizationID> 26
11.9. <allowSubsetChoice> 27
11.10. <policy> 27
11.11. <maxCaptures> 28
11.12. <individual> 29
11.13. <description> 29
11.14. <priority> 29
11.15. <lang> 30
11.16. <mobility> 30
11.17. <relatedTo> 30
11.18. <view> 30
11.19. <presentation> 31
11.20. <embeddedText> 31
11.21. <capturedPeople> 32
11.21.1. <personIDREF> 32
12. Audio captures 32
12.1. <sensitivityPattern> 33
13. Video captures 33
14. Text captures 34
15. Other capture types 34
16. <captureScene> 35
16.1. <sceneInformation> 36
16.2. <sceneViews> 36

16.3. sceneID attribute 36
16.4. scale attribute 36
17. <sceneView> 37
17.1. <mediaCaptureIDs> 38
17.2. sceneViewID attribute 38
18. <encodingGroup> 38
18.1. <maxGroupBandwidth> 39
18.2. <encodingIDList> 39
18.3. encodingGroupID attribute 39
19. <simultaneousSet> 39
19.1. setID attribute 40
19.2. mediaType attribute 40
19.3. <mediaCaptureIDREF> 41
19.4. <sceneViewIDREF> 41
19.5. <captureSceneIDREF> 41
20. <globalView> 41
21. <people> 41
21.1. <person> 42
21.1.1. personID attribute 42
21.1.2. <personInfo> 42
21.1.3. <personType> 43
22. <captureEncoding> 43
22.1. <captureID> 44
22.2. <encodingID> 44
22.3. <configuredContent> 44
23. <clueInfo> 44
24. XML Schema extensibility 45
24.1. Example of extension 46
25. Security considerations 48
26. IANA considerations 49
26.1. XML namespace registration 49
26.2. XML Schema registration 50
26.3. MIME Media Type Registration for "application/clue_info+xml" 50
26.4. Registry for acceptable <view> values 51
26.5. Registry for acceptable <presentation> values 51
26.6. Registry for acceptable <sensitivityPattern> values . . 51
26.7. Registry for acceptable <personType> values 52
27. Sample XML file 52
28. MCC example 60
29. Diff with draft-ietf-clue-data-model-schema-16 version 71
30. Diff with draft-ietf-clue-data-model-schema-15 version 71
31. Diff with draft-ietf-clue-data-model-schema-14 version 71
32. Diff with draft-ietf-clue-data-model-schema-13 version 71
33. Diff with draft-ietf-clue-data-model-schema-12 version 71
34. Diff with draft-ietf-clue-data-model-schema-11 version 71
35. Diff with draft-ietf-clue-data-model-schema-10 version 71
36. Diff with draft-ietf-clue-data-model-schema-09 version 72

37. Diff with draft-ietf-clue-data-model-schema-08 version 72
38. Diff with draft-ietf-clue-data-model-schema-07 version 72
39. Diff with draft-ietf-clue-data-model-schema-06 version 72
40. Diff with draft-ietf-clue-data-model-schema-04 version 73
41. Diff with draft-ietf-clue-data-model-schema-03 version 74
42. Diff with draft-ietf-clue-data-model-schema-02 version 74
43. Acknowledgments 74
44. References 74
44.1. Normative References 74
44.2. Informative References 76

1. Introduction

This document provides an XML schema file for the definition of CLUE data model types. For the benefit of the reader, the term 'CLUE' stands for "ControLling mUltiple streams for tElepresence" and is the name of the IETF working group in which this document, as well as other companion documents, has been developed. A thorough definition of the CLUE framework can be found in [I-D.ietf-clue-framework].

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 document aims at the definition of a coherent structure for 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

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].

3. Definitions

This document refers to the same definitions used in [I-D.ietf-clue-framework], except for the "CLUE Participant" definition. We briefly recall herein some of the main terms used in the document.

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

Capture: Same as Media Capture.

Capture Device: A device that converts physical input, such as audio, video or text, 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 correspond to a real region in physical space, such as a room. A Capture Scene includes attributes and one or more Capture Scene Views, with each view including one or more Media Captures.

Capture Scene View: A list of Media Captures of the same media type that together form one way to represent the entire Capture Scene.

CLUE Participant: This term is imported from the CLUE protocol ([I-D.ietf-clue-protocol]) document.

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 A CLUE-capable device which is 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.

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: A CLUE-capable device that intends to receive Capture Encodings.

Media Provider: A CLUE-capable device that intends to send Capture Encodings.

Multiple Content Capture: A Capture that mixes and/or switches other Captures of a single type (e.g., all audio or all video.) Particular 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.

Multipoint Control Unit (MCU): A CLUE-capable device that connects two or more endpoints together into one single multimedia conference [RFC7667]. An MCU includes an [RFC4353] like Mixer, without the [RFC4353] requirement to send media to each participant.

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

Provider: Same as Media Provider.

Render: The process of reproducing the received Streams like, for instance, displaying of the remote video on the Media Consumer's screens, or playing of the remote audio through loudspeakers.

Scene: Same as Capture Scene.

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.

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

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.

4. XML Schema

This section contains the CLUE data model schema definition.

The element and attribute definitions are formal representations 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 5)

<encodingGroups>: the list of encoding groups (Section 6)

<captureScenes>: the list of capture scenes (Section 7)

<simultaneousSets>: the list of simultaneous transmission sets (Section 8)

<globalViews>: the list of global views sets (Section 9)

<people>: meta data about the participants represented in the telepresence session (Section 21)

<captureEncodings>: the list of instantiated capture encodings (Section 10)

All of the above refers to concepts that have been introduced in [I-D.ietf-clue-framework] and further detailed in 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"
   version="1.0">

<!-- Import xcard XML schema -->
<xs:import namespace="urn:ietf:params:xml:ns:vcard-4.0"
schemaLocation=
  "http://www.iana.org/assignments/xml-registry/schema/vcard-4.0.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="globalViews" type="globalViewsType"/>
```

```
<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="captureSceneIDREF" type="xs:IDREF"/>
    <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="policyType" minOccurs="0"/>
        <xs:element name="maxCaptures" type="maxCapturesType"
         minOccurs="0"/>
```

```
        <xs:element name="allowSubsetChoice" type="xs:boolean"
         minOccurs="0"/>
      </xs:sequence>
      <xs:element name="individual" type="xs:boolean" fixed="true"/>
    </xs:choice>
    <!-- optional fields -->
    <xs:element name="encGroupIDREF" type="xs:IDREF" minOccurs="0"/>
    <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"
     maxOccurs="unbounded"/>
        <xs:element name="mobility" type="mobilityType" minOccurs="0" />
    <xs:element ref="presentation" minOccurs="0" />
    <xs:element ref="embeddedText" minOccurs="0" />
    <xs:element ref="view" minOccurs="0" />
    <xs:element name="capturedPeople" type="capturedPeopleType"
     minOccurs="0"/>
    <xs:element name="relatedTo" type="xs:IDREF" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="captureID" type="xs:ID" use="required"/>
  <xs:attribute name="mediaType" type="xs:string" use="required"/>

</xs:complexType>

<!-- POLICY TYPE -->
<xs:simpleType name="policyType">
 <xs:restriction base="xs:string">
      <xs:pattern value="([a-zA-Z0-9])+[:]([0-9])+"/>
    </xs:restriction>
</xs:simpleType>

<!-- CONTENT TYPE -->
<xs:complexType name="contentType">
 <xs:sequence>
   <xs:element name="mediaCaptureIDREF" type="xs:IDREF"
   minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="sceneViewIDREF" 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:simpleType name="positiveShort">
        <xs:restriction base="xs:unsignedShort">
                <xs:minInclusive value="1">
```

```
                </xs:minInclusive>
        </xs:restriction>
</xs:simpleType>

<xs:complexType name="maxCapturesType">
        <xs:simpleContent>
                <xs:extension base="positiveShort">
                        <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 ref="personType" 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 ELEMENT -->
<xs:element name="personType" type="xs:string">
 <xs:annotation>
  <xs:documentation>
   Acceptable values (enumerations) for this type are managed
   by IANA in the "CLUE Schema &lt;personType&gt; registry",
   accessible at TBD-IANA.
  </xs:documentation>
```

```
 </xs:annotation>
</xs:element>

<!-- VIEW ELEMENT -->
<xs:element name="view" type="xs:string">
 <xs:annotation>
  <xs:documentation>
   Acceptable values (enumerations) for this type are managed
   by IANA in the "CLUE Schema &lt;view&gt; registry",
   accessible at TBD-IANA.
  </xs:documentation>
 </xs:annotation>
</xs:element>

<!-- PRESENTATION ELEMENT -->
<xs:element name="presentation" type="xs:string">
 <xs:annotation>
  <xs:documentation>
   Acceptable values (enumerations) for this type are managed
   by IANA in the "CLUE Schema &lt;presentation&gt; registry",
   accessible at TBD-IANA.
  </xs:documentation>
 </xs:annotation>
</xs:element>

<!-- SPATIAL INFORMATION TYPE -->
<xs:complexType name="spatialInformationType">
 <xs:sequence>
  <xs:element name="captureOrigin" type="captureOriginType"
   minOccurs="0"/>
  <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>


<!-- 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 ORIGIN TYPE -->
```

```
<xs:complexType name="captureOriginType">
  <xs:sequence>
        <xs:element name="capturePoint" type="pointType"></xs:element>
        <xs:element name="lineOfCapturePoint" type="pointType"
         minOccurs="0">
        </xs:element>
  </xs:sequence>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>


<!-- 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>


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


<!-- OTHER CAPTURE TYPE -->
<xs:complexType name="otherCaptureType">
 <xs:complexContent>
```

```
  <xs:extension base="tns:mediaCaptureType">
  <xs:sequence>
  <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>

<!-- AUDIO CAPTURE TYPE -->
<xs:complexType name="audioCaptureType">
 <xs:complexContent>
  <xs:extension base="tns:mediaCaptureType">
   <xs:sequence>
        <xs:element ref="sensitivityPattern" minOccurs="0" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>


<!-- SENSITIVITY PATTERN ELEMENT -->
<xs:element name="sensitivityPattern" type="xs:string">
 <xs:annotation>
  <xs:documentation>
   Acceptable values (enumerations) for this type are managed by IANA
   in the "CLUE Schema &lt;sensitivityPattern&gt; registry", accessible
   at TBD-IANA.
  </xs:documentation>
 </xs:annotation>
</xs:element>


<!-- VIDEO CAPTURE TYPE -->
<xs:complexType name="videoCaptureType">
 <xs:complexContent>
  <xs:extension base="tns:mediaCaptureType">
   <xs:sequence>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </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="sceneViews" type="sceneViewsType" minOccurs="0"/>
  <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="mm"/>
  <xs:enumeration value="unknown"/>
  <xs:enumeration value="noscale"/>
 </xs:restriction>
</xs:simpleType>

<!-- SCENE VIEWS TYPE -->
<!-- envelope of scene views of a capture scene -->
```

```
<xs:complexType name="sceneViewsType">
 <xs:sequence>
  <xs:element name="sceneView" type="sceneViewType"
  maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>

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


<!-- CAPTURE ID LIST TYPE -->
<xs:complexType name="captureIDListType">
 <xs:sequence>
  <xs:element name="mediaCaptureIDREF" 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:unsignedLong"/>
  <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="encodingID" 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="mediaCaptureIDREF" type="xs:IDREF"
   minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="sceneViewIDREF" type="xs:IDREF"
   minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="captureSceneIDREF" 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:attribute name="mediaType" type="xs:string"/>
 <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<!-- GLOBAL VIEWS TYPE -->
<xs:complexType name="globalViewsType">
 <xs:sequence>
  <xs:element name="globalView" type="globalViewType"
  maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>

<!-- GLOBAL VIEW TYPE -->
<xs:complexType name="globalViewType">
 <xs:sequence>
   <xs:element name="sceneViewIDREF" type="xs:IDREF"
    maxOccurs="unbounded"/>
   <xs:any namespace="##other" processContents="lax" minOccurs="0"
   maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="globalViewID" type="xs:ID"/>
 <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 -->
<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="globalViews" 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:schema>
```

Following sections describe the XML schema in more detail. As a general remark, please notice that optional elements that don’t define what their absence means are intended to be associated with undefined properties.

5. <mediaCaptures>

<mediaCaptures> represents the list of one or more media captures available at the Media Provider's side. Each media capture is represented by a <mediaCapture> element (Section 11).

6. <encodingGroups>

<encodingGroups> represents the list of the encoding groups organized on the Media Provider's side. Each encoding group is represented by an <encodingGroup> element (Section 18).

7. <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 16).

8. <simultaneousSets>

<simultaneousSets> contains the simultaneous sets indicated by the Media Provider. Each simultaneous set is represented by a <simultaneousSet> element. (Section 19).

9. <globalViews>

<globalViews> 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 view" and it is represented by a <globalView> element. (Section 20).

10. <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 22).

11. <mediaCapture>

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 specializations 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="captureSceneIDREF" type="xs:IDREF"/>
     <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="policyType" minOccurs="0"/>
         <xs:element name="maxCaptures" type="maxCapturesType"
          minOccurs="0"/>
         <xs:element name="allowSubsetChoice" type="xs:boolean"
          minOccurs="0"/>
       </xs:sequence>
       <xs:element name="individual" type="xs:boolean" fixed="true"/>
     </xs:choice>
     <!-- optional fields -->
     <xs:element name="encGroupIDREF" type="xs:IDREF" minOccurs="0"/>
     <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"
      maxOccurs="unbounded"/>
     <xs:element name="mobility" type="mobilityType" minOccurs="0" />
     <xs:element ref="presentation" minOccurs="0" />
     <xs:element ref="embeddedText" minOccurs="0" />
     <xs:element ref="view" minOccurs="0" />
     <xs:element name="capturedPeople" type="capturedPeopleType"
      minOccurs="0"/>
     <xs:element name="relatedTo" type="xs:IDREF" minOccurs="0"/>
   </xs:sequence>
   <xs:attribute name="captureID" type="xs:ID" use="required"/>
   <xs:attribute name="mediaType" type="xs:string" use="required"/>
 </xs:complexType>
```

11.1. captureID attribute

The "captureID" attribute is a mandatory field containing the identifier of the media capture. Such an identifier serves as the way the capture is referenced from other data model elements (e.g., simultaneous sets, capture encodings, and others via <mediaCaptureIDREF>).

11.2. mediaType attribute

The "mediaType" attribute is a mandatory attribute specifying the media type of the capture. Common standard values are "audio", "video", "text", as defined in [RFC6838]. Other values can be provided. It is assumed that implementations agree on the interpretation of those other values. The "mediaType" attribute is as generic as possible. Here is why: (i) the basic media capture type is an abstract one; (ii) "concrete" definitions for the standard ([RFC6838]) audio, video and text capture types have been specified; (iii) a generic "otherCaptureType" type has been defined; (iv) the "mediaType" attribute has been generically defined as a string, with no particular template. From the considerations above, it is clear that if one chooses to rely on a brand new media type and wants to interoperate with others, an application-level agreement is needed on how to interpret such information.

11.3. <captureSceneIDREF>

<captureSceneIDREF> is a mandatory field containing the value of the identifier of the capture scene the media capture is defined in, i.e., the value of the sceneID (Section 16.3) attribute of that capture scene. 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 11.5). Such coordinates refer to the space of coordinates defined for the capture scene containing the capture.

11.4. <encGroupIDREF>

<encGroupIDREF> is an optional field containing the identifier of the encoding group the media capture is associated with, i.e., the value of the encodingGroupID (Section 18.3) attribute of that encoding group. Media captures that are not associated with any encoding group can not be instantiated as media streams.

11.5. <spatialInformation>

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

Captures are spatially definable when at least (i) it is possible to provide the coordinates of the device position within the telepresence room of origin (capture point) together with its capturing direction specified by a second point (point on line of capture), or (ii) it is possible to provide the represented area within the telepresence room, by listing the coordinates of the four co-planar points identifying the plane of interest (area of capture). The coordinates of the above mentioned points MUST be 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. Captures of this kind are for example those related to recordings, text captures, DVDs, registered presentations, or external streams that are played in the telepresence room and transmitted to remote sites.

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. By comparing the <spatialInformation> element of different media captures within the same capture scene, a consumer can better determine the spatial relationships between them and render them correctly. Non spatially definable captures do not embed such element in their XML description: they are instead characterized by having the <nonSpatiallyDefinable> tag set to "true" (see Section 11.6).

The definition of the spatial information type is the following:

```
<!-- SPATIAL INFORMATION TYPE -->
<xs:complexType name="spatialInformationType">
 <xs:sequence>
  <xs:element name="captureOrigin" type="captureOriginType"
   minOccurs="0"/>
  <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 <captureOrigin> contains the coordinates of the capture device that is taking the capture (i.e., the capture point), as well as, optionally, the pointing direction (i.e., the point on line of capture) (see Section 11.5.1).

The <captureArea> is an optional field containing four points defining the captured area covered by the capture (see Section 11.5.2).

The scale of the points coordinates is specified in the scale (Section 16.4) attribute of the capture scene the media capture belongs to. Indeed, all the spatially definable media captures referring to the same capture scene share the same coordinate system and express their spatial information according to the same scale.

11.5.1. <captureOrigin>

The <captureOrigin> element is used to represent the position and optionally the line of capture of a capture device. <captureOrigin> MUST be included in spatially definable audio captures, while it is optional for spatially definable video captures.

The XML Schema definition of the <captureOrigin> element type is the following:

```
<!-- CAPTURE ORIGIN TYPE -->
<xs:complexType name="captureOriginType">
  <xs:sequence>
        <xs:element name="capturePoint" type="pointType"/>
        <xs:element name="lineOfCapturePoint" type="pointType"
         minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</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>
```

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

The <captureOrigin> element includes a mandatory <capturePoint> element and an optional <lineOfCapturePoint> element, both of the type "pointType". <capturePoint> specifies the three coordinates identifying the position of the capture device. <lineOfCapturePoint> is another pointType 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. For a spatially definable video capture, if the point on line of capture is provided, it MUST belong to the region between the point of capture and the capture area. For a spatially definable audio capture, if the point on line of capture is not provided, the sensitivity pattern should be considered omnidirectional.

11.5.2. <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. <captureArea> MUST be included in the spatial information of spatially definable video captures, while it MUST NOT be associated with audio captures.

The XML representation of that area is provided through a set of four point-type elements, <bottomLeft>, <bottomRight>, <topLeft>, and <topRight> that MUST be co-planar. The four coplanar points are identified from the perspective of the capture device. The XML schema definition is the following:

```
<!-- 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>
```

11.6. <nonSpatiallyDefinable>

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

11.7. <content>

A media capture can be (i) an individual media capture or (ii) a multiple content capture (MCC). 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 an MCC, then it MAY show in its XML data model representation the <content> element. It is composed by a list of media capture identifiers ("mediaCaptureIDREF") and capture scene view identifiers ("sceneViewIDREF"), where the latter ones are used as shortcuts to refer to multiple capture identifiers. The referenced captures are used to create the MCC according to a certain strategy. If the <content> element does not appear in a MCC, or it has no child elements, then the MCC is assumed to be made of multiple sources but no information regarding those sources is provided.

```
<!-- CONTENT TYPE -->
<xs:complexType name="contentType">
 <xs:sequence>
   <xs:element name="mediaCaptureIDREF" type="xs:IDREF"
   minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="sceneViewIDREF" 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>
```

11.8. <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 all times captures coming from the same sources. It is the Media Provider that determines what the source for the captures is. In this way, the Media Provider can choose how to group together single captures for the purpose of keeping them synchronized according to the <synchronizationID> element.

11.9. <allowSubsetChoice>

<allowSubsetChoice> is an optional boolean element for multiple content captures. It indicates whether or not the Provider allows the Consumer to choose a specific subset of the captures referenced by the MCC. If this attribute is true, and the MCC references other captures, then the Consumer MAY specify in a CONFIGURE message a specific subset of those captures to be included in the MCC, and the Provider MUST then include only that subset. If this attribute is false, or the MCC does not reference other captures, then the Consumer MUST NOT select a subset. If <allowSubsetChoice> is not shown in the XML description of the MCC, its value is to be considered "false".

11.10. <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 the <mediaCaptureIDREF> list. The <policy> value is in the form of a token that indicates the policy and an index representing an instance of the policy, separated by a ":" (e.g., SoundLevel:2, RoundRobin:0, etc.). The XML schema defining the type of the <policy> element is the following:

```
<!-- POLICY TYPE -->
<xs:simpleType name="policyType">
 <xs:restriction base="xs:string">
      <xs:pattern value="([a-zA-Z0-9])+[:]([0-9])+"/>
    </xs:restriction>
</xs:simpleType>
```

At the time of writing, only two switching policies are defined in [I-D.ietf-clue-framework]:

SoundLevel: the content of the MCC is determined by a sound level detection algorithm. The loudest (active) speaker (or a previous speaker, depending on the index value) is contained in the MCC. Index 0 represents the most current instance of the policy, i.e., the currently active speaker, 1 represents the previous instance,

i.e., the previous active speaker, and so on.

RoundRobin: the content of the MCC is determined by a time based algorithm.

Other values for the <policy> element can be used. In this case, it is assumed that implementations agree on the meaning of those other values and/or those new switching policies are defined in later documents.

11.11. <maxCaptures>

<maxCaptures> is an optional element that can be used only for multiple content captures (MCC). It provides information about the number of media captures that can be represented in the multiple content capture at a time. If <maxCaptures> is not provided, all the media captures listed in the <content> element can appear at a time in the capture encoding. The type definition is provided below.

```
<!-- MAX CAPTURES TYPE -->
<xs:simpleType name="positiveShort">
        <xs:restriction base="xs:unsignedShort">
                <xs:minInclusive value="1">
                </xs:minInclusive>
        </xs:restriction>
</xs:simpleType>

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

When the "exactNumber" attribute is set to "true", 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 "<=" the <maxCaptures> value.

For instance, an audio MCC having the <maxCaptures> value set to 1 means that a media stream from the MCC will only contain audio from a single one of its constituent captures at a time. On the other hand, if the <maxCaptures> value is set to 4 and the exactNumber attribute

is set to "true", it would mean that the media stream received from the MCC will always contain a mix of audio from exactly four of its constituent captures.

11.12. <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 11.7 to Section 11.11) are mutually exclusive, according to the <choice> section within the XML Schema definition of the media capture type.

11.13. <description>

<description> is used to provide human-readable textual information. This element is included in the XML definition of media captures, capture scenes and capture scene views to the aim of providing human-readable description of, respectively, media captures, capture scenes and capture scene views. According to the data model definition of a media capture (Section 11)), zero or more <description> elements can be used, each 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 can be seen, <description> is a string element with an attribute ("lang") indicating the language used in the textual description. Such an attribute is compliant with the Language-Tag ABNF production from [RFC5646].

11.14. <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. If no priority is assigned, no assumptions regarding relative importance of the media capture can be assumed.

11.15. <lang>

<lang> is an optional element containing the language used in the capture. Zero or more <lang> elements can appear in the XML description of a media capture. Each such element has to be compliant with the Language-Tag ABNF production from [RFC5646].

11.16. <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:

static SHOULD NOT change for the duration of the CLUE session, across multiple ADVERTISEMENT messages.

dynamic MAY change in each new ADVERTISEMENT message. Can be assumed to remain unchanged until there is a new ADVERTISEMENT message.

highly-dynamic MAY change dynamically, even between consecutive ADVERTISEMENT messages. The spatial information provided in an ADVERTISEMENT message is simply a snapshot of the current values at the time when the message is sent.

11.17. <relatedTo>

The optional <relatedTo> element contains the value of the captureID attribute (Section 11.1) of the media capture to which the considered media capture refers. The media capture marked with a <relatedTo> element can be for example the translation of the referred media capture in a different language.

11.18. <view>

The <view> element is an optional tag describing what is represented in the spatial area covered by a media capture. It has been specified as a simple string with an annotation pointing to an ad hoc defined IANA registry:

```
<!-- VIEW ELEMENT -->
<xs:element name="view" type="xs:string">
 <xs:annotation>
  <xs:documentation>
   Acceptable values (enumerations) for this type are managed
   by IANA in the "CLUE Schema <view> registry",
   accessible at TBD-IANA.
  </xs:documentation>
 </xs:annotation>
</xs:element>
```

The current possible values, as per the CLUE framework document [I-D.ietf-clue-framework], are: "room", "table", "lectern", "individual", and "audience".

11.19. <presentation>

The <presentation> element is an optional tag used for media captures conveying information about presentations within the telepresence session. It has been specified as a simple string with an annotation pointing to an ad hoc defined IANA registry:

```
<!-- PRESENTATION ELEMENT -->
<xs:element name="presentation" type="xs:string">
 <xs:annotation>
  <xs:documentation>
   Acceptable values (enumerations) for this type are managed
   by IANA in the "CLUE Schema <presentation> registry",
   accessible at TBD-IANA.
  </xs:documentation>
 </xs:annotation>
</xs:element>
```

The current possible values, as per the CLUE framework document [I-D.ietf-clue-framework], are "slides" and "images".

11.20. <embeddedText>

The <embeddedText> element is a boolean element indicating that there is text embedded in the media capture (e.g., in a 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>
```

11.21. <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.

11.21.1. <personIDREF>

<personIDREF> contains the identifier of the represented person, i.e., the value of the related personID attribute (Section 21.1.1). Metadata about the represented participant can be retrieved by accessing the <people> list (Section 21).

12. 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 ref="sensitivityPattern" minOccurs="0" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

An example of audio-specific information that can be included is represented by the <sensitivityPattern> element. (Section 12.1).

12.1. <sensitivityPattern>

The <sensitivityPattern> element is an optional field describing the characteristics of the nominal sensitivity pattern of the microphone capturing the audio signal. It has been specified as a simple string with an annotation pointing to an ad hoc defined IANA registry:

```
<!-- SENSITIVITY PATTERN ELEMENT -->
<xs:element name="sensitivityPattern" type="xs:string">
 <xs:annotation>
  <xs:documentation>
   Acceptable values (enumerations) for this type are managed by IANA
   in the "CLUE Schema <sensitivityPattern> registry", accessible
   at TBD-IANA.
  </xs:documentation>
 </xs:annotation>
</xs:element>
```

The current possible values, as per the CLUE framework document [I-D.ietf-clue-framework], are "uni", "shotgun", "omni", "figure8", "cardioid" and "hyper-cardioid".

13. Video captures

Video captures, similarly to audio captures, extend the information of a generic media capture with video-specific features.

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:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:extension>
```

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

14. Text captures

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

There are no known properties of a text-based media which aren't already covered by the generic mediaCaptureType. Text captures are hence defined as follows:

```
<!-- TEXT CAPTURE TYPE -->
<xs:complexType name="textCaptureType">
 <xs:complexContent>
  <xs:extension base="tns:mediaCaptureType">
  <xs:sequence>
  <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Text captures MUST be marked as non spatially definable (i.e., they MUST present in their XML description the <nonSpatiallyDefinable> (Section 11.6) element set to "true").

15. Other capture types

Other media capture types can be described by using the CLUE data model. They can be represented by exploiting the "otherCaptureType" type. This media capture type is conceived to be filled in with elements defined within extensions of the current schema, i.e., with elements defined in other XML schemas (see Section 24 for an example). The otherCaptureType inherits all the features envisioned for the abstract mediaCaptureType.

The XML Schema representation of the otherCaptureType is the following:

```
<!-- OTHER CAPTURE TYPE -->
<xs:complexType name="otherCaptureType">
 <xs:complexContent>
  <xs:extension base="tns:mediaCaptureType">
  <xs:sequence>
  <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

When defining new media capture types that are going to be described by means of the <otherMediaCapture> element, spatial properties of such new media capture types SHOULD be defined (e.g., whether or not they are spatially definable, whether or not they should be associated with an area of capture, or other properties that may be defined).

16. <captureScene>

A Media Provider organizes the available captures 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 media captures and capture scene views, that are sets of media captures of the same media type. Each capture scene view is 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="sceneViews" type="sceneViewsType" minOccurs="0"/>
  <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>
```

Each capture scene is identified by a "sceneID" attribute. The <captureScene> element can contain zero or more textual <description> elements, defined as in Section 11.13. Besides <description>, there is the optional <sceneInformation> element (Section 16.1), which contains structured information about the scene in the vcard format, and the optional <sceneViews> element (Section 16.2), which is the list of the capture scene views. When no <sceneViews> is provided, the capture scene is assumed to be made of all the media captures which contain the value of its sceneID attribute in their mandatory captureSceneIDREF attribute.

16.1. <sceneInformation>

The <sceneInformation> element contains optional information about the capture scene according to the vcard format, as specified in the Xcard RFC [RFC6351].

16.2. <sceneViews>

The <sceneViews> element is a mandatory field of a capture scene containing the list of scene views. Each scene view is represented by a <sceneView> element (Section 17).

```
<!-- SCENE VIEWS TYPE -->
<!-- envelope of scene views of a capture scene -->
<xs:complexType name="sceneViewsType">
 <xs:sequence>
  <xs:element name="sceneView" type="sceneViewType"
  maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
```

16.3. sceneID attribute

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

16.4. 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:

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

"unknown" - the scale is the same for every media capture in the capture scene but the unity of measure is undefined. Systems which are not aware of specific physical dimensions yet still know relative distances should select "unknown" in the scale attribute of the capture scene to be described.

"noscale" - there is no 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="mm"/>
  <xs:enumeration value="unknown"/>
  <xs:enumeration value="noscale"/>
 </xs:restriction>
</xs:simpleType>
```

17. <sceneView>

A <sceneView> element represents a capture scene view, which contains a set of media captures of the same media type describing a capture scene.

A <sceneView> element is characterized as follows.

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

One or more optional <description> elements provide human-readable information about what the scene view contains. <description> is defined as already seen in Section 11.13.

The remaining child elements are described in the following subsections.

17.1. <mediaCaptureIDs>

The <mediaCaptureIDs> is the list of the identifiers of the media captures included in the scene view. It is an element of the captureIDListType type, which is defined as a sequence of <mediaCaptureIDREF>, each 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="mediaCaptureIDREF" type="xs:IDREF"
  maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
```

17.2. sceneViewID attribute

The sceneViewID attribute is a mandatory attribute containing the identifier of the capture scene view represented by the <sceneView> element.

18. <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 media captures. The definition of the <encodingGroup> element is the following:

```
   <!-- ENCODING GROUP TYPE -->
   <xs:complexType name="encodingGroupType">
    <xs:sequence>
     <xs:element name="maxGroupBandwidth" type="xs:unsignedLong"/>
     <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.

18.1. <maxGroupBandwidth>

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

18.2. <encodingIDList>

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

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

18.3. encodingGroupID attribute

The encodingGroupID attribute contains the identifier of the encoding group.

19. <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="mediaCaptureIDREF" type="xs:IDREF"
   minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="sceneViewIDREF" type="xs:IDREF"
   minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="captureSceneIDREF" 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:attribute name="mediaType" type="xs:string"/>
 <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

Besides the identifiers of the captures (<mediaCaptureIDREF> elements), also the identifiers of capture scene views and of capture scene can be exploited as shortcuts (<sceneViewIDREF> and <captureSceneIDREF> elements). As an example, let’s consider the situation where there are two capture scene views (S1 and S7). S1 contains captures AC11, AC12, AC13. S7 contains captures AC71, AC72. Provided that AC11, AC12, AC13, AC71, AC72 can be simultaneously sent to the media consumer, instead of having 5 <mediaCaptureIDREF> elements listed in the simultaneous set (i.e., one <mediaCaptureIDREF> for AC11, one for AC12, and so on), there can be just two <sceneViewIDREF> elements (one for S1 and one for S7).

19.1. setID attribute

The "setID" attribute is a mandatory field containing the identifier of the simultaneous set.

19.2. mediaType attribute

The "mediaType" attribute is an optional attribute containing the media type of the captures referenced by the simultaneous set.

When only capture scene identifiers are listed within a simultaneous set, the media type attribute MUST appear in the XML description in order to determine which media captures can be simultaneously sent together.

19.3. <mediaCaptureIDREF>

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

19.4. <sceneViewIDREF>

<sceneViewIDREF> contains the identifier of the scene view containing a group of captures that are able to be sent simultaneously with the other captures of the simultaneous set.

19.5. <captureSceneIDREF>

<captureSceneIDREF> contains the identifier of the capture scene where all the included captures of a certain media type are able to be sent together with the other captures of the simultaneous set.

20. <globalView>

<globalView> is a set of captures of the same media type representing a summary of the complete Media Provider's offer. The content of a global view is expressed by leveraging only scene view identifiers, put within <sceneViewIDREF> elements. Each global view is identified by a unique identifier within the "globalViewID" attribute.

```
<!-- GLOBAL VIEW TYPE -->
<xs:complexType name="globalViewType">
 <xs:sequence>
   <xs:element name="sceneViewIDREF" type="xs:IDREF"
    maxOccurs="unbounded"/>
   <xs:any namespace="##other" processContents="lax" minOccurs="0"
   maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="globalViewID" type="xs:ID"/>
 <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

21. <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>
```

21.1. <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 21.1.2) and his conference role(s) (via one or more <personType> elements, see Section 21.1.3). Furthermore, it has a mandatory "personID" attribute (Section 21.1.1).

```
<!-- PERSON TYPE -->
<xs:complexType name="personType">
 <xs:sequence>
  <xs:element name="personInfo" type="xcard:vcardType" maxOccurs="1"
   minOccurs="0"/>
  <xs:element ref="personType" 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>
```

21.1.1. personID attribute

The "personID" attribute carries the identifier of a represented person. Such an identifier can be used to refer to the participant, as in the <capturedPeople> element in the media captures representation (Section 11.21).

21.1.2. <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 in Appendix A of [I-D.ietf-ecrit-additional-data]. As such schema specifies, the <fn> element within <vcard> is mandatory.

21.1.3. <personType>

The value of the <personType> element determines the role of the represented participant within the telepresence session organization. It has been specified as a simple string with an annotation pointing to an ad hoc defined IANA registry:

```
<!-- PERSON TYPE ELEMENT -->
<xs:element name="personType" type="xs:string">
 <xs:annotation>
  <xs:documentation>
   Acceptable values (enumerations) for this type are managed
   by IANA in the "CLUE Schema <personType> registry",
   accessible at TBD-IANA.
  </xs:documentation>
 </xs:annotation>
</xs:element>
```

The current possible values, as per the CLUE framework document [I-D.ietf-clue-framework], are: "presenter", "timekeeper", "attendee", "minute taker", "translator", "chairman", "vice-chairman", "observer".

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

22. <captureEncoding>

A capture encoding is given from the association of a media capture with an individual encoding, to form a capture stream as defined in [I-D.ietf-clue-framework]. Capture encodings are used within CONFIGURE messages from a Media Consumer to a Media Provider for representing the streams desired by the Media Consumer. For each desired stream, the Media Consumer needs to be allowed to specify: (i) the capture identifier of the desired capture that has been advertised by the Media Provider; (ii) the encoding identifier of the encoding to use, among those advertised by the Media Provider; (iii) optionally, in case of multi-content captures, the list of the capture identifiers of the desired captures. All the mentioned identifiers are intended to be included in the ADVERTISEMENT message that the CONFIGURE message refers to. The XML model of <captureEncoding> 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>
```

22.1. <captureID>

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

22.2. <encodingID>

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

22.3. <configuredContent>

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

23. <clueInfo>

The <clueInfo> element includes all the information needed to represent the Media Provider’s description of its telepresence capabilities according to the CLUE framework. Indeed, it is made by:

- the list of the available media captures (<mediaCaptures> (Section 5))
- the list of encoding groups (<encodingGroups> (Section 6))

the list of capture scenes (<captureScenes> (Section 7))

the list of simultaneous transmission sets (<simultaneousSets> (Section 8))

the list of global views sets (<globalViews> (Section 9))

meta data about the participants represented in the telepresence session (<people> (Section 21))

It has been conceived only for data model testing purposes and, though it resembles the body of an ADVERTISEMENT message, it is not actually used in the CLUE protocol message definitions. The telepresence capabilities descriptions compliant to this data model specification that can be found in Section 27 and Section 28 are provided by using the <clueInfo> element.

```
<!-- 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="globalViews" 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>
```

24. 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. Extensibility was purposefully favored as much as possible based on expectations about custom

implementations. Hence, the schema offers people enough flexibility as to define custom extensions, without losing compliance with the standard. This is achieved by leveraging <xs:any> elements and <xs: anyAttribute> attributes, which is a common approach with schemas, still matching the UPA (Unique Particle Attribution) constraint.

24.1. Example of extension

When extending the CLUE data model, a new schema with a new namespace associated with it needs to be specified.

In the following, an example of extension is provided. The extension defines a new audio capture attribute ("newAudioFeature") and an attribute for characterizing the captures belonging to an "otherCaptureType" defined by the user. An XML document compliant with the extension is also included. The XML file results validated against the current CLUE data model schema.

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

<!-- Import xcard XML schema -->
<xs:import namespace="urn:ietf:params:xml:ns:vcard-4.0"
schemaLocation=
 "http://www.iana.org/assignments/xml-registry/schema/vcard-4.0.xsd"/>

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

<!-- ELEMENT DEFINITIONS -->
<xs:element name="newAudioFeature" type="xs:string"/>
<xs:element name="otherMediaCaptureTypeFeature" type="xs:string"/>

</xs:schema>
```

```
<?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"
xmlns:ns3="urn:ietf:params:xml:ns:clue-info-ext"
clueInfoID="NapoliRoom">
    <mediaCaptures>
        <mediaCapture
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="audioCaptureType"
        captureID="AC0"
        mediaType="audio">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
            <individual>true</individual>
            <encGroupIDREF>EG1</encGroupIDREF>
            <ns3:newAudioFeature>newAudioFeatureValue
            </ns3:newAudioFeature>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="otherCaptureType"
        captureID="OMC0"
        mediaType="other media type">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
            <encGroupIDREF>EG1</encGroupIDREF>
            <ns3:otherMediaCaptureTypeFeature>OtherValue
            </ns3:otherMediaCaptureTypeFeature>
        </mediaCapture>
    </mediaCaptures>
    <encodingGroups>
        <encodingGroup encodingGroupID="EG1">
            <maxGroupBandwidth>300000</maxGroupBandwidth>
            <encodingIDList>
                <encodingID>ENC4</encodingID>
                <encodingID>ENC5</encodingID>
            </encodingIDList>
        </encodingGroup>
    </encodingGroups>
    <captureScenes>
        <captureScene scale="unknown" sceneID="CS1"/>
    </captureScenes>
</clueInfo>
```

25. Security considerations

This document defines an XML Schema data model for telepresence scenarios. The modeled information is identified in the CLUE framework as necessary in order to enable a full-fledged media stream negotiation and rendering. Indeed, the XML elements herein defined are used within CLUE protocol messages to describe both the media streams representing the Media Provider's telepresence offer and the desired selection requested by the Media Consumer. Security concerns described in [I-D.ietf-clue-framework], Section 15, apply to this document.

Data model information carried within CLUE messages SHOULD be accessed only by authenticated endpoints. Indeed, authenticated access is strongly advisable, especially if you convey information about individuals (<personalInfo>) and/or scenes (<sceneInformation>). There might be more exceptions, depending on the level of criticality that is associated with the setup and configuration of a specific session. In principle, one might even decide that no protection at all is needed for a particular session; here is why authentication has not been identified as a mandatory requirement.

Going deeper into details, some information published by the Media Provider might reveal sensitive data about who and what is represented in the transmitted streams. The vCard included in the <personInfo> elements (Section 21.1) mandatorily contains the identity of the represented person. Optionally vCards can also carry the person's contact addresses, together with his/her photo and other personal data. Similar privacy-critical information can be conveyed by means of <sceneInformation> elements (Section 16.1) describing the capture scenes. The <description> elements (Section 11.13) also can specify details about the content of media captures, capture scenes and scene views that should be protected.

Integrity attacks to the data model information encapsulated in CLUE messages can invalidate the success of the telepresence session's setup by misleading the Media Consumer's and Media Provider's interpretation of the offered and desired media streams.

The assurance of the authenticated access and of the integrity of the data model information is up to the involved transport mechanisms, namely the CLUE protocol [I-D.ietf-clue-protocol] and the CLUE data channel [I-D.ietf-clue-datachannel].

XML parsers need to be robust with respect to malformed documents. Reading malformed documents from unknown or untrusted sources could result in an attacker gaining privileges of the user running the XML

parser. In an extreme situation, the entire machine could be compromised.

26. IANA considerations

This document registers a new XML namespace, a new XML schema, the MIME type for the schema and four new registries associated, respectively, with acceptable <view>, <presentation>, <sensitivityPattern> and <personType> values.

26.1. XML namespace registration

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

Registrant Contact: IETF CLUE Working Group <clue@ietf.org>, Roberta Presta <roberta.presta@unina.it>

XML:

BEGIN

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
 "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="content-type"
        content="text/html;charset=iso-8859-1"/>
  <title> CLUE Data Model Namespace</title>
 </head>
 <body>
  <h1>Namespace for CLUE Data Model</h1>
  <h2>urn:ietf:params:xml:ns:clue-info</h2>
  <p>See
   <a href="http://www.rfc-editor.org/rfc/rfcXXXX.txt"> RFC XXXX</a>.
   <!--[[NOTE TO IANA/RFC-EDITOR: Please update RFC URL
    and replace XXXX with the RFC number for this specification.]]-->
  </p>
 </body>
</html>
```

END

26.2. XML Schema registration

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

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

Registrant Contact: CLUE working group (clue@ietf.org), Roberta Presta (roberta.presta@unina.it).

Schema: The XML for this schema can be found as the entirety of Section 4 of this document.

26.3. MIME Media Type Registration for "application/clue_info+xml"

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

To: ietf-types@iana.org

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

MIME media type name: application

MIME subtype name: clue_info+xml

Required parameters: (none)

Optional parameters: charset
Same as the charset parameter of "application/xml" as specified in [RFC7303], Section 3.2.

Encoding considerations: Same as the encoding considerations of "application/xml" as specified in [RFC7303], Section 3.2.

Security considerations: This content type is designed to carry data related to telepresence information. Some of the data could be considered private. This media type does not provide any protection and thus other mechanisms such as those described in Section 25 are required to protect the data. This media type does not contain executable content.

Interoperability considerations: None.

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

Applications that use this media type: CLUE-capable telepresence systems.

Additional Information: Magic Number(s): (none),
File extension(s): .clue,
Macintosh File Type Code(s): TEXT.

Person & email address to contact for further information: Roberta Presta (roberta.presta@unina.it).

Intended usage: LIMITED USE

Author/Change controller: The IETF

Other information: This media type is a specialization of application/xml [RFC7303], and many of the considerations described there also apply to application/clue_info+xml.

26.4. Registry for acceptable <view> values

IANA is requested to create a registry of acceptable values for the the <view> tag as defined in Section 11.18. The initial values for this registry are "room", "table", "lectern", "individual", and "audience".

New values are assigned by Expert Review per [RFC5226]. This reviewer will ensure that the requested registry entry conforms to the prescribed formatting.

IANA is further requested to update this draft with the URL to the new registry in Section 11.18, marked as "TBD-IANA".

26.5. Registry for acceptable <presentation> values

IANA is requested to create a registry of acceptable values for the the <presentation> tag as defined in Section 11.19. The initial values for this registry are "slides" and "images".

New values are assigned by Expert Review per [RFC5226]. This reviewer will ensure that the requested registry entry conforms to the prescribed formatting.

IANA is further requested to update this draft with the URL to the new registry in Section 11.19, marked as "TBD-IANA".

26.6. Registry for acceptable <sensitivityPattern> values

IANA is requested to create a registry of acceptable values for the the <sensitivityPattern> tag as defined in Section 12.1. The initial values for this registry are "uni", "shotgun", "omni", "figure8", "cardioid" and "hyper-cardioid".

New values are assigned by Expert Review per [RFC5226]. This reviewer will ensure that the requested registry entry conforms to the prescribed formatting.

IANA is further requested to update this draft with the URL to the new registry in Section 12.1, marked as "TBD-IANA".

26.7. Registry for acceptable <personType> values

IANA is requested to create a registry of acceptable values for the the <personType> tag as defined in Section 21.1.3. The initial values for this registry are "presenter", "timekeeper", "attendee", "minute taker", "translator", "chairman", "vice-chairman", "observer".

New values are assigned by Expert Review per [RFC5226]. This reviewer will ensure that the requested registry entry conforms to the prescribed formatting.

IANA is further requested to update this draft with the URL to the new registry in Section 21.1.3, marked as "TBD-IANA".

27. 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 ADVERTISEMENT.

There are three cameras, where the central one is also capable of capturing a zoomed-out view of the overall telepresence room. Besides the three video captures coming from the cameras, the Media Provider makes available a further multi-content capture of 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 role details are also provided.

Media captures are arranged into four capture scene views:

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, 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). On the other hand, VC3 and VC4 cannot be simultaneous either, since VC3, the loudest segment of the room, might be at a certain point in time focusing on the central part of the room, i.e., the same as VC1. The simultaneous sets would then be the following:

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

SS2 made by VC0, VC2, VC4

```
<?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="audioCaptureType" captureID="AC0" mediaType="audio">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                    <lineOfCapturePoint>
                        <x>0.0</x>
                        <y>1.0</y>
                        <z>10.0</z>
                    </lineOfCapturePoint>
                </captureOrigin>
            </spatialInformation>
            <individual>true</individual>
```

```
            <encGroupIDREF>EG1</encGroupIDREF>
            <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>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC0" mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>-2.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                </captureOrigin>
                <captureArea>
                        <bottomLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                                <x>-1.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                                <x>-1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
```

```
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
            <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>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="VC1" mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                </captureOrigin>
                <captureArea>
                        <bottomLeft>
                                <x>-1.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                                <x>1.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                                <x>-1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                                <x>1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <individual>true</individual>
```

```
            <encGroupIDREF>EG0</encGroupIDREF>
            <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>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="VC2" mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>2.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                </captureOrigin>
                <captureArea>
                        <bottomLeft>
                                <x>1.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                                <x>1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
```

```
            <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>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="VC3" mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureArea>
                        <bottomLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <content>
                <sceneViewIDREF>SE1</sceneViewIDREF>
            </content>
            <policy>SoundLevel:0</policy>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">loudest room segment</description>
            <priority>2</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>individual</view>
```

```
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="VC4" mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                </captureOrigin>
                <captureArea>
                        <bottomLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>7.0</z>
                        </bottomLeft>
                        <bottomRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>7.0</z>
                        </bottomRight>
                        <topLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>13.0</z>
                        </topLeft>
                        <topRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>13.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">zoomed out view of all people in the
            room</description>
            <priority>2</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>room</view>
            <capturedPeople>
                <personIDREF>alice</personIDREF>
                <personIDREF>bob</personIDREF>
                <personIDREF>ciccio</personIDREF>
```

```
            </capturedPeople>
        </mediaCapture>
    </mediaCaptures>
    <encodingGroups>
        <encodingGroup encodingGroupID="EG0">
            <maxGroupBandwidth>600000</maxGroupBandwidth>
            <encodingIDList>
                <encodingID>ENC1</encodingID>
                <encodingID>ENC2</encodingID>
                <encodingID>ENC3</encodingID>
            </encodingIDList>
        </encodingGroup>
        <encodingGroup encodingGroupID="EG1">
            <maxGroupBandwidth>300000</maxGroupBandwidth>
            <encodingIDList>
                <encodingID>ENC4</encodingID>
                <encodingID>ENC5</encodingID>
            </encodingIDList>
        </encodingGroup>
    </encodingGroups>
    <captureScenes>
        <captureScene scale="unknown" sceneID="CS1">
            <sceneViews>
                <sceneView sceneViewID="SE1">
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC0</mediaCaptureIDREF>
                        <mediaCaptureIDREF>VC1</mediaCaptureIDREF>
                        <mediaCaptureIDREF>VC2</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE2">
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC3</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE3">
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC4</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE4">
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>AC0</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
            </sceneViews>
        </captureScene>
    </captureScenes>
```

```
    <simultaneousSets>
        <simultaneousSet setID="SS1">
            <mediaCaptureIDREF>VC3</mediaCaptureIDREF>
            <sceneViewIDREF>SE1</sceneViewIDREF>
        </simultaneousSet>
        <simultaneousSet setID="SS2">
            <mediaCaptureIDREF>VC0</mediaCaptureIDREF>
            <mediaCaptureIDREF>VC2</mediaCaptureIDREF>
            <mediaCaptureIDREF>VC4</mediaCaptureIDREF>
        </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>
```

28. MCC example

Enhancing the scenario presented in the previous example, the Media Provider 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="audioCaptureType" captureID="AC0" mediaType="audio">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                    <lineOfCapturePoint>
                        <x>0.0</x>
                        <y>1.0</y>
                        <z>10.0</z>
                    </lineOfCapturePoint>
                </captureOrigin>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG1</encGroupIDREF>
            <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>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="VC0" mediaType="video">
```

```
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.5</x>
                        <y>1.0</y>
                        <z>0.5</z>
                    </capturePoint>
                    <lineOfCapturePoint>
                        <x>0.5</x>
                        <y>0.0</y>
                        <z>0.5</z>
                    </lineOfCapturePoint>
                </captureOrigin>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
            <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>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="videoCaptureType" captureID="VC1" mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                </captureOrigin>
                <captureArea>
                        <bottomLeft>
                                <x>-1.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                                <x>1.0</x>
                        <y>20.0</y>
```

```
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                                <x>-1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                                <x>1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                    </captureArea>
                </spatialInformation>
                <individual>true</individual>
                <encGroupIDREF>EG0</encGroupIDREF>
                <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>
            </mediaCapture>
            <mediaCapture
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="videoCaptureType" captureID="VC2" mediaType="video">
                <captureSceneIDREF>CS1</captureSceneIDREF>
                <spatialInformation>
                    <captureOrigin>
                            <capturePoint>
                            <x>2.0</x>
                            <y>0.0</y>
                            <z>10.0</z>
                        </capturePoint>
                    </captureOrigin>
                    <captureArea>
                            <bottomLeft>
                                    <x>1.0</x>
                            <y>20.0</y>
                            <z>9.0</z>
                            </bottomLeft>
                            <bottomRight>
                                    <x>3.0</x>
                            <y>20.0</y>
                            <z>9.0</z>
```

```
                        </bottomRight>
                        <topLeft>
                                <x>1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                    </captureArea>
                </spatialInformation>
                <individual>true</individual>
                <encGroupIDREF>EG0</encGroupIDREF>
                <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>
            </mediaCapture>
            <mediaCapture
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="videoCaptureType" captureID="VC3" mediaType="video">
                <captureSceneIDREF>CS1</captureSceneIDREF>
                <spatialInformation>
                    <captureArea>
                            <bottomLeft>
                                    <x>-3.0</x>
                            <y>20.0</y>
                            <z>9.0</z>
                            </bottomLeft>
                            <bottomRight>
                                    <x>3.0</x>
                            <y>20.0</y>
                            <z>9.0</z>
                            </bottomRight>
                            <topLeft>
                                    <x>-3.0</x>
                            <y>20.0</y>
                            <z>11.0</z>
                            </topLeft>
                            <topRight>
                                    <x>3.0</x>
```

```
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                    </captureArea>
                </spatialInformation>
                <content>
                    <sceneViewIDREF>SE1</sceneViewIDREF>
                </content>
                <policy>SoundLevel:0</policy>
                <encGroupIDREF>EG0</encGroupIDREF>
                <description lang="en">loudest room segment</description>
                <priority>2</priority>
                <lang>it</lang>
                <mobility>static</mobility>
                <view>individual</view>
            </mediaCapture>
            <mediaCapture
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:type="videoCaptureType" captureID="VC4" mediaType="video">
                <captureSceneIDREF>CS1</captureSceneIDREF>
                <spatialInformation>
                    <captureOrigin>
                        <capturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                        </capturePoint>
                    </captureOrigin>
                    <captureArea>
                        <bottomLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>7.0</z>
                        </bottomLeft>
                        <bottomRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>7.0</z>
                        </bottomRight>
                        <topLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>13.0</z>
                        </topLeft>
                        <topRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>13.0</z>
```

```
                    </topRight>
                </captureArea>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">
              zoomed out view of all people in the room
            </description>
            <priority>2</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>room</view>
            <capturedPeople>
                <personIDREF>alice</personIDREF>
                <personIDREF>bob</personIDREF>
                <personIDREF>ciccio</personIDREF>
            </capturedPeople>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC5" mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureArea>
                        <bottomLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <content>
                <sceneViewIDREF>SE1</sceneViewIDREF>
```

```
            </content>
            <policy>SoundLevel:1</policy>
            <description lang="en">penultimate loudest room segment
            </description>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>individual</view>
         </mediaCapture>
         <mediaCapture
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="videoCaptureType" captureID="VC6" mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureArea>
                        <bottomLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                                <x>-3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                                <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <content>
                <sceneViewIDREF>SE1</sceneViewIDREF>
            </content>
            <policy>SoundLevel:2</policy>
            <description lang="en">last but two loudest room segment
            </description>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>individual</view>
         </mediaCapture>
         <mediaCapture
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
          xsi:type="videoCaptureType" captureID="VC7" mediaType="video">
             <captureSceneIDREF>CS1</captureSceneIDREF>
             <spatialInformation>
                 <captureArea>
                         <bottomLeft>
                                 <x>-3.0</x>
                         <y>20.0</y>
                         <z>9.0</z>
                         </bottomLeft>
                         <bottomRight>
                                 <x>3.0</x>
                         <y>20.0</y>
                         <z>9.0</z>
                         </bottomRight>
                         <topLeft>
                                 <x>-3.0</x>
                         <y>20.0</y>
                         <z>11.0</z>
                         </topLeft>
                         <topRight>
                                 <x>3.0</x>
                         <y>20.0</y>
                         <z>11.0</z>
                         </topRight>
                 </captureArea>
             </spatialInformation>
             <content>
                 <mediaCaptureIDREF>VC3</mediaCaptureIDREF>
                 <mediaCaptureIDREF>VC5</mediaCaptureIDREF>
                 <mediaCaptureIDREF>VC6</mediaCaptureIDREF>
             </content>
             <maxCaptures exactNumber="true">3</maxCaptures>
             <encGroupIDREF>EG0</encGroupIDREF>
             <description lang="en">big picture of the current speaker +
             pips about previous speakers</description>
             <priority>3</priority>
             <lang>it</lang>
             <mobility>static</mobility>
             <view>individual</view>
         </mediaCapture>
     </mediaCaptures>
     <encodingGroups>
         <encodingGroup encodingGroupID="EG0">
             <maxGroupBandwidth>600000</maxGroupBandwidth>
             <encodingIDList>
                 <encodingID>ENC1</encodingID>
                 <encodingID>ENC2</encodingID>
                 <encodingID>ENC3</encodingID>
```

```
            </encodingIDList>
        </encodingGroup>
        <encodingGroup encodingGroupID="EG1">
            <maxGroupBandwidth>300000</maxGroupBandwidth>
            <encodingIDList>
                <encodingID>ENC4</encodingID>
                <encodingID>ENC5</encodingID>
            </encodingIDList>
        </encodingGroup>
    </encodingGroups>
    <captureScenes>
        <captureScene scale="unknown" sceneID="CS1">
            <sceneViews>
                <sceneView sceneViewID="SE1">
                    <description lang="en">participants' individual
                    videos</description>
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC0</mediaCaptureIDREF>
                        <mediaCaptureIDREF>VC1</mediaCaptureIDREF>
                        <mediaCaptureIDREF>VC2</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE2">
                    <description lang="en">loudest segment of the
                    room</description>
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC3</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE5">
                    <description lang="en">loudest segment of the
                    room + pips</description>
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC7</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE4">
                    <description lang="en">room audio</description>
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>AC0</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE3">
                    <description lang="en">room video</description>
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC4</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
```

```
            </sceneViews>
        </captureScene>
    </captureScenes>
    <simultaneousSets>
        <simultaneousSet setID="SS1">
            <mediaCaptureIDREF>VC3</mediaCaptureIDREF>
            <mediaCaptureIDREF>VC7</mediaCaptureIDREF>
            <sceneViewIDREF>SE1</sceneViewIDREF>
        </simultaneousSet>
        <simultaneousSet setID="SS2">
            <mediaCaptureIDREF>VC0</mediaCaptureIDREF>
            <mediaCaptureIDREF>VC2</mediaCaptureIDREF>
            <mediaCaptureIDREF>VC4</mediaCaptureIDREF>
        </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>
```

29. Diff with draft-ietf-clue-data-model-schema-16 version

As per Alexey Melnikov's and Stefan Winter's comments: replaced wrong references to RFC2119 in section 11.3 and section 11.5. The updated reference is to RFC5646.

30. Diff with draft-ietf-clue-data-model-schema-15 version

Applied modifications as per the following reviews: (i) Alexey Melnikov's discuss and comments (abstract amendments, typo corrections, insertion of references, etc.); (ii) Kathleen Moriarty's discuss and comments (amendments to the Security Considerations section); (iii) Stefan Winter's OPS-DIR review (use of enumerated types in the schema).

31. Diff with draft-ietf-clue-data-model-schema-14 version

Applied modifications as per the following reviews: (i) Henry S. Thompson's APPS-DIR review; (ii) Stefan Winter's OPS-DIR review; (iii) Francis Dupont's GEN-ART review; (iv) Rich Salz's review (as part of the security directorate's ongoing effort to review all IETF documents being processed by the IESG.)

32. Diff with draft-ietf-clue-data-model-schema-13 version

Applied modifications as per the latest Area Director (Alissa Cooper's) review comments.

33. Diff with draft-ietf-clue-data-model-schema-12 version

Removed some typos and inconsistencies. Applied modifications as per Alissa Cooper's review comments.

34. Diff with draft-ietf-clue-data-model-schema-11 version

Applied modifications as per Mark Duckworth's review (example corrections and maxCapturesType modification)

maxCapturesType has been changed from positiveInteger to unsignedShort excluding value 0.

35. Diff with draft-ietf-clue-data-model-schema-10 version

Minor modifications have been applied to address nits at page https://www.ietf.org/tools/idnits?url=https://www.ietf.org/archive/id/draft-ietf-clue-data-model-schema-10.txt.

36. Diff with draft-ietf-clue-data-model-schema-09 version

- We have introduced a <captureOrigin> element containing a mandatory <capturePoint> and an optional <lineOfCapturePoint> in the definition of <spatialInformation> as per Paul's review
- A new type definition for switching policies (resembled by <policy> element) has been provided in order to have acceptable values in the form of "token:index".
- Minor modifications suggested in WGLC reviews have been applied.

37. Diff with draft-ietf-clue-data-model-schema-08 version

- Typos correction

38. Diff with draft-ietf-clue-data-model-schema-07 version

- IANA Considerations: text added
- maxCaptureEncodings removed
- personTypeType values aligned with CLUE framework
- allowSubsetChoice added for multiple content captures
- embeddedText moved from videoCaptureType definition to mediaCaptureType definition
- typos removed from section Terminology

39. Diff with draft-ietf-clue-data-model-schema-06 version

- Capture Scene Entry/Entries renamed as Capture Scene View/Views in the text, <sceneEntry>/<sceneEntries> renamed as <sceneView>/ <sceneViews> in the XML schema.
- Global Scene Entry/Entries renamed as Global View/Views in the text, <globalSceneEntry>/<globalSceneEntries> renamed as <globalView>/<globalViews>
- Security section added.
- Extensibility: a new type is introduced to describe other types of media capture (otherCaptureType), text and example added.
- Spatial information section updated: capture point optional, text now is coherent with the framework one.

- Audio capture description: <sensitivityPattern> added, <audioChannelFormat> removed, <captureArea> disallowed.
- Simultaneous set definition: added <captureSceneIDREF> to refer to capture scene identifiers as shortcuts and an optional mediaType attribute which is mandatory to use when only capture scene identifiers are listed.
- Encoding groups: removed the constraint of the same media type.
- Updated text about media captures without <encodingGroupIDREF> (optional in the XML schema).
- "mediaType" attribute removed from homogeneous groups of capture (scene views and globlal views)
- "mediaType" attribute removed from the global view textual description.
- "millimeters" scale value changed in "mm"

40. 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>.

41. 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.

42. 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 has been provided.

A multiple content capture example has been added.

Obsolete comments and references have been removed.

43. Acknowledgments

The authors thank all the CLUErs for their precious feedbacks and support. Thanks also to Alissa Cooper, whose AD reviews helped us improve the quality of the document.

44. References

44.1. Normative References

[I-D.ietf-clue-datachannel] Holmberg, C., "CLUE Protocol data channel", draft-ietf-clue-datachannel-14 (work in progress), August 2016.

[I-D.ietf-clue-framework] Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-25 (work in progress), January 2016.

[I-D.ietf-clue-protocol] Presta, R. and S. Romano, "CLUE protocol", draft-ietf-clue-protocol-08 (work in progress), May 2016.

[I-D.ietf-ecrit-additional-data] Gellens, R., Rosen, B., Tschofenig, H., Marshall, R., and J. Winterbottom, "Additional Data Related to an Emergency Call", draft-ietf-ecrit-additional-data-38 (work in progress), April 2016.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <http://www.rfc-editor.org/info/rfc5646>.

[RFC6351] Perreault, S., "xCard: vCard XML Representation", RFC 6351, DOI 10.17487/RFC6351, August 2011, <http://www.rfc-editor.org/info/rfc6351>.

[RFC7303] Thompson, H. and C. Lilley, "XML Media Types", RFC 7303, DOI 10.17487/RFC7303, July 2014, <h

ttp://www.rfc-editor.org/info/
rfc7303>.

44.2. Informative References

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, DOI 10.17487/RFC4353, February 2006, <http://www.rfc-editor.org/info/rfc4353>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <http://www.rfc-editor.org/info/rfc6838>.

[RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <http://www.rfc-editor.org/info/rfc7667>.

Authors' Addresses

Roberta Presta
University of Napoli
Via Claudio 21
Napoli 80125
Italy

EMail: roberta.presta@unina.it

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

EMail: spromano@unina.it

CLUE WG
Internet Draft
Intended status: Standards Track
Expires: August 10, 2014

M. Duckworth, Ed.
Polycom
A. Pepperell
Acano
S. Wenger
Vidyo
February 10, 2014

Framework for Telepresence Multi-Streams
draft-ietf-clue-framework-14.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 August 10, 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.....15
7. Media Captures and Capture Scenes.....16
7.1. Media Captures.....16
7.1.1. Media Capture Attributes.....17
7.2. Multiple Content Capture.....22
7.2.1. MCC Attributes.....23
7.3. Capture Scene.....27
7.3.1. Capture Scene attributes.....30
7.3.2. Capture Scene Entry attributes.....31
8. Simultaneous Transmission Set Constraints.....31
9. Encodings.....33
9.1. Individual Encodings.....33
9.2. Encoding Group.....34
9.3. Associating Captures with Encoding Groups.....35
10. Consumer's Choice of Streams to Receive from the Provider....36
10.1. Local preference.....39
10.2. Physical simultaneity restrictions.....39
10.3. Encoding and encoding group limits.....39
11. Extensibility.....40
12. Examples - Using the Framework (Informative).....40
12.1. Provider Behavior.....40
12.1.1. Three screen Endpoint Provider.....41
12.1.2. Encoding Group Example.....47
12.1.3. The MCU Case.....48
12.2. Media Consumer Behavior.....49

12.2.1. One screen Media Consumer.....50
12.2.2. Two screen Media Consumer configuring the example..50
12.2.3. Three screen Media Consumer configuring the example51
12.3. Multipoint Conference utilizing Multiple Content Captures51
12.3.1. Single Media Captures and MCC in the same Advertisement.....51
12.3.2. Several MCCs in the same Advertisement.....54
12.3.3. Heterogeneous conference with switching and composition.....56
13. Acknowledgements.....63
14. IANA Considerations.....63
15. Security Considerations.....63
16. Changes Since Last Version.....65
17. Authors' Addresses.....70

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 draft-ietf-clue-telepresence-use-cases and to meet the requirements in draft-ietf-clue-telepresence-requirements.

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, the formatting of the media streams it can send, 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 example cases 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 containing 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: 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 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

Model: a set of assumptions a telepresence system of a given vendor adheres to and expects the remote telepresence system(s) also to adhere to.

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 a 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, i.e. audio capture, video capture.

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 is sending the media and a Media Consumer represents the entity that is receiving 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 is no mechanism that can associate these Media Captures with each other in space and time.

(2) The mere fact that there are multiple capture 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, is not supporting 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 formatting of the media streams 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 MUST 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, describes 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 each Capture Scene.

In addition, the Provider can send the Consumer a description of the Individual Encodings it can send in terms of the media attributes of the Encodings, in particular, audio and video parameters such as bandwidth, frame rate, macroblocks per second. Note that this is OPTIONAL, and intended to minimize the number of options a later SDP offer-answer would have to include in the SDP in case of complex setups, as should become clearer shortly when discussing an outline of the call flow.

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 and maximum video coding performance measured in macroblocks/second.

The following diagram illustrates the information contained in an Advertisement.

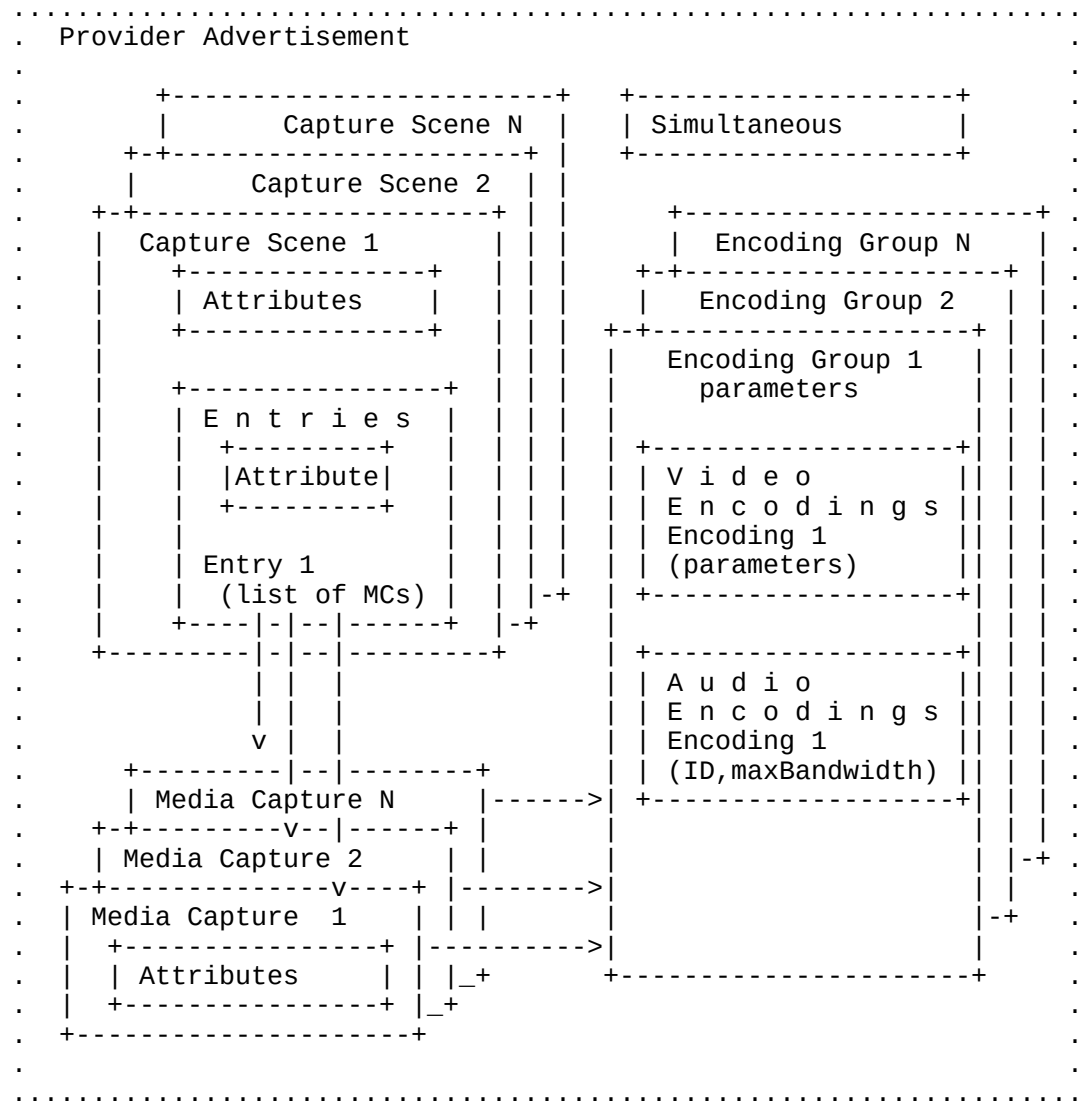


Figure 1: Advertisement Structure

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.

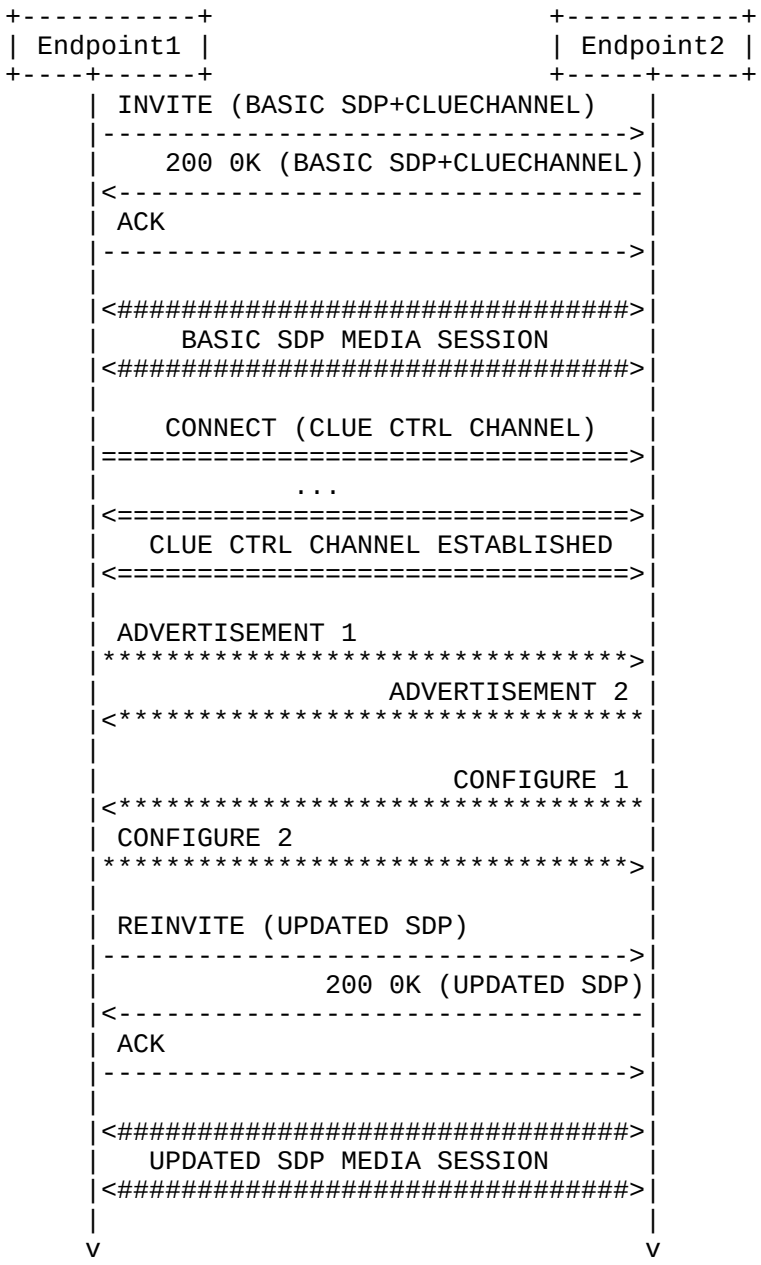


Figure 2: Basic Information Flow

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. Obviously, this is not a particularly sensible design choice in almost all conceivable cases.

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:

- Simple systems which do not have multiple Media Captures to associate spatially need not use the coordinate model.

- 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.

- 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:

- It can represent the immediate output of a physical source (e.g. camera, microphone) or 'synthetic' source (e.g. laptop computer, DVD player).
- It can represent the output of an audio mixer or video composer
- It can represent a concept such as 'the loudest speaker'
- 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. 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 in 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 (composed/switched video, 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, participant information and type).
- Control information for use inside the CLUE protocol suite.

The sub-sections below define the Capture attributes.

7.1.1.1. 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).

7.1.1.2. 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. The Media Consumer can use this information to adjust how it renders the received media if it so chooses.

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.

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 participant 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 Channel Format

The Audio Channel Format attribute is a field with enumerated values which describes the method of encoding used for audio. A value of ’mono’ means the Audio Capture has one channel. ’stereo’ means the Audio Capture has two audio channels, left and right.

This attribute applies only to Audio Captures. A single stereo capture is different from two mono captures that have a left-right spatial relationship. A stereo capture maps to a single Capture

Encoding, while each mono audio capture maps to a separate Capture Encoding.

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 of the Capture, which could be in multiple languages.

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 participants

Individual - Captures an individual participant

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. Participant Information

The participant information attribute allows a Provider to provide specific information regarding the conference participants in a Capture. 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 participants 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. Participant Type

The participant type attribute indicates the type of participant/s contained in the capture in the conference with respect to the

meeting agenda. As a capture may include multiple participants the attribute may contain multiple value. However values shall not be repeated within the attribute.

An Advertiser associates the participant 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 participant type attribute knows with some certainly that the capture contains that participant type. The capture may contain other participant types but the Advertiser has not been able to determine that this is the case.

The types of Captured participants include:

- Chairman - the participant responsible for running the conference according to the agenda.
- Vice-Chairman - the participant responsible for assisting the chairman in running the meeting.
- Minute Taker - the participant responsible for recording the minutes of the conference 4. Member - the participant has no particular responsibilities with respect to running the meeting.
- Presenter - the participant is scheduled on the agenda to make a presentation in the meeting. Note: This is not related to any "active speaker" functionality.
- Translator - the participant is providing some form of translation or commentary in the meeting.
- Timekeeper - the participant is responsible for maintaining the meeting schedule.

Furthermore the participant 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. 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.

Max Captures 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.

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.

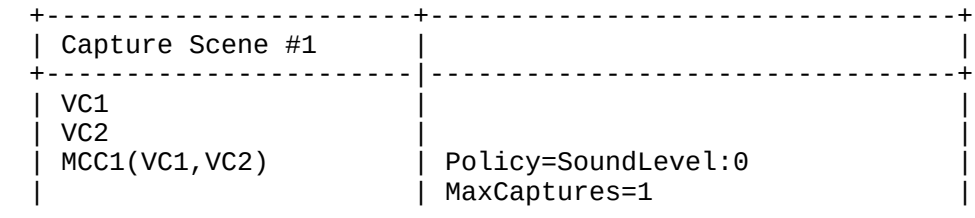
The tokens are:

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.



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	Description=room
CSE(VC1,VC2,VC3)	
CSE(AC1)	
Capture Scene #2	
VC4	Description=Left
VC5	Description=Centre
VC6	Description=Right
AC2	Description=room
CSE(VC4,VC5,VC6)	

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

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. 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 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 of the Capture Scene Entry, which could be in multiple languages;

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 1. 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 provider 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.

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.

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 4.

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.

The parameters of an Individual Encoding represent the maximum values for certain aspects of the encoding. A particular instantiation into a Capture Encoding MAY use lower values than these maximums if that is applicable for the media in question. For example, most video codec specifications require a conformant decoder to decode resolutions and frame rates smaller than what has

been negotiated as a maximum, so downgrading the CLUE maximum values for macroblocks/second is appropriate. On the other hand, downgrading the sample rate of G.711 audio below 8kHz is not specified in G.711 and therefore not applicable in the sense described here.

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;
- Maxmimum 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.

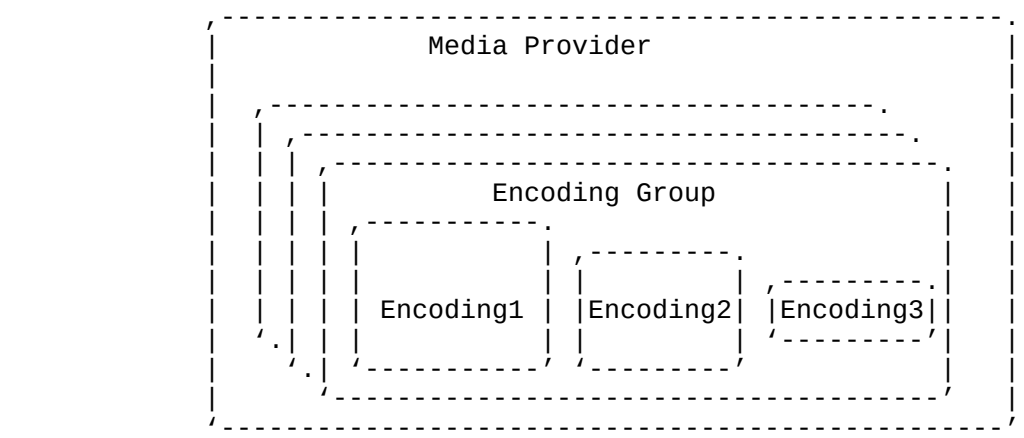


Figure 3: Encoding Group Structure

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 ofCaptures 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, one Individual Encoding, and includes the encoding parameter values. 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, parameters such as the resolution, frame rate, bandwidth, etc. for each Capture Encodings for its chosen Captures. Upon receipt of this Configure from the Consumer, common knowledge is established between Provider and Consumer regarding sensible choices for the media streams and their parameters. 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:

- 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
- 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
- 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. Telepresence is a relatively new industry and while we can foresee certain directions, we also do not know everything about how it will develop. 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:

- Adding more types of media, such as telemetry, can done by defining additional types of Captures in addition to audio and video.
- Adding new functionalities, such as 3-D, say, may require additional attributes describing the Captures.
- Adding a new codecs, such as H.265, can be accomplished by defining new encoding variables.

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.

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

- Each camera can provide one Capture for each 1/3 section of the table
- 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)
- 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)
- 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:

- VC0- (the camera-left camera stream), encoding group=EG0, switched=false, view=table
- VC1- (the center camera stream), encoding group=EG1, switched=false, view=table
- VC2- (the camera-right camera stream), encoding group=EG2, switched=false, view=table
- VC3- (the loudest panel stream), encoding group=EG1, switched=true, view=table
- VC4- (the loudest panel stream with PiPs), encoding group=EG1, composed=true, switched=true, view=room
- VC5- (the zoomed out view of all people in the room), encoding group=EG1, composed=false, switched=false, view=room
- VC6- (presentation stream), encoding group=EG1, presentation, switched=false

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.

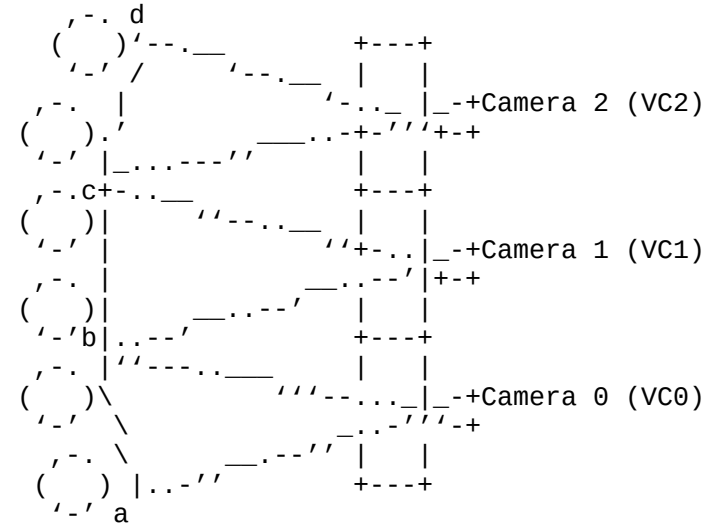


Figure 4: Room Layout

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)
VC3	(-2011,2850,0)	(2011,2850,0)	(-2011,2850,757)	(2011,3000,757)
VC4	(-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)
VC3 none
VC4 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, the "composed" boolean attribute doesn't say anything about how a capture is composed, so the media consumer can't tell based on this attribute that VC4 is composed of a "loudest panel with PiPs".

Audio Captures:

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

Areas of capture:

	bottom left	bottom right	top left	top right
AC0	(-2011,2850,0)	(-673,3000,0)	(-2011,2850,757)	(-673,3000,757)
AC1	(673,3000,0)	(2011,2850,0)	(673,3000,757)	(2011,3000,757)
AC2	(-673,3000,0)	(673,3000,0)	(-673,3000,757)	(673,3000,757)
AC3	(-2011,2850,0)	(2011,2850,0)	(-2011,2850,757)	(2011,3000,757)
AC4	none			

The physical simultaneity information is:

Simultaneous transmission set #1 {VC0, VC1, VC2, VC3, VC4, 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 VC3 or VC4. Also, using every member in the set simultaneously may not make sense - for example VC3(loudest) and VC4 (loudest with PIP). (In addition, there are encoding constraints that make choosing all of the VCs in a set impossible. VC1, VC3, VC4, 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 VC3 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
VC3
VC4
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 VC3, which automatically shows the person who is talking. Similarly for the VC4 and VC5 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.

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).

12.1.2. 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

VC3, VC4 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 (VC3) or a switched view with PiPs (VC4). 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 (VC3, VC4 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.

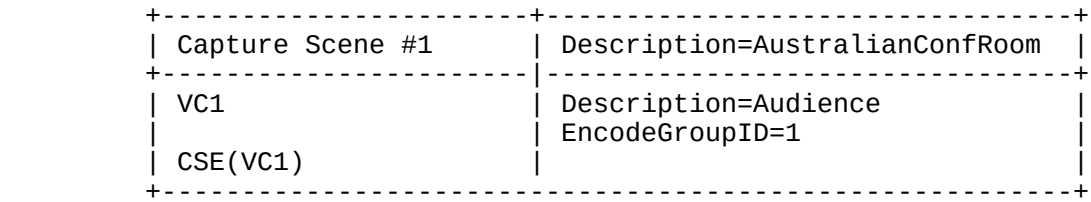


Table 10: Advertisement received from Endpoint A

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

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

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) MCC2(VC2)	Policy=RoundRobin:1 MaxCaptures=1 EncodingGroup=1 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
	EncodingGroup=1
VC2	CaptureArea=Centre
	EncodingGroup=1
VC3	CaptureArea=Right
	EncodingGroup=1
CSE(VC1,VC2,VC3)	

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

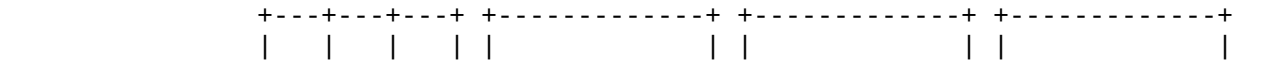
Endpoint F - 2 screens, 1 cameras

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.



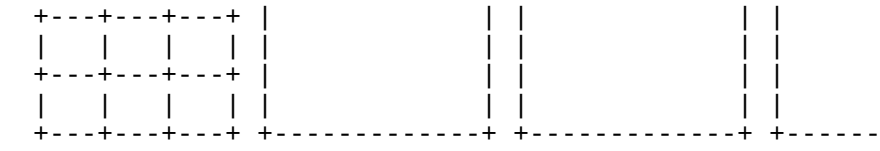


Figure 7: Endpoint A - 4 Screen Display

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.

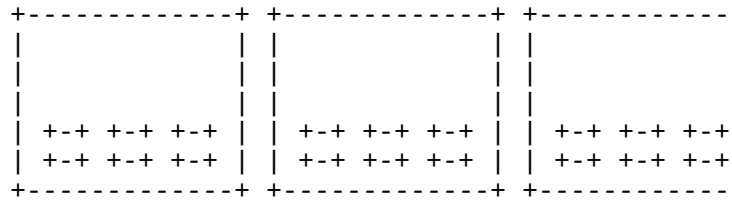


Figure 8: Endpoint B - 3 Screen Display with PiPs

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 be 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	Left
VC11	Center
VC12	Right
AC3	
CSE(VC10,VC11,VC12)	
CSE(AC3)	
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
MCC4(VC4,VC5,VC6,VC7, VC8,VC9,VC10,VC11, VC12,VC13,VC14,VC15)	MaxCaptures=1 Policy=SoundLevel:0 EncodingGroup=1
MCC5(VC4,VC5,VC6,VC7, VC8,VC9,VC10,VC11, VC12,VC13,VC14,VC15)	MaxCaptures=1 Policy=SoundLevel:1 EncodingGroup=1
to	to
MCC12(VC4,VC5,VC6,VC7, VC8,VC9,VC10,VC11, VC12,VC13,VC14,VC15)	MaxCaptures=1 Policy=SoundLevel:8 EncodingGroup=1
CSE(MCC4,MCC5,MCC6, MCC7,MCC8,MCC9,	

MCC10,MCC11,MCC12)	

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(MCC4,MCC4,MCC6, MCC7,MCC8,MCC9, MCC10,MCC11,MCC12) CSE(MCC13)	MaxCaptures=9 EncodingGroup=1

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

As MaxCaptures is 9 it indicates that the capture encoding contains information from up to 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.

13. Acknowledgements

Allyn Romanow and Brian Baldino were authors of early versions. Mark Gorzyinski contributed much to the approach. We want to thank Stephen Botzko for helpful discussions on audio.

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 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.

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 WG
Internet Draft
Intended status: Standards Track
Expires: July 8, 2016

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

January 8, 2016

Framework for Telepresence Multi-Streams
draft-ietf-clue-framework-25.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 and SDP negotiation 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 July 8, 2016.

Copyright Notice

Copyright (c) 2016 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 and Motivation....7
5. Description of the Framework/Model....10
6. Spatial Relationships....15
7. Media Captures and Capture Scenes....17
7.1. Media Captures....17
7.1.1. Media Capture Attributes....18
7.2. Multiple Content Capture....24
7.2.1. MCC Attributes....25
7.3. Capture Scene....30
7.3.1. Capture Scene attributes....33
7.3.2. Capture Scene View attributes....33
7.4. Global View List....34
8. Simultaneous Transmission Set Constraints....35
9. Encodings....37
9.1. Individual Encodings....37
9.2. Encoding Group....38
9.3. Associating Captures with Encoding Groups....39
10. Consumer's Choice of Streams to Receive from the Provider....40
10.1. Local preference....43
10.2. Physical simultaneity restrictions....43
10.3. Encoding and encoding group limits....43
11. Extensibility....44
12. Examples - Using the Framework (Informative)....44
12.1. Provider Behavior....44
12.1.1. Three screen Endpoint Provider....44
12.1.2. Encoding Group Example....51
12.1.3. The MCU Case....52

12.2. Media Consumer Behavior.....53
12.2.1. One screen Media Consumer.....53
12.2.2. Two screen Media Consumer configuring the example..54
12.2.3. Three screen Media Consumer configuring the example55
12.3. Multipoint Conference utilizing Multiple Content Captures55
12.3.1. Single Media Captures and MCC in the same Advertisement.....55
12.3.2. Several MCCs in the same Advertisement.....59
12.3.3. Heterogeneous conference with switching and composition.....60
12.3.4. Heterogeneous conference with voice activated switching.....67
13. Acknowledgements.....70
14. IANA Considerations.....70
15. Security Considerations.....70
16. Changes Since Last Version.....73
17. Normative References.....81
18. Informative References.....82
19. Authors' Addresses.....83

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 multiple audio and video streams 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]. This includes cases using multiple media streams that are not necessarily telepresence.

This document occasionally refers to the term "CLUE", in capital letters. CLUE is an acronym for "ControLling mUltiple streams for tElepresence", which is the name of the IETF working group in which this document and certain companion documents have been developed. Often, CLUE-something refers to something that has been designed by the CLUE working group; for example, this document may be called the CLUE-framework.

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. Companion documents describe the signaling details [I-D.ietf-clue-signaling] and the data model [I-D.ietf-clue-data-model-schema] and protocol [I-D.ietf-clue-protocol].

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. 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.

Capture: Same as Media Capture.

Capture Device: A device that converts physical input, such as audio, video or text, 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 correspond to a real region in physical space, such as a room. A Capture Scene includes attributes and one or more Capture Scene Views, with each view including one or more Media Captures.

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

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

CLUE-enabled call: A call in which two CLUE-capable devices have successfully negotiated support for a CLUE data channel in SDP [RFC4566]. 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.

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: short for Individual Encoding.

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

Endpoint: A CLUE-capable device which is 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.

Global View: A set of references to one or more Capture Scene Views of the same media type that are defined within Scenes of the same advertisement. A Global View is a suggestion from the Provider to the Consumer for one set of CSVs that provide a useful representation of all the scenes in the advertisement.

Global View List: A list of Global Views included in an Advertisement. A Global View List may include Global Views of different media types.

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

Multipoint Control Unit (MCU): a CLUE-capable 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: a CLUE-capable device that intends to receive Capture Encodings.

Media Provider: a CLUE-capable device that intends to send Capture Encodings.

Multiple Content Capture (MCC): A Capture that mixes and/or switches other Captures of a single type. (E.g. all audio or all video.) Particular 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 within a scene 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.

Scene: Same as Capture Scene

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.

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 and Motivation

This section provides an overview of the functional elements defined in this document to represent a telepresence or multistream 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 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 MCUs, 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 MCUs 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-capable device 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 could associate these Media Captures with each other in space and time, in a cross-vendor interoperable way.

(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 are 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 streams 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 streams 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. 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 it 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. The Media Provider and Media Consumer may use information in CLUE messages to reduce the complexity of SIP offer/answer messages. Also, there are aspects of the control of both Endpoints and 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 indicating support for CLUE as specified in this document and its companion documents.

5. Description 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, after CLUE has been enabled in a call, 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 [I-D.ietf-clue-data-model-schema]is based around two main concepts: a Capture and an Encoding. A Media Capture (MC), such as of type 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 [RFC4566].

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.

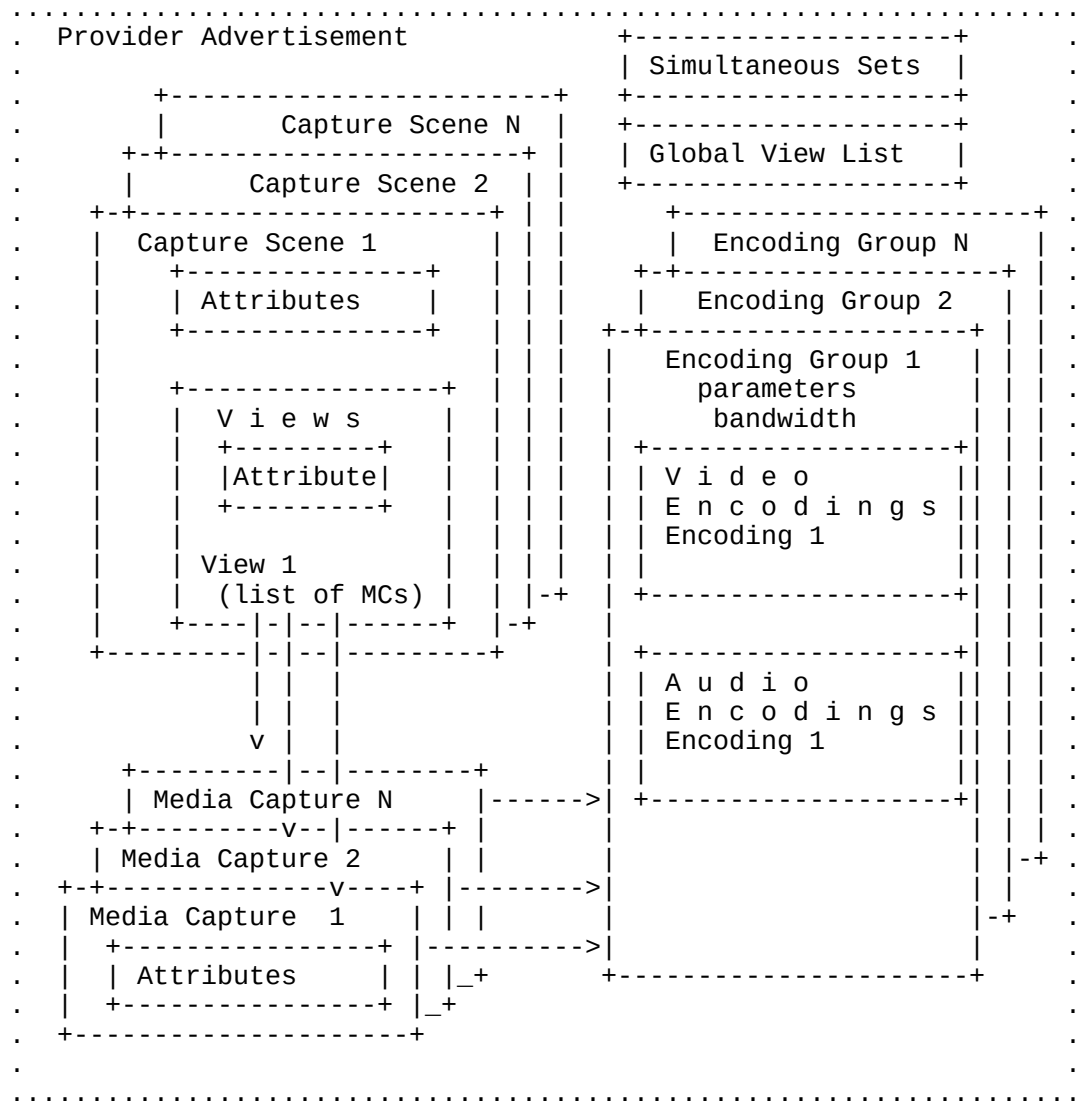


Figure 1: Advertisement Structure

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.

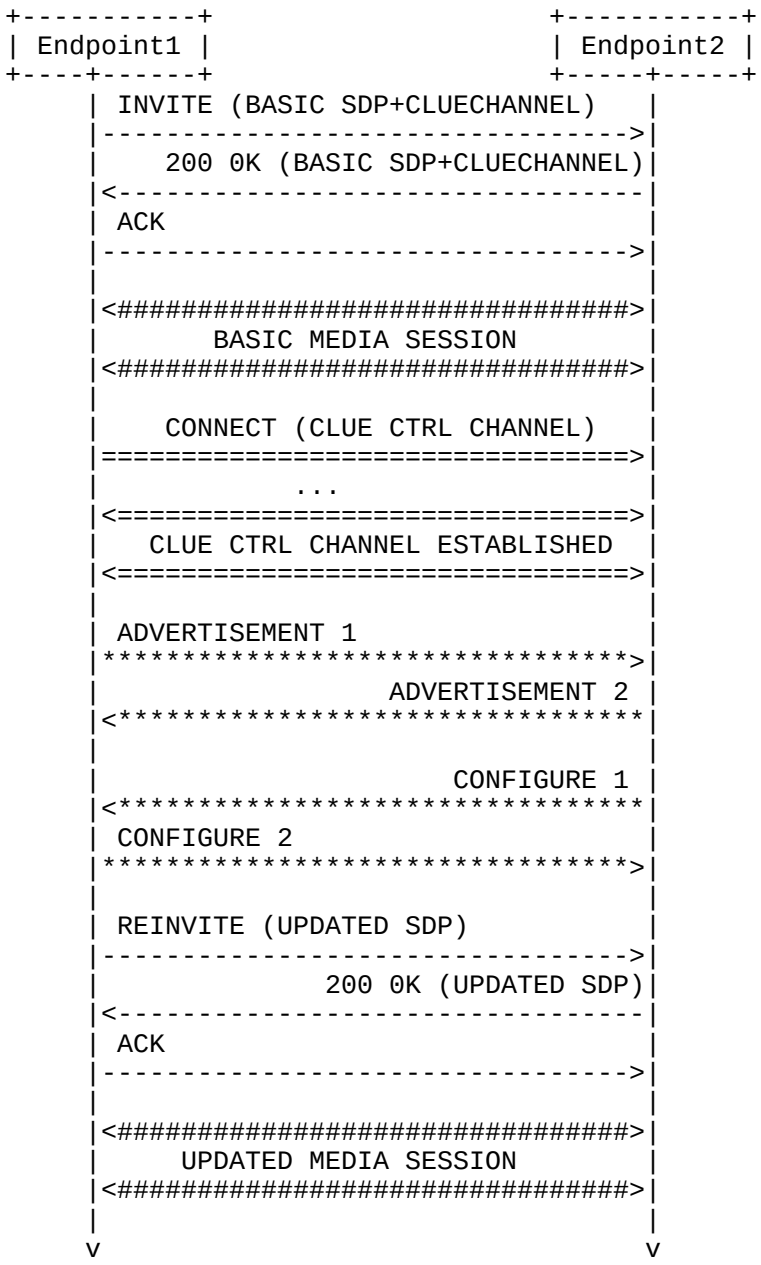


Figure 2: Basic Information Flow

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 several purposes. It uses it, along with information from an offer/answer exchange, to construct a CLUE Configure message to tell the Provider what the Consumer wishes to receive. Also, the Consumer may use the information provided to tailor the SDP it is going to send during any 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. 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. The Consumer can also limit the number of encodings it must set up resources to receive, and not waste resources on unwanted encodings, because it has the Provider’s Advertisement information ahead of time to determine what it really wants to receive. The Consumer can also use the Advertisement information for local rendering decisions.

This initial CLUE exchange is 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 effect of setting up the media transmission itself, involving potentially security exchanges, ICE, and whatnot. This step is plain vanilla SIP.

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, a Configure Message requesting the Provider to place a different Capture source into a Capture Encoding, 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, to the Consumer, information regarding the Provider's capabilities to deliver Media, and attributes related to that Media such as their spatial relationship. 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 the Consumer follows the CLUE protocol and, therefore, gracefully receives and responds to the Provider's information using a Configure operation).

A CLUE-capable device interoperates with a device that does not support CLUE. The CLUE-capable device can determine, by the result of the initial offer/answer exchange, if the other device supports and wishes to use CLUE. The specific mechanism for this is described in [I-D.ietf-clue-signaling]. If the other device does not use CLUE, then the CLUE-capable device falls 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:

- Each Capture Scene has a distinct coordinate system, unrelated to the coordinate systems of other scenes.

- Simple systems which do not have multiple Media Captures to associate spatially need not use the coordinate model, although it can still be useful to provide an Area of Capture.

- 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 provide those real-world measurements to enable the best user experience for advanced receiving systems that can utilize this information. Systems which don't know specific physical dimensions but still know relative distances MUST use 'unknown scale'. 'No scale' is intended to be used only where Media Captures from different devices (with potentially different scales) will be forwarded alongside one another (e.g. in the case of an MCU).

  * "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.

- The coordinate system is right-handed 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 left to right, from the point of view of an observer at the front of the room looking toward the back
Y increases from the front of the room to the back of the room
Z increases from low to high (i.e. floor to ceiling)

Cameras in a scene typically point in the direction of increasing Y, from front to back. But there could be multiple cameras pointing in different directions. If the physical space does not have a well-defined front and back, the provider chooses any direction for X and Y and Z consistent with right-handed coordinates.

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:

- It can represent the immediate output of a physical source (e.g. camera, microphone) or 'synthetic' source (e.g. laptop computer, DVD player)
- It can represent the output of an audio mixer or video composer
- It can represent a concept such as 'the loudest speaker'
- 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 may be associated with one or more Capture Scene Views
. A Media Capture has exactly one set of spatial information
. A Media Capture can be the source of at most one Capture Encoding

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

7.1.1. 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. All these attributes are optional. 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
- Other descriptive information to help the Consumer choose between captures (e.g. description, presentation, view, priority, language, person information and type)

The sub-sections below define the Capture attributes.

7.1.1.1. 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.

7.1.1.2. 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, toward the direction it is pointing; the first point being the Point of Capture (see above).

Together, the Point of Capture and Point on Line of Capture define the direction and 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". This attribute applies only to video captures, not other types of media. By comparing the Area of Capture for different Video Captures within the same Capture Scene a Consumer can determine the spatial relationships between them and render them correctly.

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 Video Capture might be spatially related to other Captures in the same Scene, but there is no detailed information on the relationship.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 CLUE session. 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. Highly 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 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. Description

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

7.1.1.7. 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. Refer to [I-D.ietf-clue-data-model-schema] for how to extend the model.

7.1.1.8. 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. Possible values are:

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

7.1.1.9. 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.

The Language attribute is coded per [RFC5646].

7.1.1.10. 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.

7.1.1.11. Person Type

The Person Type attribute indicates the type of people contained in the capture 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 MUST 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 certainly 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:

- Chair - the person responsible for running the meeting according to the agenda.
- Vice-Chair - the person responsible for assisting the chair in running the meeting.
- Minute Taker - the person responsible for recording the minutes of the meeting.
- Attendee - the person has no particular responsibilities with respect to running the meeting.
- Observer - an Attendee without the right to influence the discussion.
- 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 types.

7.1.1.12. 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 importance of the Capture can be assumed.

7.1.1.13. 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.

7.1.1.14. 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 multiplexed (temporally and/or spatially) or mixed 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. MCCs either contain the referenced Captures and no others, or have no referenced captures and therefore may contain any Capture.

One or more MCCs may also be specified in a CSV. 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	{MC attributes}
VC2	{MC attributes}
VC3	{MC attributes}
MCC1(VC1,VC2,VC3)	{MC and MCC attributes}
CSV(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. MCC Attributes

Media Capture 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)
CSV(MCC1)	

Table 2: Example of MCC and Single Media Capture attributes

The sub-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 or equal to 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 <= 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 media capture 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 an index representing an instance of the policy. The same index value can be used for multiple MCCs.

The tokens are:

SoundLevel - This indicates that the content of the MCC is determined by a sound level detection algorithm. The loudest (active) speaker (or a previous speaker, depending on the index value) 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. An 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
CSV(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 same 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 into a combined "source" for the purpose of switching them together to keep 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	Description=Room
CSV(VC1,VC2,VC3)	
CSV(AC1)	
Capture Scene #2	
VC4	Description=Left
VC5	Description=Centre
VC6	Description=Right
AC2	Description=Room
CSV(VC4,VC5,VC6)	
CSV(AC2)	
Capture Scene #3	
VC7	
AC3	
Capture Scene #4	
VC8	
AC4	
Capture Scene #5	
MCC1(VC1,VC4,VC7)	SynchronisationID=1
	MaxCaptures=1
MCC2(VC2,VC5,VC8)	SynchronisationID=1
	MaxCaptures=1
MCC3(VC3,VC6)	MaxCaptures=1
MCC4(AC1,AC2,AC3,AC4)	SynchronisationID=1
	MaxCaptures=1
CSV(MCC1,MCC2,MCC3)	
CSV(MCC4)	

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.

7.2.1.4. Allow Subset Choice

The Allow Subset Choice MCC attribute is a boolean value, indicating whether or not the Provider allows the Consumer to choose a specific subset of the Captures referenced by the MCC. If this attribute is true, and the MCC references other Captures, then the Consumer MAY select (in a Configure message) a specific subset of those Captures to be included in the MCC, and the Provider MUST then include only that subset. If this attribute is false, or the MCC does not reference other Captures, then the Consumer MUST NOT select a subset.

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 Views (CSV), with each CSV including one or more Media Captures of the same media type. There can also be Media Captures that are not included in a Capture Scene View. 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 Views. An MCU 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 Views for Video Captures, and several Capture Scene Views for Audio Captures. A particular Capture MAY be included in more than one Capture Scene View.

A Provider MAY express spatial relationships between Captures that are included in the same Capture Scene. However, there is no spatial relationship between Media Captures from different Capture Scenes. In other words, Capture Scenes each 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 Views in a Capture Scene are different alternatives the Provider is suggesting for representing the Capture Scene. Each Capture Scene View is given an advertisement unique identity. The order of Capture Scene Views within a Capture Scene has no significance. The Media Consumer can choose to receive all Media Captures from one Capture Scene View 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 Views. Different Capture Scene Views of the same media type are not necessarily mutually exclusive alternatives. Also note that the presence of multiple Capture Scene Views (with potentially multiple encoding options in each view) 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 View MUST be of the same media type - it is not possible to mix audio and video captures in

the same Capture Scene View, for instance. The Provider MUST be capable of encoding and sending all Captures (that have an encoding group) in a single Capture Scene View simultaneously. The order of Captures within a Capture Scene View has no significance. A Consumer can decide to receive all the Captures in a single Capture Scene View, 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 Views, all subject to the constraints set by Simultaneous Transmission Sets, as discussed in section 8.

When a Provider advertises a Capture Scene with multiple CSVs, it is essentially signaling that there are multiple representations of the same Capture Scene available. In some cases, these multiple views would be used simultaneously (for instance a "video view" and an "audio view"). In some cases the views would conceptually be alternatives (for instance a view consisting of three Video Captures covering the whole room versus a view 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 View that most closely matched the Consumer’s number of display devices or screen layout.

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

1. (VC0, VC1, VC2) - left, center and right camera Video Captures

2. (MCC3) - 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 view 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 view (MCC3) and the third view (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 7.1.1.10. 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 View attributes

A Capture Scene can include one or more Capture Scene Views in addition to the Capture Scene wide attributes described above. Capture Scene View attributes apply to the Capture Scene View as a

whole, i.e. to all Captures that are part of the Capture Scene View.

Capture Scene View attributes include:

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

7.4. Global View List

An Advertisement can include an optional Global View list. Each item in this list is a Global View. The Provider can include multiple Global Views, 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 View 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 CSV represents a particular scene.

As an example, suppose an advertisement has three scenes, and each scene has three CSVs, ranging from one to three video captures in each CSV. The Provider is advertising a total of nine video Captures across three scenes. The Provider can use the Global View 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 Global View containing just a single CSV with three Captures and nothing from the other two scenes. Or a Provider might suggest a Global View containing three different CSVs, one from each scene, with a single video Capture in each.

Some additional rules:

- The ordering of Global Views in the Global View list is insignificant.
- The ordering of CSVs within each Global View is insignificant.
- A particular CSV may be used in multiple Global Views.
- The Provider must be capable of encoding and sending all Captures within the CSVs of a given Global View simultaneously.

The following figure shows an example of the structure of Global Views in a Global View List.

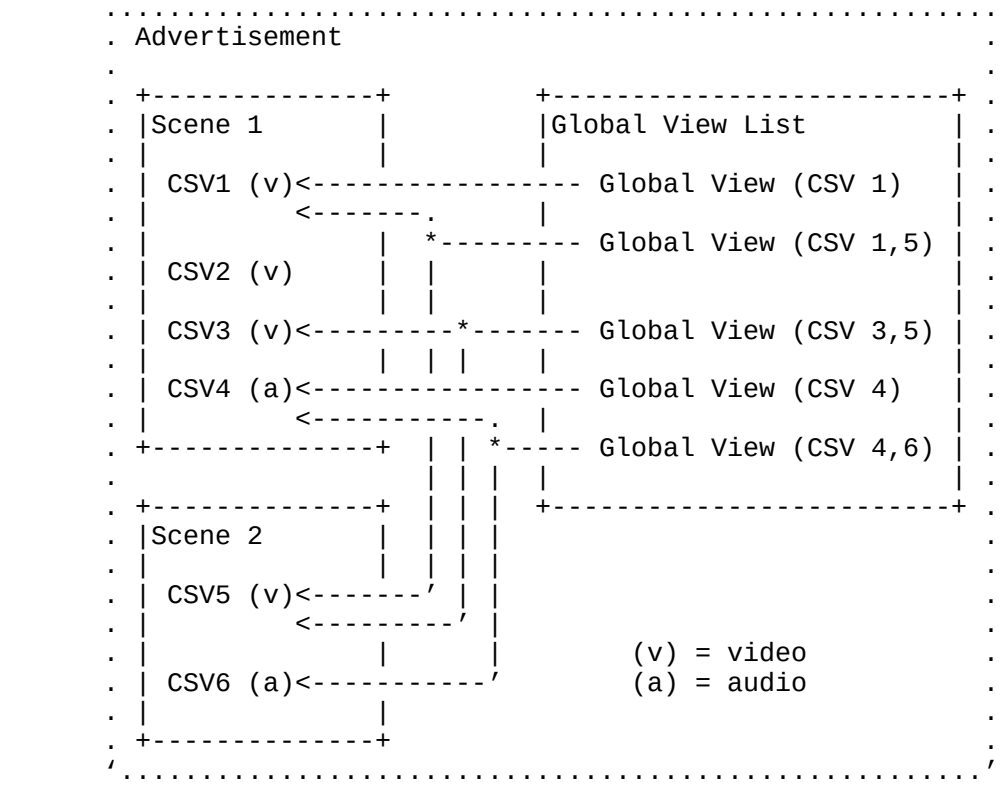


Figure 3: Global View List Structure

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 Individual Encodings and 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 Transmission 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 Transmission Sets in its Advertisement. These 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 View to be used simultaneously. Similarly, the Simultaneous Transmission Sets MUST reflect the simultaneity expressed by any Global View.

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

If an Advertisement does not include Simultaneous Transmission Sets, then the Provider MUST be able to simultaneously provide all the Captures from any one CSV of each media type from each Capture Scene. Likewise, if there are no Simultaneous Transmission Sets and there is a Global View list, then the Provider MUST be able to simultaneously provide all the Captures from any particular Global View (of each media type) from the Global View list.

If an Advertisement includes multiple Capture Scene Views in a Capture Scene then the Consumer MAY choose one Capture Scene View 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 the 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 as 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. A single Encoding Group MAY refer to Encodings for different media types.

The Encoding Group data structure contains:

- Maximum bitrate for all encodings in the group combined;
- A list of identifiers for the Individual Encodings 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 this group value.

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

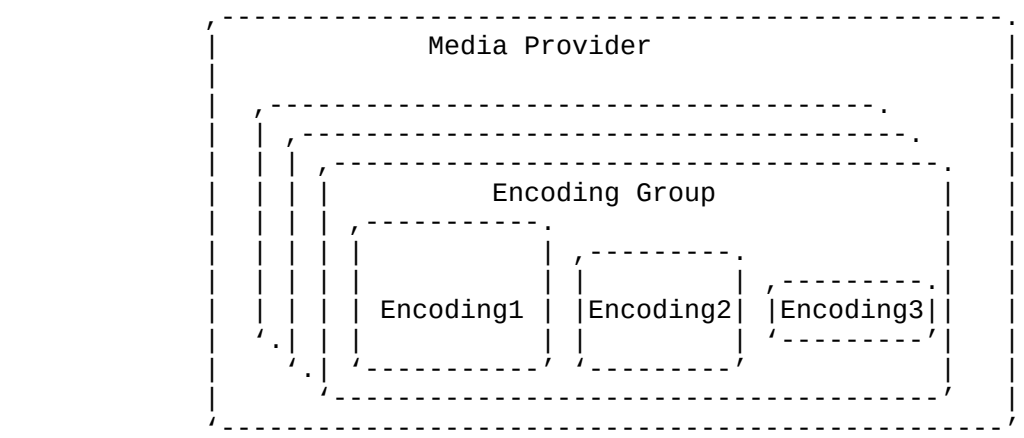


Figure 4: Encoding Group Structure

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, including MCCs, MAY be associated with one Encoding Group. To be eligible for configuration, a Media Capture MUST be associated with one Encoding Group, which is used to instantiate that Capture into a Capture Encoding. When an MCC is configured all the Media Captures referenced by the MCC will appear in the Capture Encoding according to the attributes of the chosen encoding of the MCC. This allows an Advertiser to specify encoding 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
CSV(VC1)	
CSV(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 Capture Encodings 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.

It is a protocol conformance requirement that the Encoding Groups MUST allow all the Captures in a particular Capture Scene View 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 an 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 (or any Captures, if none are referenced) 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.

- If the MCC attribute Allow Subset Choice is true, then the Consumer may or may not choose to receive all the indicated Captures. 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 MCC 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, or the Allow Subset Choice attribute is false, 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.

For each Capture the Consumer wants to receive, it configures one 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, and an SDP offer defining the Individual Encodings, 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:

- 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
- if the Consumer is an MCU, it may choose to receive loudest speaker streams (in order to perform its own media composition) and avoid pre-composed video Captures
- 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, 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 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:

- Adding more types of media, such as telemetry, can done by defining additional types of Captures in addition to audio and video.
- Adding new functionalities, such as 3-D video Captures, 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

- Each camera can provide one Capture for each 1/3 section of the table

- 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)

- A single Capture representing the active speaker with the other 2 Captures shown picture in picture (PiP) within the stream can be provided (again, implemented inside the endpoint)

- A Capture showing a zoomed out view of all 6 seats in the room can be provided

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

Video Captures:

- VC0- (the left camera stream), encoding group=EG0, view=table

- VC1- (the center camera stream), encoding group=EG1, view=table

- VC2- (the right camera stream), encoding group=EG2, view=table

- MCC3- (the loudest panel stream), encoding group=EG1, view=table, MaxCaptures=1, policy=SoundLevel

- MCC4- (the loudest panel stream with PiPs), encoding group=EG1, view=room, MaxCaptures=3, policy=SoundLevel

- VC5- (the zoomed out view of all people in the room), encoding group=EG1, view=room

- 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 captures 2 people. The six seats are not all in a straight line.

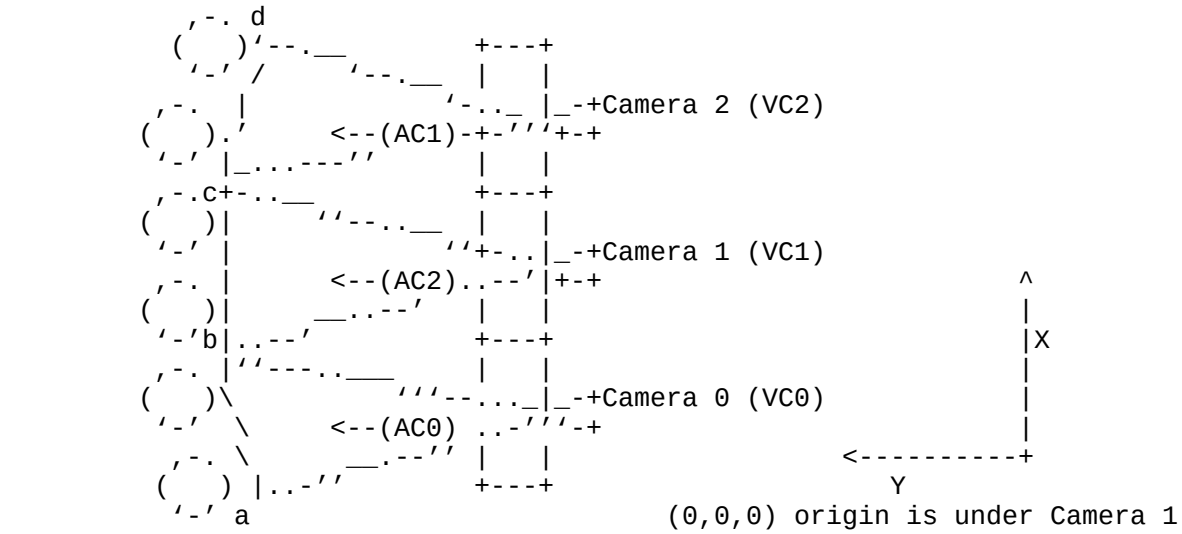


Figure 5: Room Layout Top View

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 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:

Three ceiling microphones are located between the cameras and the table, at the same height as the cameras. The microphones point down at an angle toward the seating positions.

- AC0 (left), encoding group=EG3
- AC1 (right), encoding group=EG3
- AC2 (center) encoding group=EG3
- AC3 being a simple pre-mixed audio stream from the room (mono), encoding group=EG3
- 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 view in the Capture Scene. For example VC1 and MCC3 at the same time. The information below about Encodings is a summary of what would be conveyed in SDP, not directly in the CLUE Advertisement.

```
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 6: 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 7: 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 Views 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 View

Capture Scene #1
VC0, VC1, VC2
MCC3
MCC4
VC5
AC0, AC1, AC2
AC3

Capture Scene #2
VC6
AC4

Table 7: Example Capture Scene Views

Different Capture Scenes are distinct from each other, and are non-overlapping. A Consumer can choose a view 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 VC5 alternatives.

As in the video case, the different views 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 CSV it is capable of receiving.

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

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

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

12.1.2. Encoding Group Example

This is an example of an Encoding Group to illustrate how it can express dependencies between Encodings. The information below about Encodings is a summary of what would be conveyed in SDP, not directly in the CLUE Advertisement.

```
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. Each MCC is for video. A single Audio Capture is provided for all single and multi-screen configurations that can be associated (e.g. lip-synced) with any combination of Video Captures (the MCCs) at the consumer.

Capture Scene #1	
MCC	for a single screen consumer
MCC1, MCC2	for a two screen consumer
MCC3, MCC4, MCC5	for a three screen consumer
MCC6, MCC7, MCC8, MCC9	for a four screen consumer
AC0	AC representing all participants
CSV(MCC0)	
CSV(MCC1,MCC2)	
CSV(MCC3,MCC4,MCC5)	
CSV(MCC6,MCC7, MCC8,MCC9)	
CSV(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
CSV(VC10)	
CSV(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 View 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 views 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 views by themselves, not grouped together in a single view, 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 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 CSV(VC1)	Description=Audience EncodeGroupID=1

Table 10: Advertisement received from Endpoint A

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

Table 11: Advertisement received from Endpoint B

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

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 CSV in Capture Scene #4 as per the example below.

Advertisement sent to Endpoint F

Capture Scene #1	Description=AustralianConfRoom
VC1 CSV(VC1)	Description=Audience
Capture Scene #2	**Description=ChinaConfRoom**
VC2 VC3 CSV(VC2, VC3)	Description=Speaker EncodingGroup=1 Description=Audience
Capture Scene #3	**Description=USAConfRoom**
VC4 CSV(VC4)	Description=Audience
Capture Scene #4	
MCC1(VC1,VC3,VC4) MCC2(VC2) CSV2(MCC1,MCC2)	Policy=RoundRobin:1 MaxCaptures=1 EncodingGroup=1 AllowSubset=True MaxCaptures=1 EncodingGroup=1

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

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
CSV(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	
CSV(VC1,VC2,VC3)	
Capture Scene #2	**Description=ChinaConfRoom**
VC4	
VC5	
VC6	
CSV(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
CSV(MCC1,MCC2,MCC3)	

Table 18: Advertisement sent to Endpoint F

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.

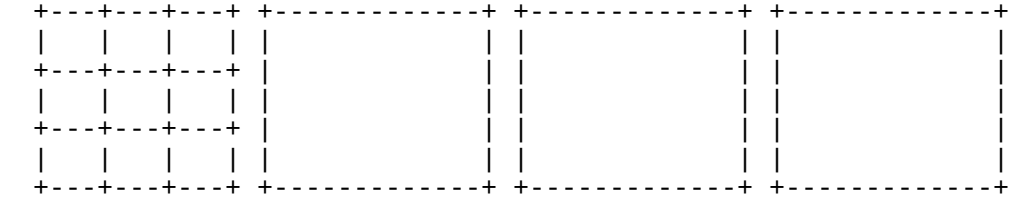


Figure 8: Endpoint A - 4 Screen Display

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.

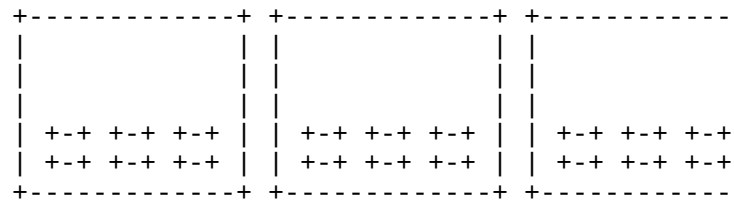


Figure 9: Endpoint B - 3 Screen Display with PiPs

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 be 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
CSV1(VC1, VC2, VC3)	
CSV2(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
CSV1(VC1)	
CSV2(AC1)	

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

Rather than considering what is displayed 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	CaptureArea=Left
VC5	CaptureArea=Center
VC6	CaptureArea=Right
AC1	
CSV(VC4,VC5,VC6)	
CSV(AC1)	
Capture Scene #2	**Description=Endpoint C**
VC7	CaptureArea=Left
VC8	CaptureArea=Center
VC9	CaptureArea=Right
AC2	
CSV(VC7,VC8,VC9)	
CSV(AC2)	
Capture Scene #3	**Description=Endpoint D**

VC10 VC11 VC12 AC3 CSV(VC10,VC11,VC12) CSV(AC3)	CaptureArea=Left CaptureArea=Center CaptureArea=Right
Capture Scene #4	Description=Endpoint E
VC13 AC4 CSV(VC13) CSV(AC4)	
Capture Scene #5	Description=Endpoint F
VC14 AC5 CSV(VC14) CSV(AC5)	
Capture Scene #6	Description=Endpoint G
VC15 AC6 CSV(VC15) CSV(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
CSV(MCC1,MCC2,MCC3)		
CSV(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
CSV(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) CSV(MCC13)	MaxCaptures=9 EncodingGroup=1

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. The Provider can use the RTCP CaptureId SDES item and associated RTP header extension, as described in [I-D.ietf-clue-rtp-mapping], to convey this information to the Consumer.

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
	EncodingGroup=1
VC2	CaptureArea=Center
	EncodingGroup=1
VC3	CaptureArea=Right
	EncodingGroup=1
MCC1(VC1,VC2,VC3)	MaxCaptures=1
	CaptureArea=whole scene
	Policy=SoundLevel:0
	EncodingGroup=1
AC1	CaptureArea=whole scene
	EncodingGroup=2
CSV1(VC1, VC2, VC3)	
CSV2(MCC1)	
CSV3(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
CSV1(MCC1,MCC2,MCC3 CSV2(MCC4) CSV3(MCC5,MCC6)	

Table 26: Advertisement sent from the MCU to C

The MCU advertises one scene, with four video MCCs. Three of them in CSV1 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 CSV3 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.

An MCU 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 [RFC5239]. 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 an MCU 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 conferencing protocol, thus the security considerations in [RFC4579] MUST be followed. Other security issues related to MCUs are discussed in the XCON framework [RFC5239]. The use of xCard with potentially sensitive information provides another reason to implement recommendations of section 11/[RFC5239].

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 MCUs. 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 [RFC3261], including user authentication and authorization, MUST be supported. In addition, the media MUST be secured. DTLS/SRTP MUST be supported and SHOULD be used unless the media, which is based on RTP, is secured by other means (see [RFC7201] [RFC7202]). Media security is also discussed in [I-D.ietf-clue-signaling] and [I-D.ietf-clue-rtp-mapping]. Note that SIP call setup is done before any CLUE specific information is available so the authentication and authorization are based on the SIP mechanisms. The entity that will be authenticated may use the Endpoint identity or the endpoint user identity; this is an application issue and not a CLUE specific issue.

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. The CLUE data model [I-D.ietf-clue-data-model-schema] defines through an XML schema the syntax to be used. Some of the information which could possibly introduce privacy concerns is the xCard information as described in section 7.1.1.10. The decision about which xCard information to send in the CLUE channel is an application policy for point to point and multipoint calls based on the authenticated identity that can be the endpoint identity or the user of the endpoint. For example the telepresence multipoint application can authenticate a user before starting a CLUE exchange with the telepresence system and have a policy per user.

In addition, the (text) description field in the Media Capture attribute (section 7.1.1.6) 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 View (7.3.2) attributes. An implementation SHOULD give users control over what sensitive information is sent in an Advertisement. One other important consideration for the information in the xCard as well as the description field in the Media Capture and Capture Scene View 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 MUST NOT be used to make any authorization decisions.

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 [I-D.ietf-clue-datachannel].

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 24 to 25:

Updates from IESG review.

1. A few clarifications in various places.
2. Change references to RFC5239 and RFC5646 from informative to normative.

Changes from 23 to 24:

1. Updates to Security Considerations section.
2. Update version number of references to other CLUE documents in progress.

Changes from 22 to 23:

1. Updates to Security Considerations section.
2. Update version number of references to other CLUE documents in progress.
3. Change some "MAY" to "may".
4. Fix a few grammatical errors.

Changes from 21 to 22:

1. Add missing references.
2. Update version number of referenced working group drafts.
3. Minor updates for idnits issues.

Changes from 20 to 21:

1. Clarify CLUE can be useful for multi-stream non-telepresence cases.
2. Remove unnecessary ambiguous sentence about optional use of CLUE protocol.

3. Clarify meaning if Area of Capture is not specified.
4. Remove use of "conference" where it didn't fit according to the definition. Use "CLUE session" or "meeting" instead.
5. Embedded Text Attribute: Remove restriction it is for video only.
6. Minor cleanup in section 12 examples.
7. Minor editorial corrections suggested by Christian Groves.

Changes from 19 to 20:

1. Define term "CLUE" in introduction.
2. Add MCC attribute Allow Subset Choice.
3. Remove phrase about reducing SDP size, replace with potentially saving consumer resources.
4. Change example of a CLUE exchange that does not require SDP exchange.
5. Language attribute uses RFC5646.
6. Change Member person type to Attendee. Add Observer type.
7. Clarify DTLS/SRTP MUST be supported.
8. Change SHOULD NOT to MUST NOT regarding using xCard or description information for authorization decisions.
9. Clarify definition of Global View.
10. Refer to signaling doc regarding interoperating with a device that does not support CLUE.
11. Various minor editorial changes from working group last call feedback.
12. Capitalize defined terms.

Changes from 18 to 19:

1. Remove the Max Capture Encodings media capture attribute.
2. Refer to RTP mapping document in the MCC example section.
3. Update references to current versions of drafts in progress.

Changes from 17 to 18:

1. Add separate definition of Global View List.
2. Add diagram for Global View List structure.
3. Tweak definitions of Media Consumer and Provider.

Changes from 16 to 17:

1. Ticket #59 - rename Capture Scene Entry (CSE) to Capture Scene View (CSV)

2. Ticket #60 - rename Global CSE List to Global View List

3. Ticket #61 - Proposal for describing the coordinate system. Describe it better, without conflicts if cameras point in different directions.

4. Minor clarifications and improved wording for Synchronisation Identity, MCC, Simultaneous Transmission Set.

5. Add definitions for CLUE-capable device and CLUE-enabled call, taken from the signaling draft.

6. Update definitions of Capture Device, Media Consumer, Media Provider, Endpoint, MCU, MCC.

7. Replace "middle box" with "MCU".

8. Explicitly state there can also be Media Captures that are not included in a Capture Scene View.

9. Explicitly state "A single Encoding Group MAY refer to encodings for different media types."

10. In example 12.1.1 add axes and audio captures to the diagram, and describe placement of microphones.

11. Add references to data model and signaling drafts.

12. Split references into Normative and Informative sections. Add heading number for references section.

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.

17. Normative References

[I-D.ietf-clue-datachannel]
Holmberg, C., "CLUE Protocol Data Channel", draft-ietf-clue-datachannel-11 (work in progress), November 2015.

[I-D.ietf-clue-data-model-schema]
Presta, R., Romano, S P., "An XML Schema for the CLUE data model", draft-ietf-clue-data-model-schema-11 (work in progress), October 2015.

[I-D.ietf-clue-protocol]
Presta, R. and S. Romano, "CLUE protocol", draft-ietf-clue-protocol-06 (work in progress), October 2015.

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

[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.

[RFC4566] Handley, M., Jacobsen, V., Perkins, C., "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4579] Johnston, A., Levin, O., "SIP Call Control - Conferencing for User Agents", RFC 4579, August 2006

[RFC5239] Barnes, M., Boulton, C., Levin, O., "A Framework for Centralized Conferencing", RFC 5239, June 2008.

[RFC5646] Phillips, A., Davis, M., "Tags for Identifying Languages", RFC 5646, September 2009.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, August 2011.

[RFC6351] Perreault, S., "xCard: vCard XML Representation", RFC 6351, August 2011.

18. Informative References

[I-D.ietf-clue-rtp-mapping]
Even, R., Lennox, J., "Mapping RP streams to CLUE media captures", draft-ietf-clue-rtp-mapping-05 (work in progress), October 2015.

[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.

[RFC7201] Westerlund, M., Perkins, C., "Options for Securing RTP Sessions", RFC 7201, April 2014.

[RFC7202] Perkins, C., Westerlund, M., "Why RTP Does Not Mandate a Single Media Security Solution ", RFC 7202, April 2014.

[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.

19. 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 WG
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2014

R. Even
Huawei Technologies
J. Lennox
Vidyo

October 21, 2013

Mapping RTP streams to CLUE media captures
draft-ietf-clue-rtp-mapping-01.txt

Abstract

This document describes mechanisms and recommended practice for mapping RTP media streams defined in SDP to CLUE media captures.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 24, 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 2
2. Terminology 2
3. RTP topologies for CLUE 3
4. Mapping CLUE Capture Encodings to RTP streams 5
4.1. Review of current directions in MMUSIC, AVText and AVTcore 6
4.2. Requirements of a solution 7
4.3. Static Mapping 8
4.4. Dynamic mapping 9
4.5. Recommendations 9
5. Application to CLUE Media Requirements 10
6. Examples 11
6.1. Static mapping 12
6.2. Dynamic Mapping 14
7. Acknowledgements 15
8. IANA Considerations 15
9. Security Considerations 15
10. References 15
10.1. Normative References 15
10.2. Informative References 16
Authors' Addresses 17

1. Introduction

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

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

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

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119[RFC2119] and indicate requirement levels for compliant RTP implementations.

3. RTP topologies for CLUE

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

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

In the Media Mixer topology, the peers communicate only with the mixer. The mixer provides mixed or composited media streams, using its own SSRC for the sent streams. There are two cases here. In the first case the mixer may have separate RTP sessions with each peer (similar to the point to point topology) terminating the RTCP sessions on the mixer; this is known as Topo-RTCP-Terminating MCU in [RFC5117]. In the second case, the mixer can use a conference-wide RTP session similar to RFC 5117’s Topo-mixer or Topo-Video-switching. The major difference is that for the second case, the mixer uses conference-wide RTP sessions, and distributes the RTCP reports to all the RTP session participants, enabling them to learn all the CNAMEs and SSRCs of the participants and know the contributing source or sources (CSRCs) of the original streams from the RTP header. In the first case, the Mixer terminates the RTCP and the participants cannot know all the available sources based on the RTCP information. The

conference roster information including conference participants, endpoints, media and media-id (SSRC) can be available using the conference event package [RFC4575] element.

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

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

The above topologies demonstrate two major RTP/RTCP behaviors:

1. The mixer may either use the source SSRC when forwarding RTP packets, or use its own created SSRC. Still the mixer will distribute all RTCP information to all participants creating conference-wide RTP session/s. This allows the participants to learn the available RTP sources in each RTP session. The original source information will be the SSRC or in the CSRC depending on the topology. The point to point case behaves like this.

2. The mixer terminates the RTCP from the source, creating separate RTP sessions with the peers. In this case the participants will not receive the source SSRC in the CSRC. Since this is usually a mixer topology, the source information is available from the SIP conference event package [RFC4575]. Subscribing to the conference event package allows each participant to know the SSRCs of all sources in the conference.

4. Mapping CLUE Capture Encodings to RTP streams

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

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

Even though Session multiplexing is supported by CLUE, for scaling reasons, CLUE recommends using SSRC multiplexing in a single or multiple sessions. So we need to look at how to map RTP streams to Media Captures Encodings when SSRC multiplexing is used.

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

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

2. The SSRCs are dynamic, representing the original source and are relayed by the Mixer/MCU to the participants.

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

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

4.1. Review of current directions in MMUSIC, AVText and AVTcore

Editor's note: This section provides an overview of the RFCs and drafts that can be used a base for a mapping solution. This section is for information only, and if the WG thinks that it is the right direction, the authors will bring the required work to the relevant WGs.

The solution needs to also support the simulcast case where more than one RTP session may be advertised for a Media Capture. Support of such simulcast is out of scope for CLUE.

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

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

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

[I-D.westerlund-avtcore-max-ssrc] proposes a signaling solution for how to use multiple SSRCs within one RTP session.

[I-D.westerlund-avtext-rtcp-sdes-srcname] provides an extension that may be send in SDP, as an RTCP SDES information or as an RTP header extension that uniquely identifies a single media source. It defines an hierarchical order of the SRCNAME parameter that can be used to for example to describe multiple resolution from the same source (see section 5.1 of [I-D.westerlund-avtcore-rtp-simulcast]). Still all the examples are using RTP session multiplexing.

Other documents reviewed by the authors but are currently not used in a proposed solution include:

[I-D.lennox-mmusic-sdp-source-selection] specifies how participants in a multimedia session can request a specific source from a remote party.

[I-D.westerlund-avtext-codec-operation-point](expired) extends the codec control messages by specifying messages that let participants communicate a set of codec configuration parameters.

Using the above documents it is possible to negotiate the max number of received and sent RTP streams inside an RTP session (m-line or bundled m-line). This allows also offering allowed combinations of codec configurations using different payload type numbers

Examples: max-recv-ssrc:{96:2 & 97:3) where 96 and 96 are different payload type numbers. Or max-send-ssrc{*:4}.

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

4.2. Requirements of a solution

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

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

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

Media-3: It must be possible for a Clue endpoint or MCU to simultaneously send sources corresponding to static, to composited, and to switched captures, in the same transport flow. (Any given device might not necessarily be able send all of these source types; but for those that can, it must be possible for them to be sent simultaneously.)

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

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

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

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

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

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

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

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

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

4.3. Static Mapping

Static mapping is widely used in current MCU implementations. It is also common for a point to point symmetric use case when both endpoints have the same capabilities. For capture encodings with static SSRCs, it is most straightforward to indicate this mapping outside the media stream, in the CLUE or SDP signaling. An SDP source attribute [RFC5576] can be used to associate CLUE encoding using appIds[I-D.even-mmusic-application-token] with SSRCs in SDP. Each SSRC will have an appId value that will be specified also in the CLUE media capture as an attribute. The provider advertisement could, if it wished, use the same SSRC for media capture encodings that are mutually exclusive. (This would be natural, for example, if two advertised captures are implemented as different configurations of the same physical camera, zoomed in or out.). Section 6 provide an example of an SDP offer and CLUE advertisement.

4.4. Dynamic mapping

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

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

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

If the appIds are chosen by the media sender, by offering a particular capture encoding to multiple recipients with the same ID, this requires the sender to only produce one version of the stream (assuming outgoing payload type numbers match). This reduces the cost in the multicast case, although does not necessarily help in the switching case.

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

4.5. Recommendations

The recommendation is that endpoints MUST support both the static declaration of capture encoding SSRCs, and appId in every media packet. For low bandwidth situations, this may be considered excessive overhead; in which case endpoints MAY support the approach where appIds are sent selectively. The SDP offer MAY specify the SSRC mapping to capture encoding. In the case of static mapping topologies there will be no need to use the header extensions in the media, since the SSRC for the RTP stream will remain the same during the call unless a collision is detected and handled according to RFC5576 [RFC5576]. If the used topology uses dynamic mapping then

the appId will be used to indicate the RTP stream switch for the media capture. In this case the SDP description may be used to negotiate the initial SSRC but this will be left for the implementation. Note that if the SSRC is defined explicitly in the SDP the SSRC collision should be handled as in RFC5576.

5. Application to CLUE Media Requirements

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

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

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

This applies naturally for static sources with a Switched Mixer. For dynamic sources with a Source-Projecting middlebox, this just requires the appId in the header extension element to be updated appropriately.

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

For a Switched Mixer, this applies naturally. For a Source-Projecting middlebox, this is done based on the appId.

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

For a Switched Mixer, this is done based on the CSRC, if the mixer is providing CSRCs; if for a Source-Projecting middlebox, this is done based on the SSRC.

Media-8: It must be possible for a source to move among switched captures without requiring a refresh of decoder state (e.g., for

video, a fresh I-frame), when this is unnecessary. However, it must also be possible for a receiver to indicate when a refresh of decoder state is in fact necessary.

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

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

For a Source-Projecting middlebox, this can be accomplished by sending multiple dynamic appIds for the same source; this can also be done for an environment with a hybrid of mixer topologies and static and dynamic captures, described below in Section 6. It is not possible for static captures from a Switched Mixer.

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

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

6. Examples

It is possible for a CLUE device to send multiple instances of the topologies in Section 3 simultaneously. For example, an MCU which uses a traditional audio bridge with switched video would be a Mixer topology for audio, but a Switched Mixer or a Source-Projecting middlebox for video. In the latter case, the audio could be sent as a static source, whereas the video could be dynamic.

More notably, it is possible for an endpoint to send the same sources both for static and dynamic captures. Consider the example [I-D.ietf-clue-framework], where an endpoint can provide both three

cameras (VC0, VC1, and VC2) for left, center, and right views, as well as a switched view (VC3) of the loudest panel.

It is possible for a consumer to request both the (VC0 - VC2) set and VC3. It is worth noting that the content of VC3 is, at all times, exactly the content of one of VC0, VC1, or VC2. Thus, if the sender uses the Source-Projecting middlebox topology for VC3, the consumer that receives these three sources would not need to send any additional media traffic over just sending (VC0 - VC2).

In this case, the advertiser could describe VC0, VC1, and VC2 in its initial advertisement or SDP with static SSRCs, whereas VC3 would need to be dynamic. The role of VC3 would move among VC0, VC1, or VC2, indicated by the appId RTP header extension on those streams' RTP packets.

6.1. Static mapping

Using the video capture example from the framework for a three camera system with four monitors where one is for the presentation stream [I-D.ietf-clue-framework] document:

- VC0- (the camera-left camera stream, purpose=main, switched:no
- VC1- (the center camera stream, purpose=main, switched:no
- VC2- (the camera-right camera stream), purpose=main, switched:no
- VC3- (the loudest panel stream), purpose=main, switched:yes
- VC4- (the loudest panel stream with PiPs), purpose=main, composed=true; switched:yes
- VC5- (the zoomed out view of all people in the room), purpose=main, composed=no; switched:no
- VC6- (presentation stream), purpose=presentation, switched:no

Where the physical simultaneity information is:

{VC0, VC1, VC2, VC3, VC4, VC6}

{VC0, VC2, VC5, VC6}

In this case the provider can send up to six simultaneous streams and receive four one for each monitor. This is the maximum case but it can be further limited by the capture scene entries which may propose sending only three camera streams and one presentation, still since

the consumer can select any media captures that can be sent simultaneously the offer will specify 6 streams where VC5 and VC1 are using the same resource and are mutually exclusive.

In the Advertisement there may be two capture scenes:

The first capture scene may have four entries:

{VC0, VC1, VC2}

{VC3}

{VC4}

{VC5}

The second capture scene will have the following single entry.

{VC6}

We assume that an intermediary will need to look at CLUE if want to have better decision on handling specific RTP streams for example based on them being part of the same capture scene so the SDP will not group streams by capture scene.

The SIP offer may be

m=video 49200 RTP/AVP 99

a=extmap:1 urn:ietf:params:rtp-hdrex:appId/ for support of dynamic mapping

a=rtpmap:99 H264/90000

a=max-send-ssrc:{*:6}

a=max-recv-ssrc:{*:4}

a=ssrc:11111 appId:1

a=ssrc:22222appId:2

a=ssrc:33333 appId:3

a=ssrc:44444appId:4

a=ssrc:55555 appId:5

a=ssrc:66666 appId:6

In the above example the provider can send up to five main streams and one presentation stream.

Note that VC1 and VC5 have the same SSRC since they are using the same resource.

- VC0- (the camera-left camera stream, purpose=main, switched:no, appId =1
- VC1- (the center camera stream, purpose=main, switched:no, appId =2
- VC2- (the camera-right camera stream), purpose=main, switched:no, appId =3
- VC3- (the loudest panel stream), purpose=main, switched:yes, appId =4
- VC4- (the loudest panel stream with PiPs), purpose=main, composed=true; switched:yes, appId =5
- VC5- (the zoomed out view of all people in the room), purpose=main, composed=no; switched:no, appId =2
- VC6- (presentation stream), purpose=presentation, switched:no, appId =6

Note: We can allocate an SSRC for each MC which will not require the indirection of using an appId. This will require if a switch to dynamic is done to provide information about which SSRC is being replaced by the new one.

6.2. Dynamic Mapping

For topologies that use dynamic mapping there is no need to provide the SSRCs in the offer (they may not be available if the offers from the sources will not include them when connecting to the mixer or remote endpoint) In this case the appId will be specified first in the advertisement.

The SIP offer may be

m=video 49200 RTP/AVP 99

a=extmap:1 urn:ietf:params:appId

```
a=rtpmap:99 H264/90000

a=max-send-ssrc:{*:4}

a=max-recv-ssrc:{*:4}
```

This will work for ssrc multiplex. It is not clear how it will work when RTP streams of the same media are not multiplexed in a single RTP session. How to know which encoding will be in which of the different RTP sessions.

7. Acknowledgements

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

8. IANA Considerations

TBD

9. Security Considerations

TBD.

10. References

10.1. Normative References

[I-D.even-mmusic-application-token]
Even, R., Lennox, J., and Q. Wu, "The Session Description Protocol (SDP) Application Token Attribute", draft-even-mmusic-application-token-01 (work in progress), September 2013.

[I-D.ietf-clue-framework]
Romanow, A., Duckworth, M., Pepperell, A., and B. Baldino, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-06 (work in progress), July 2012.

[I-D.lennox-clue-rtp-usage]
Lennox, J., Witty, P., and A. Romanow, "Real-Time Transport Protocol (RTP) Usage for Telepresence Sessions", draft-lennox-clue-rtp-usage-04 (work in progress), June 2012.

[I-D.westerlund-avtcore-max-ssrc]
Westerlund, M., Burman, B., and F. Jansson, "Multiple Synchronization sources (SSRC) in RTP Session Signaling",

draft-westerlund-avtcore-max-ssrc-02 (work in progress), July 2012.

[I-D.westerlund-avtext-rtcp-sdes-srcname]
Westerlund, M., Burman, B., and P. Sandgren, "RTCP SDES Item SRCNAME to Label Individual Sources", draft-westerlund-avtext-rtcp-sdes-srcname-01 (work in progress), July 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

[I-D.ietf-clue-telepresence-use-cases]
Romanow, A., Botzko, S., Duckworth, M., Even, R., and I. Communications, "Use Cases for Telepresence Multi-streams", draft-ietf-clue-telepresence-use-cases-04 (work in progress), August 2012.

[I-D.lennox-mmusic-sdp-source-selection]
Lennox, J. and H. Schulzrinne, "Mechanisms for Media Source Selection in the Session Description Protocol (SDP)", draft-lennox-mmusic-sdp-source-selection-04 (work in progress), March 2012.

[I-D.westerlund-avtcore-rtp-simulcast]
Westerlund, M., Burman, B., Lindqvist, M., and F. Jansson, "Using Simulcast in RTP sessions", draft-westerlund-avtcore-rtp-simulcast-01 (work in progress), July 2012.

[I-D.westerlund-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-westerlund-avtcore-rtp-topologies-update-01 (work in progress), October 2012.

[I-D.westerlund-avtext-codec-operation-point]
Westerlund, M., Burman, B., and L. Hamm, "Codec Operation Point RTCP Extension", draft-westerlund-avtext-codec-operation-point-00 (work in progress), March 2012.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with 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.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.

[RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.

[RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.

Authors' Addresses

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

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

Email: jonathan@vidyo.com

CLUE WG
Internet-Draft
Intended status: Standards Track
Expires: August 31, 2017

R. Even
Huawei Technologies
J. Lennox
Vidyo

February 27, 2017

Mapping RTP streams to CLUE Media Captures
draft-ietf-clue-rtp-mapping-14.txt

Abstract

This document describes how the Real Time transport Protocol (RTP) is used in the context of the CLUE protocol (ControLling mUltiple streams for tElepresence). It also describes the mechanisms and recommended practice for mapping RTP media streams defined in Session Description Protocol (SDP) to CLUE Media Captures and defines a new RTP header extension (CaptureId).

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 August 31, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Terminology 3
3. RTP topologies for CLUE 3
4. Mapping CLUE Capture Encodings to RTP streams 4
5. MCC Constituent CaptureID definition 5
5.1. RTCP CaptureID SDES Item 5
5.2. RTP Header Extension 6
6. Examples 6
7. Communication Security 7
8. Acknowledgments 8
9. IANA Considerations 8
10. Security Considerations 8
11. References 10
11.1. Normative References 10
11.2. Informative References 11
Authors' Addresses 13

1. Introduction

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

SIP Offer/Answer [RFC3264] uses SDP [RFC4566] to describe the RTP[RFC3550] media streams. Each RTP stream has a unique Synchronization Source (SSRC) within its RTP session. The content of the RTP stream is created by an encoder in the endpoint. This may be an original content from a camera or a content created by an intermediary device like an MCU (Multipoint Control Unit).

This document makes recommendations for the CLUE architecture about how RTP and RTCP streams should be encoded and transmitted, and how their relation to CLUE Media Captures should be communicated. The proposed solution supports multiple RTP topologies [RFC7667].

With regards to the media (audio, video and timed text), systems that support CLUE use RTP for the media, SDP for codec and media transport negotiation (CLUE individual encodings) and the CLUE protocol for Media Capture description and selection. In order to associate the

media in the different protocols there are three mapping that need to be specified:

1. CLUE individual encodings to SDP

2. RTP streams to SDP (this is not a CLUE specific mapping)

3. RTP streams to MC to map the received RTP steam to the current MC in the MCC.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119[RFC2119] and indicate requirement levels for RTP processing in compliant CLUE implementations.

The definitions from the CLUE framework document [I-D.ietf-clue-framework] section 3 are used by this document as well.

3. RTP topologies for CLUE

The typical RTP topologies used by CLUE Telepresence systems specify different behaviors for RTP and RTCP distribution. A number of RTP topologies are described in [RFC7667]. For CLUE telepresence, the relevant topologies include Point-to-Point, as well as Media-Mixing mixers, Media- Switching mixers, and Selective Forwarding Middleboxs.

In the Point-to-Point topology, one peer communicates directly with a single peer over unicast. There can be one or more RTP sessions, each sent on a separate 5-tuple, and having a separate SSRC space, with each RTP session carrying multiple RTP streams identified by their SSRC. All SSRCs are recognized by the peers based on the information in the RTCP Source description (SDES) report that includes the CNAME and SSRC of the sent RTP streams. There are different Point-to-Point use cases as specified in CLUE use case [RFC7205]. In some cases, a CLUE session which, at a high-level, is point-to-point may nonetheless have an RTP stream which is best described by one of the mixer topologies. For example, a CLUE endpoint can produce composite or switched captures for use by a receiving system with fewer displays than the sender has cameras. The Media Capture may be described using an MCC.

For the Media Mixer topology [RFC7667], the peers communicate only with the mixer. The mixer provides mixed or composited media streams, using its own SSRC for the sent streams. If needed by CLUE

endpoint, the conference roster information including conference participants, endpoints, media and media-id (SSRC) can be determined using the conference event package [RFC4575] element.

Media-switching mixers and Selective Forwarding Middleboxes behave as described in [RFC7667]

4. Mapping CLUE Capture Encodings to RTP streams

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

Most video conferencing systems today can separate multiple RTP sources by placing them into RTP sessions using the SDP description; the video conferencing application can also have some knowledge about the purpose of each RTP session. For example, video conferencing applications that have a primary video source and a slides video source can send each media source in a separate RTP session with a content attribute [RFC4796] enabling different application behavior for each received RTP media source. Demultiplexing is straightforward because each media capture is sent as a single RTP stream, with each RTP stream being sent in a separate RTP session, on a distinct UDP 5-tuple. This will also be true for mapping the RTP streams to Media Captures Encodings if each Media Capture Encodings uses a separate RTP session, and the consumer can identify it based on the receiving RTP port. In this case, SDP only needs to label the RTP session with an identifier that can be used to identify the Media Capture in the CLUE description. The SDP label attribute serves as this identifier.

Each Capture Encoding MUST be sent as a separate RTP stream. CLUE endpoints MUST support sending each such RTP stream in a separate RTP session signalled by an SDP m= line. They MAY also support sending some or all of the RTP streams in a single RTP session, using the mechanism described in [I-D.ietf-mmusic-sdp-bundle-negotiation] to relate RTP streams to SDP m= lines.

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

5. MCC Constituent CaptureID definition

For a MCC which can represent multiple switched MCs there is a need to know which MC is represented in the current RTP stream at any given time. This requires a mapping from the SSRC of the RTP stream conveying a particular MCC to the constituent MC. In order to address this mapping this document defines an RTP header extension and SDES item that includes the captureID of the original MC, allowing the consumer to use the original source MC's attributes like the spatial information.

This mapping temporarily associates the SSRC of the RTP stream conveying a particular MCC with the captureID of the single original MC that is currently switched into the MCC. This mapping cannot be used for the composed case where more than one original MC is composed into the MCC simultaneously.

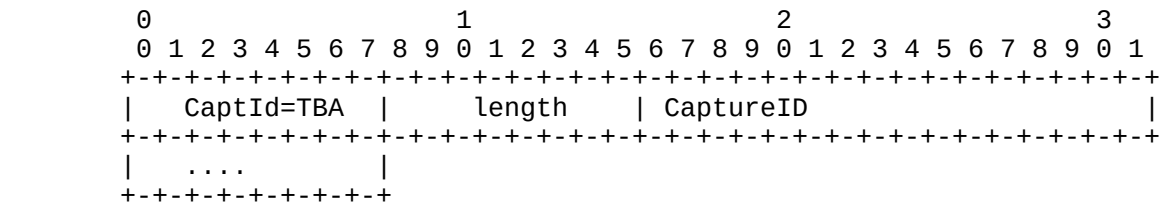
If there is only one MC in the MCC then the media provider MUST send the captureID of the current constituent MC in the RTP Header Extension and as a RTCP CaptureID SDES item. When the media provider switches the MC it sends within an MCC, it MUST send the captureID value for the MC just switched into the MCC in an RTP Header Extension and as a RTCP CaptureID SDES item as specified in [RFC7941]

If there is more than one MC composed into the MCC then the media provider MUST NOT send any of the MCs' captureIDs using this mechanism. However, if an MCC is sending contributing source (CSRC) information in the RTP header for a composed capture, it MAY send the captureID values in the RTCP SDES packets giving source information for the SSRC values sent as contributing sources (CSRCs).

If the media provider sends the captureID of a single MC switched into an MCC, then later sends one composed stream of multiple MCs in the same MCC, it MUST send the special value "-", a single dash character, as the captureID RTP Header Extension and RTCP CaptureID SDES item. The single dash character indicates there is no applicable value for the MCC constituent CaptureID. The media consumer interprets this as meaning that any previous CaptureID value associated with this SSRC no longer applies. As [I-D.ietf-clue-data-model-schema] defines the captureID syntax as "xs:ID", the single dash character is not a legal captureID value, so there is no possibility of confusing it with an actual captureID.

5.1. RTCP CaptureID SDES Item

This document specifies a new RTCP SDES item.



Note to the RFC Editor: Please replace TBA with the value assigned by IANA.

This CaptureID is a variable-length UTF-8 string corresponding either to a CaptureID negotiated in the CLUE protocol, or the single character "-".

This SDES item MUST be sent in an SDES packet within a compound RTCP packet unless support for Reduced-size RTCP has been negotiated as specified in RFC 5506 [RFC5506], in which case it can be sent as an SDES packet in a non-compound RTCP packet.

5.2. RTP Header Extension

The CaptureID is also carried in an RTP header extension [RFC5285], using the mechanism defined in [RFC7941].

Support is negotiated within SDP using the URN "urn:ietf:params:rtp-hdrext:sdes:CaptureID".

The CaptureID is sent in a RTP Header Extension because for switched captures, receivers need to know which original MC corresponds to the media being sent for an MCC, in order to correctly apply geometric adjustments to the received media.

As discussed in [RFC7941], there is no need to send the CaptId Header Extension with all RTP packets. Senders MAY choose to send it only when a new MC is sent. If such a mode is being used, the header extension SHOULD be sent in the first few RTP packets to reduce the risk of losing it due to packet loss. See [RFC7941] for more discussion of this.

6. Examples

In this partial advertisement the Media Provider advertises a composed capture VC7 made of 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).

```
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="ns2:videoCaptureType" captureID="VC7" mediaType="video">
          <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
          <ns2:nonSpatiallyDefinable>true</ns2:nonSpatiallyDefinable>
          <ns2:content>
                <ns2:captureIDREF>VC3</ns2:captureIDREF>
                <ns2:captureIDREF>VC5</ns2:captureIDREF>
                <ns2:captureIDREF>VC6</ns2:captureIDREF>
          </ns2:content>
                  <ns2:maxCaptures>3</ns2:maxCaptures>
            <ns2:allowSubsetChoice>false</ns2:allowSubsetChoice>
          <ns2:description lang="en">big picture of the current speaker
            pips about previous speakers</ns2:description>
            <ns2:priority>1</ns2:priority>
            <ns2:lang>it</ns2:lang>
            <ns2:mobility>static</ns2:mobility>
            <ns2:view>individual</ns2:view>
        </ns2:mediaCapture>
```

In this case the media provider will send capture IDs VC3, VC5 or VC6 as an RTP header extension and RTCP SDES message for the RTP stream associated with the MC.

Note that this is part of the full advertisement message example from CLUE data model[I-D.ietf-clue-data-model-schema] example and is not a valid xml document.

7. Communication Security

CLUE endpoints MUST support RTP/SAVPF profile and SRTP [RFC3711]. CLUE endpoints MUST support DTLS [RFC6347] and DTLS-SRTP [RFC5763] [RFC5764] for SRTP keying.

All media channels SHOULD be secure via SRTP and the RTP/SAVPF profile unless the RTP media and its associated RTCP are secure by other means (see [RFC7201] [RFC7202]).

All CLUE implementations MUST implement DTLS 1.0, with the cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA with the the P-256 curve [FIPS186]. The DTLS-SRTP protection profile SRTP_AES128_CM_HMAC_SHA1_80 MUST be supported for SRTP.Encrypted SRTP Header extensions [RFC6904] MUST be supported.

Implementations SHOULD implement DTLS 1.2 with the TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 cipher suite. Implementations MUST favor cipher suites which support PFS over non-PFS cipher suites and SHOULD favor AEAD over non-AEAD cipher suites.

NULL Protection profiles MUST NOT be used for RTP or RTCP.

CLUE endpoint MUST generate short-term persistent RTCP CNAMES, as specified in [RFC7022], and thus can't be used for long term tracking of the users.

8. Acknowledgments

The authors would like to thanks Allyn Romanow and Paul Witty for contributing text to this work. Magnus Westerlund helped drafting the security section.

9. IANA Considerations

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

Extension URI: urn:ietf:params:rtp-hdrext:sdes:CaptId

Description: CLUE CaptId

Contact: ron.even.tlv@gmail.com

Reference: RFC XXXX

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

Value	Abbrev	Name	Reference
TBA	CCID	CLUE CaptId	[RFCXXXX]

Note to the RFC Editor: Please replace RFCXXXX with this RFC number.

10. Security Considerations

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

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

profile and DTLS-SRTP keying [RFC5764] as defined in the communication security section of this memo Section 7

RTCP packets convey a Canonical Name (CNAME) identifier that is used to associate RTP packet streams that need to be synchronised across related RTP sessions. Inappropriate choice of CNAME values can be a privacy concern, since long-term persistent CNAME identifiers can be used to track users across multiple calls. The communication security section of this memo Section 7 mandates generation of short-term persistent RTCP CNAMES, as specified in [RFC7022] so they can't be used for long term tracking of the users.

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

1. the RTCP bandwidth could be configured to make the regular reporting interval so large that effective congestion control cannot be maintained, potentially leading to denial of service due to congestion caused by the media traffic;

2. the RTCP interval could be configured to a very small value, causing endpoints to generate high rate RTCP traffic, potentially leading to denial of service due to the non-congestion controlled RTCP traffic; and

3. RTCP parameters could be configured differently for each endpoint, with some of the endpoints using a large reporting interval and some using a smaller interval, leading to denial of service due to premature participant timeouts due to mismatched timeout periods which are based on the reporting interval (this is a particular concern if endpoints use a small but non-zero value for the RTP/AVPF minimal receiver report interval (trr-int) [RFC4585], as discussed in [I-D.ietf-avtcore-rtp-multi-stream]).

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

interval small enough that the RTCP bandwidth would exceed the media bandwidth.

The guidelines in [RFC6562] apply when using variable bit rate (VBR) audio codecs such as Opus.

The use of the encryption of the header extensions are RECOMMENDED, unless there are known reasons, like RTP middleboxes performing voice activity based source selection or third party monitoring that will greatly benefit from the information, and this has been expressed using API or signalling. If further evidence are produced to show that information leakage is significant from audio level indications, then use of encryption needs to be mandated at that time.

In multi-party communication scenarios using RTP Middleboxes; this middleboxes are REQUIRED, by this protocol, to not weaken the sessions' security. The middlebox SHOULD maintain the confidentiality, integrity and perform source authentication. The middlebox MAY perform checks that prevents any endpoint participating in a conference to impersonate another. Some additional security considerations regarding multi-party topologies can be found in [RFC7667]

The CaptureID is created as part of the CLUE protocol. The CaptId SDES item is used to convey the same CaptureID value in the SDES item. When sending the SDES item the security consideration specified in the security section of [RFC7941] and in the communication security section of this memo Section 7 are applicable. Note that since the CaptureID is carried also in CLUE protocol messages it is RECOMMENDED that this SDES item use at least similar protection profiles as the CLUE protocol messages carried in the CLUE data channel. .

11. References

11.1. Normative References

[I-D.ietf-clue-data-model-schema]
Presta, R. and S. Romano, "An XML Schema for the CLUE data model", draft-ietf-clue-data-model-schema-17 (work in progress), August 2016.

[I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-25 (work in progress), January 2016.

[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-36 (work in progress), October 2016.

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

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <http://www.rfc-editor.org/info/rfc3711>.

[RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May 2010, <http://www.rfc-editor.org/info/rfc5763>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <http://www.rfc-editor.org/info/rfc5764>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, DOI 10.17487/RFC6904, April 2013, <http://www.rfc-editor.org/info/rfc6904>.

[RFC7941] Westerlund, M., Burman, B., Even, R., and M. Zanaty, "RTP Header Extension for the RTP Control Protocol (RTCP) Source Description Items", RFC 7941, DOI 10.17487/RFC7941, August 2016, <http://www.rfc-editor.org/info/rfc7941>.

11.2. Informative References

[FIPS186] National Institute of Standards and Technology, "Digital Signature Standard", FIPS PUB 186-4, July 2013.

[I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-11 (work in progress), December 2015.

[I-D.ietf-clue-signaling]
Kyzivat, P., Xiao, L., Groves, C., and R. Hansen, "CLUE Signaling", draft-ietf-clue-signaling-10 (work in progress), January 2017.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <http://www.rfc-editor.org/info/rfc3264>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, DOI 10.17487/RFC3556, July 2003, <http://www.rfc-editor.org/info/rfc3556>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, Ed., "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, DOI 10.17487/RFC4575, August 2006, <http://www.rfc-editor.org/info/rfc4575>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, DOI 10.17487/RFC4796, February 2007, <http://www.rfc-editor.org/info/rfc4796>.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <http://www.rfc-editor.org/info/rfc5124>.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <http://www.rfc-editor.org/info/rfc5285>.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <http://www.rfc-editor.org/info/rfc5506>.

[RFC6562] Perkins, C. and JM. Valin, "Guidelines for the Use of Variable Bit Rate Audio with Secure RTP", RFC 6562, DOI 10.17487/RFC6562, March 2012, <http://www.rfc-editor.org/info/rfc6562>.

[RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, DOI 10.17487/RFC7022, September 2013, <http://www.rfc-editor.org/info/rfc7022>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

[RFC7202] Perkins, C. and M. Westerlund, "Securing the RTP Framework: Why RTP Does Not Mandate a Single Media Security Solution", RFC 7202, DOI 10.17487/RFC7202, April 2014, <http://www.rfc-editor.org/info/rfc7202>.

[RFC7205] Romanow, A., Botzko, S., Duckworth, M., and R. Even, Ed., "Use Cases for Telepresence Multistreams", RFC 7205, DOI 10.17487/RFC7205, April 2014, <http://www.rfc-editor.org/info/rfc7205>.

[RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <http://www.rfc-editor.org/info/rfc7667>.

Authors' Addresses

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@huawei.com

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

Email: jonathan@vidyo.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 8, 2015

R. Jesup
Mozilla
S. Loreto
Ericsson
M. Tuexen
Muenster Univ. of Appl. Sciences
January 4, 2015

WebRTC Data Channels
draft-ietf-rtcweb-data-channel-13.txt

Abstract

The WebRTC framework specifies protocol support for direct interactive rich communication using audio, video, and data between two peers' web-browsers. This document specifies the non-media data transport aspects of the WebRTC framework. It provides an architectural overview of how the Stream Control Transmission Protocol (SCTP) is used in the WebRTC context as a generic transport service allowing WEB-browsers to exchange generic data from peer to peer.

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 July 8, 2015.

Copyright Notice

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

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

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Conventions 3
3. Use Cases 3
3.1. Use Cases for Unreliable Data Channels 4
3.2. Use Cases for Reliable Data Channels 4
4. Requirements 4
5. SCTP over DTLS over UDP Considerations 6
6. The Usage of SCTP for Data Channels 8
6.1. SCTP Protocol Considerations 8
6.2. SCTP Association Management 9
6.3. SCTP Streams 9
6.4. Data Channel Definition 10
6.5. Opening a Data Channel 10
6.6. Transferring User Data on a Data Channel 11
6.7. Closing a Data Channel 12
7. Security Considerations 13
8. IANA Considerations 13
9. Acknowledgments 14
10. References 14
10.1. Normative References 14
10.2. Informative References 15
Authors' Addresses 16

1. Introduction

In the WebRTC framework, communication between the parties consists of media (for example audio and video) and non-media data. Media is sent using SRTP, and is not specified further here. Non-media data is handled by using SCTP [RFC4960] encapsulated in DTLS. DTLS 1.0 is defined in [RFC4347] and the present latest version, DTLS 1.2, is defined in [RFC6347].

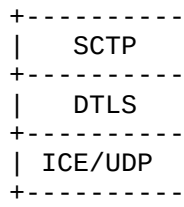


Figure 1: Basic stack diagram

The encapsulation of SCTP over DTLS (see [I-D.ietf-tsvwg-sctp-dtls-encaps]) over ICE/UDP (see [RFC5245]) provides a NAT traversal solution together with confidentiality, source authentication, and integrity protected transfers. This data transport service operates in parallel to the SRTP media transports, and all of them can eventually share a single UDP port number.

SCTP as specified in [RFC4960] with the partial reliability extension defined in [RFC3758] and the additional policies defined in [I-D.ietf-tsvwg-sctp-prpolicies] provides multiple streams natively with reliable, and the relevant partially-reliable delivery modes for user messages. Using the reconfiguration extension defined in [RFC6525] allows to increase the number of streams during the lifetime of an SCTP association and to reset individual SCTP streams. Using [I-D.ietf-tsvwg-sctp-ndata] allows to interleave large messages to avoid the monopolization and adds the support of prioritizing of SCTP streams.

The remainder of this document is organized as follows: Section 3 and Section 4 provide use cases and requirements for both unreliable and reliable peer to peer data channels; Section 5 discusses SCTP over DTLS over UDP; Section 6 provides the specification of how SCTP should be used by the WebRTC protocol framework for transporting non-media data between WEB-browsers.

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 [RFC2119].

3. Use Cases

This section defines use cases specific to data channels. Please note that this section is informational only.

3.1. Use Cases for Unreliable Data Channels

U-C 1: A real-time game where position and object state information is sent via one or more unreliable data channels. Note that at any time there may be no SRTP media channels, or all SRTP media channels may be inactive, and that there may also be reliable data channels in use.

U-C 2: Providing non-critical information to a user about the reason for a state update in a video chat or conference, such as mute state.

3.2. Use Cases for Reliable Data Channels

U-C 3: A real-time game where critical state information needs to be transferred, such as control information. Such a game may have no SRTP media channels, or they may be inactive at any given time, or may only be added due to in-game actions.

U-C 4: Non-realtime file transfers between people chatting. Note that this may involve a large number of files to transfer sequentially or in parallel, such as when sharing a folder of images or a directory of files.

U-C 5: Realtime text chat during an audio and/or video call with an individual or with multiple people in a conference.

U-C 6: Renegotiation of the configuration of the PeerConnection.

U-C 7: Proxy browsing, where a browser uses data channels of a PeerConnection to send and receive HTTP/HTTPS requests and data, for example to avoid local Internet filtering or monitoring.

4. Requirements

This section lists the requirements for P2P data channels between two browsers. Please note that this section is informational only.

Req. 1: Multiple simultaneous data channels must be supported. Note that there may be 0 or more SRTP media streams in parallel with the data channels in the same PeerConnection, and the number and state (active/inactive) of these SRTP media streams may change at any time.

Req. 2: Both reliable and unreliable data channels must be supported.

Req. 3: Data channels of a PeerConnection must be congestion controlled; either individually, as a class, or in conjunction with the SRTP media streams of the PeerConnection, to ensure that data channels don't cause congestion problems for these SRTP media streams, and that the WebRTC PeerConnection does not cause excessive problems when run in parallel with TCP connections.

Req. 4: The application should be able to provide guidance as to the relative priority of each data channel relative to each other, and relative to the SRTP media streams. This will interact with the congestion control algorithms.

Req. 5: Data channels must be secured; allowing for confidentiality, integrity and source authentication. See [I-D.ietf-rtcweb-security] and [I-D.ietf-rtcweb-security-arch] for detailed info.

Req. 6: Data channels must provide message fragmentation support such that IP-layer fragmentation can be avoided no matter how large a message the JavaScript application passes to be sent. It also must ensure that large data channel transfers don't unduly delay traffic on other data channels.

Req. 7: The data channel transport protocol must not encode local IP addresses inside its protocol fields; doing so reveals potentially private information, and leads to failure if the address is depended upon.

Req. 8: The data channel transport protocol should support unbounded-length "messages" (i.e., a virtual socket stream) at the application layer, for such things as image-file-transfer; Implementations might enforce a reasonable message size limit.

Req. 9: The data channel transport protocol should avoid IP fragmentation. It must support PMTU (Path MTU) discovery and must not rely on ICMP or ICMPv6 being generated or being passed back, especially for PMTU discovery.

Req. 10: It must be possible to implement the protocol stack in the user application space.

5. SCTP over DTLS over UDP Considerations

The important features of SCTP in the WebRTC context are:

- Usage of a TCP-friendly congestion control.
- The congestion control is modifiable for integration with the SRTP media stream congestion control.
- Support of multiple unidirectional streams, each providing its own notion of ordered message delivery.
- Support of ordered and out-of-order message delivery.
- Supporting arbitrary large user messages by providing fragmentation and reassembly.
- Support of PMTU-discovery.
- Support of reliable or partially reliable message transport.

The WebRTC Data Channel mechanism does not support SCTP multihoming. The SCTP layer will simply act as if it were running on a single-homed host, since that is the abstraction that the DTLS layer (a connection oriented, unreliable datagram service) exposes.

The encapsulation of SCTP over DTLS defined in [I-D.ietf-tsvwg-sctp-dtls-encaps] provides confidentiality, source authenticated, and integrity protected transfers. Using DTLS over UDP in combination with ICE enables middlebox traversal in IPv4 and IPv6 based networks. SCTP as specified in [RFC4960] MUST be used in combination with the extension defined in [RFC3758] and provides the following features for transporting non-media data between browsers:

- Support of multiple unidirectional streams.
- Ordered and unordered delivery of user messages.
- Reliable and partial-reliable transport of user messages.

Each SCTP user message contains a Payload Protocol Identifier (PPID) that is passed to SCTP by its upper layer on the sending side and provided to its upper layer on the receiving side. The PPID can be used to multiplex/demultiplex multiple upper layers over a single SCTP association. In the WebRTC context, the PPID is used to distinguish between UTF-8 encoded user data, binary encoded userdata and the Data Channel Establishment Protocol defined in

[I-D.ietf-rtcweb-data-protocol]. Please note that the PPID is not accessible via the Javascript API.

The encapsulation of SCTP over DTLS, together with the SCTP features listed above satisfies all the requirements listed in Section 4.

The layering of protocols for WebRTC is shown in the following Figure 2.

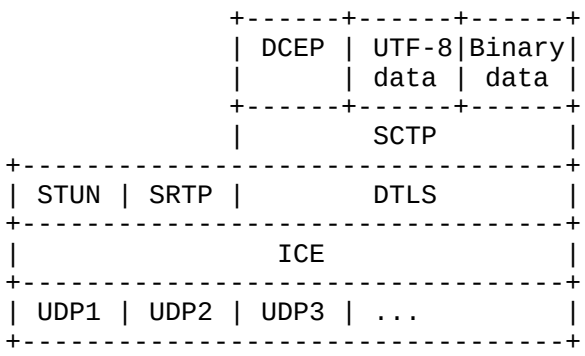


Figure 2: WebRTC protocol layers

This stack (especially in contrast to DTLS over SCTP [RFC6083] in combination with SCTP over UDP [RFC6951]) has been chosen because it

- supports the transmission of arbitrary large user messages.
- shares the DTLS connection with the SRTP media channels of the PeerConnection.
- provides privacy for the SCTP control information.

Considering the protocol stack of Figure 2 the usage of DTLS 1.0 over UDP is specified in [RFC4347] and the usage of DTLS 1.2 over UDP in specified in [RFC6347], while the usage of SCTP on top of DTLS is specified in [I-D.ietf-tsvwg-sctp-dtls-encaps]. Please note that the demultiplexing STUN vs. SRTP vs. DTLS is done as described in Section 5.1.2 of [RFC5764] and SCTP is the only payload of DTLS.

Since DTLS is typically implemented in user application space, the SCTP stack also needs to be a user application space stack.

The ICE/UDP layer can handle IP address changes during a session without needing interaction with the DTLS and SCTP layers. However, SCTP SHOULD be notified when an address changes has happened. In this case SCTP SHOULD retest the Path MTU and reset the congestion

state to the initial state. In case of a window based congestion control like the one specified in [RFC4960], this means setting the congestion window and slow start threshold to its initial values.

Incoming ICMP or ICMPv6 messages can't be processed by the SCTP layer, since there is no way to identify the corresponding association. Therefore SCTP MUST support performing Path MTU discovery without relying on ICMP or ICMPv6 as specified in [RFC4821] using probing messages specified in [RFC4820]. The initial Path MTU at the IP layer SHOULD NOT exceed 1200 bytes for IPv4 and 1280 for IPv6.

In general, the lower layer interface of an SCTP implementation should be adapted to address the differences between IPv4 and IPv6 (being connection-less) or DTLS (being connection-oriented).

When the protocol stack of Figure 2 is used, DTLS protects the complete SCTP packet, so it provides confidentiality, integrity and source authentication of the complete SCTP packet.

SCTP provides congestion control on a per-association base. This means that all SCTP streams within a single SCTP association share the same congestion window. Traffic not being sent over SCTP is not covered by the SCTP congestion control. Using a congestion control different from than the standard one might improve the impact on the parallel SRTP media streams.

SCTP uses the same port number concept as TCP and UDP do. Therefore an SCTP association uses two port numbers, one at each SCTP end-point.

6. The Usage of SCTP for Data Channels

6.1. SCTP Protocol Considerations

The DTLS encapsulation of SCTP packets as described in [I-D.ietf-tsvwg-sctp-dtls-encaps] MUST be used.

This SCTP stack and its upper layer MUST support the usage of multiple SCTP streams. A user message can be sent ordered or unordered and with partial or full reliability.

The following SCTP protocol extensions are required:

- The stream reconfiguration extension defined in [RFC6525] MUST be supported. It is used for closing channels.

- 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] are OPTIONAL.

- The partial reliability extension defined in [RFC3758] MUST be supported. In addition to the timed reliability PR-SCTP policy defined in [RFC3758], the limited retransmission policy defined in [I-D.ietf-tsvwg-sctp-prpolicies] MUST be supported. Limiting the number of retransmissions to zero combined with unordered delivery provides a UDP-like service where each user message is sent exactly once and delivered in the order received.

The support for message interleaving as defined in [I-D.ietf-tsvwg-sctp-ndata] SHOULD be used.

6.2. SCTP Association Management

In the WebRTC context, the SCTP association will be set up when the two endpoints of the WebRTC PeerConnection agree on opening it, as negotiated by JSEP (typically an exchange of SDP) [I-D.ietf-rtcweb-jsep]. It will use the DTLS connection selected via ICE; typically this will be shared via BUNDLE or equivalent with DTLS connections used to key the SRTP media streams.

The number of streams negotiated during SCTP association setup SHOULD be 65535, which is the maximum number of streams that can be negotiated during the association setup.

SCTP supports two ways of terminating an SCTP association. A graceful one, using a procedure which ensures that no messages are lost during the shutdown of the association. The second method is a non-graceful one, where one side can just abort the association.

Each SCTP end-point supervises continuously the reachability of its peer by monitoring the number of retransmissions of user messages and test messages. In case of excessive retransmissions, the association is terminated in a non-graceful way.

If an SCTP association is closed in a graceful way, all of its data channels are closed. In case of a non-graceful teardown, all data channels are also closed, but an error indication SHOULD be provided if possible.

6.3. SCTP Streams

SCTP defines a stream as a unidirectional logical channel existing within an SCTP association to another SCTP endpoint. The streams are used to provide the notion of in-sequence delivery and for

multiplexing. Each user message is sent on a particular stream, either ordered or unordered. Ordering is preserved only for ordered messages sent on the same stream.

6.4. Data Channel Definition

Data channels are defined such that their accompanying application-level API can closely mirror the API for WebSockets, which implies bidirectional streams of data and a textual field called 'label' used to identify the meaning of the data channel.

The realization of a data channel is a pair of one incoming stream and one outgoing SCTP stream having the same SCTP stream identifier. How these SCTP stream identifiers are selected is protocol and implementation dependent. This allows a bidirectional communication.

Additionally, each data channel has the following properties in each direction:

- reliable or unreliable message transmission. In case of unreliable transmissions, the same level of unreliability is used. Please note that in SCTP this is a property of an SCTP user message and not of an SCTP stream.
- in-order or out-of-order message delivery for message sent. Please note that in SCTP this is a property of an SCTP user message and not of an SCTP stream.
- A priority, which is a 2 byte unsigned integer. These priorities MUST be interpreted as weighted-fair-queuing scheduling priorities per the definition of the corresponding stream scheduler supporting interleaving in [I-D.ietf-tsvwg-sctp-ndata]. For use in WebRTC, the values used SHOULD be one of 128 ("below normal"), 256 ("normal"), 512 ("high") or 1024 ("extra high").
- an optional label.
- an optional protocol.

Please note that for a data channel being negotiated with the protocol specified in [I-D.ietf-rtcweb-data-protocol] all of the above properties are the same in both directions.

6.5. Opening a Data Channel

Data channels can be opened by using negotiation within the SCTP association, called in-band negotiation, or out-of-band negotiation. Out-of-band negotiation is defined as any method which results in an

agreement as to the parameters of a channel and the creation thereof. The details are out of scope of this document. Applications using data channels need to use the negotiation methods consistently on both end-points.

A simple protocol for in-band negotiation is specified in [I-D.ietf-rtcweb-data-protocol].

When one side wants to open a channel using out-of-band negotiation, it picks a stream. Unless otherwise defined or negotiated, the streams are picked based on the DTLS role (the client picks even stream identifiers, the server odd stream identifiers). However, the application is responsible for avoiding collisions with existing streams. If it attempts to re-use a stream which is part of an existing data channel, the addition MUST fail. In addition to choosing a stream, the application SHOULD also determine the options to use for sending messages. The application MUST ensure in an application-specific manner that the application at the peer will also know the selected stream to be used, and the options for sending data from that side.

6.6. Transferring User Data on a Data Channel

All data sent on a data channel in both directions MUST be sent over the underlying stream using the reliability defined when the data channel was opened unless the options are changed, or per-message options are specified by a higher level.

The message-orientation of SCTP is used to preserve the message boundaries of user messages. Therefore, senders MUST NOT put more than one application message into an SCTP user message. Unless the deprecated PPID-based fragmentation and reassembly is used, the sender MUST include exactly one application message in each SCTP user message.

The SCTP Payload Protocol Identifiers (PPIDs) are used to signal the interpretation of the "Payload data". The following PPIDs MUST be used (see Section 8):

WebRTC String: to identify a non-empty JavaScript string encoded in UTF-8.

WebRTC String Empty: to identify an empty JavaScript string encoded in UTF-8.

WebRTC Binary: to identify a non-empty JavaScript binary data (ArrayBuffer, ArrayBufferView or Blob).

WebRTC Binary Empty: to identify an empty JavaScript binary data (ArrayBuffer, ArrayBufferView or Blob).

SCTP does not support the sending of empty user messages. Therefore, if an empty message has to be sent, the appropriate PPID (WebRTC String Empty or WebRTC Binary Empty) is used and the SCTP user message of one zero byte is sent. When receiving an SCTP user message with one of these PPIDs, the receiver MUST ignore the SCTP user message and process it as an empty message.

The usage of the PPIDs "WebRTC String Partial" and "WebRTC Binary Partial" is deprecated. They were used for a PPID-based fragmentation and reassembly of user messages belonging to reliable and ordered data channels.

If a message with an unsupported PPID is received or some error condition related to the received message is detected by the receiver (for example, illegal ordering), the receiver SHOULD close the corresponding data channel. This implies in particular that extensions using additional PPIDs can't be used without prior negotiation.

The SCTP base protocol specified in [RFC4960] does not support the interleaving of user messages. Therefore sending a large user message can monopolize the SCTP association. To overcome this limitation, [I-D.ietf-tsvwg-sctp-ndata] defines an extension to support message interleaving, which SHOULD be used. As long as message interleaving is not supported, the sender SHOULD limit the maximum message size to 16 KB to avoid monopolization.

It is recommended that the message size be kept within certain size bounds as applications will not be able to support arbitrarily-large single messages. This limit has to be negotiated, for example by using [I-D.ietf-mmusic-sctp-sdp].

The sender SHOULD disable the Nagle algorithm (see [RFC1122]) to minimize the latency.

6.7. Closing a Data Channel

Closing of a data channel MUST be signaled by resetting the corresponding outgoing streams [RFC6525]. This means that if one side decides to close the data channel, it resets the corresponding outgoing stream. When the peer sees that an incoming stream was reset, it also resets its corresponding outgoing stream. Once this is completed, the data channel is closed. Resetting a stream sets the Stream Sequence Numbers (SSNs) of the stream back to 'zero' with a corresponding notification to the application layer that the reset

has been performed. Streams are available for reuse after a reset has been performed.

[RFC6525] also guarantees that all the messages are delivered (or abandoned) before the stream is reset.

7. Security Considerations

This document does not add any additional considerations to the ones given in [I-D.ietf-rtcweb-security] and [I-D.ietf-rtcweb-security-arch].

It should be noted that a receiver must be prepared that the sender tries to send arbitrary large messages.

8. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

This document uses six already registered SCTP Payload Protocol Identifiers (PPIDs): "DOMString Last", "Binary Data Partial", "Binary Data Last", "DOMString Partial", "WebRTC String Empty", and "WebRTC Binary Empty". [RFC4960] creates the registry "SCTP Payload Protocol Identifiers" from which these identifiers were assigned. IANA is requested to update the reference of these six assignments to point to this document and change the names of the first four PPIDs. The corresponding dates should be kept.

Therefore these six assignments should be updated to read:

Value	SCTP PPID	Reference	Date
WebRTC String	51	[RFCXXXX]	2013-09-20
WebRTC Binary Partial (Deprecated)	52	[RFCXXXX]	2013-09-20
WebRTC Binary	53	[RFCXXXX]	2013-09-20
WebRTC String Partial (Deprecated)	54	[RFCXXXX]	2013-09-20
WebRTC String Empty	56	[RFCXXXX]	2014-08-22
WebRTC Binary Empty	57	[RFCXXXX]	2014-08-22

9. Acknowledgments

Many thanks for comments, ideas, and text from Harald Alvestrand, Richard Barnes, Adam Bergkvist, Alissa Cooper, Benoit Claise, Spencer Dawkins, Gunnar Hellstrom, Christer Holmberg, Cullen Jennings, Paul Kyzivat, Eric Rescorla, Adam Roach, Irene Ruengeler, Randall Stewart, Martin Stiemerling, Justin Uberti, and Magnus Westerlund.

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.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, March 2007.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[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.

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

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, February 2012.

[I-D.ietf-tsvwg-sctp-ndata]
Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "Stream Schedulers and a New Data Chunk for the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-ndata-01 (work in progress), July 2014.

[I-D.ietf-rtcweb-data-protocol]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Establishment Protocol", draft-ietf-rtcweb-data-protocol-08 (work in progress), September 2014.

[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-07 (work in progress), December 2014.

[I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", draft-ietf-rtcweb-security-07 (work in progress), July 2014.

[I-D.ietf-rtcweb-security-arch]
Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-10 (work in progress), July 2014.

[I-D.ietf-rtcweb-jsep]
Uberti, J., Jennings, C., and E. Rescorla, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-08 (work in progress), October 2014.

[I-D.ietf-tsvwg-sctp-prpolicies]
Tuexen, M., Seggelmann, R., Stewart, R., and S. Loreto, "Additional Policies for the Partial Reliability Extension of the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-prpolicies-06 (work in progress), December 2014.

[I-D.ietf-mmusic-sctp-sdp]
Holmberg, C., Loreto, S., and G. Camarillo, "Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-11 (work in progress), December 2014.

10.2. Informative References

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

[RFC6083] Tuexen, M., Seggelmann, R., and E. Rescorla, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC 6083, January 2011.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013.

Authors' Addresses

Randell Jesup
Mozilla
US

Email: randell-ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: salvatore.loreto@ericsson.com

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
Steinfurt 48565
DE

Email: tuexen@fh-muenster.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 8, 2015

R. Jesup
Mozilla
S. Loreto
Ericsson
M. Tuexen
Muenster Univ. of Appl. Sciences
January 4, 2015

WebRTC Data Channel Establishment Protocol
draft-ietf-rtcweb-data-protocol-09.txt

Abstract

The WebRTC framework specifies protocol support for direct interactive rich communication using audio, video, and data between two peers' web-browsers. This document specifies a simple protocol for establishing symmetric Data Channels between the peers. It uses a two way handshake and allows sending of user data without waiting for the handshake to complete.

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 July 8, 2015.

Copyright Notice

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

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

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Conventions 2
3. Terminology 3
4. Protocol Overview 3
5. Message Formats 4
5.1. DATA_CHANNEL_OPEN Message 4
5.2. DATA_CHANNEL_ACK Message 7
6. Procedures 7
7. Security Considerations 8
8. IANA Considerations 9
8.1. SCTP Payload Protocol Identifier 9
8.2. New Standalone Registry for the DCEP 9
8.2.1. New Message Type Registry 9
8.2.2. New Channel Type Registry 10
9. Acknowledgments 11
10. References 11
10.1. Normative References 11
10.2. Informational References 12
Authors' Addresses 12

1. Introduction

The Data Channel Establishment Protocol (DCEP) is designed to provide, in the WebRTC Data Channel context [I-D.ietf-rtcweb-data-channel], a simple in-band method to open symmetric Data Channels. As discussed in [I-D.ietf-rtcweb-data-channel], the protocol uses the Stream Control Transmission Protocol (SCTP) [RFC4960] encapsulated in the Datagram Transport Layer Security (DTLS) as described in [I-D.ietf-tsvwg-sctp-dtls-encaps] to benefit from their already standardized transport and security features. DTLS 1.0 is defined in [RFC4347] and the present latest version, DTLS 1.2, is defined in [RFC6347].

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 [RFC2119].

3. Terminology

This document uses the following terms:

Association: An SCTP association.

Stream: A unidirectional stream of an SCTP association. It is uniquely identified by an SCTP stream identifier (0-65534). Note: the SCTP stream identifier 65535 is reserved due to SCTP INIT and INIT-ACK chunks only allowing a maximum of 65535 Streams to be negotiated (0-65534).

Stream Identifier: The SCTP stream identifier uniquely identifying a Stream.

Data Channel: Two Streams with the same Stream Identifier, one in each direction, which are managed together.

4. Protocol Overview

The Data Channel Establishment Protocol is a simple, low-overhead way to establish bidirectional Data Channels over an SCTP association with a consistent set of properties.

The set of consistent properties includes:

- reliable or unreliable message transmission. In case of unreliable transmissions, the same level of unreliability is used.
- in-order or out-of-order message delivery.
- the priority of the Data Channel.
- an optional label for the Data Channel.
- an optional protocol for the Data Channel.
- the Streams.

This protocol uses a two way handshake to open a Data Channel. The handshake pairs one incoming and one outgoing Stream, both having the same Stream Identifier, into a single bidirectional Data Channel. The peer that initiates opening a Data Channel selects a Stream Identifier for which the corresponding incoming and outgoing Streams are unused and sends a DATA_CHANNEL_OPEN message on the outgoing Stream. The peer responds with a DATA_CHANNEL_ACK message on its corresponding outgoing Stream. Then the Data Channel is open. Data Channel Establishment Protocol messages are sent on the same Stream

as the user messages belonging to the Data Channel. The demultiplexing is based on the SCTP payload protocol identifier (PPID), since the Data Channel Establishment Protocol uses a specific PPID.

Note: The opening side MAY send user messages before the DATA_CHANNEL_ACK is received.

To avoid collisions where both sides try to open a Data Channel with the same Stream Identifiers, each side MUST use Streams with either even or odd Stream Identifiers when sending a DATA_CHANNEL_OPEN message. When using SCTP over DTLS [I-D.ietf-tsvwg-sctp-dtls-encaps], the method used to determine which side uses odd or even is based on the underlying DTLS connection role: the side acting as the DTLS client MUST use Streams with even Stream Identifiers, the side acting as the DTLS server MUST use Streams with odd Stream Identifiers.

Note: There is no attempt to ensure uniqueness for the label; if both sides open a Data Channel labeled "x" at the same time, there will be two Data Channels labeled "x" - one on an even Stream pair, one on an odd pair.

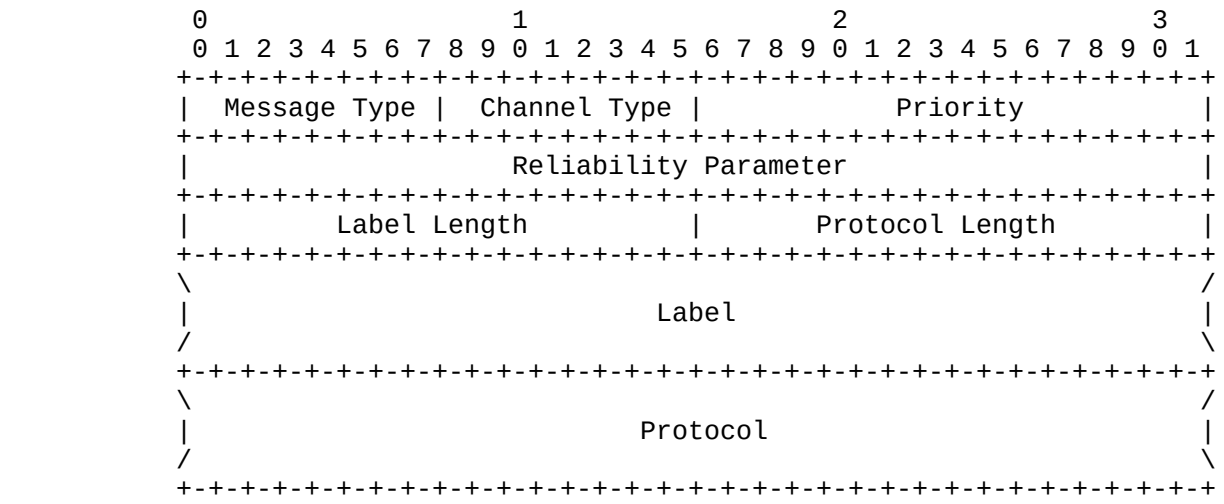
The protocol field is to ease cross-application interoperation ("federation") by identifying the user data being passed with an IANA-registered string (’WebSocket Subprotocol Name Registry’ defined in [RFC6455]), and may be useful for homogeneous applications which may create more than one type of Data Channel. Please note that there is also no attempt to ensure uniqueness for the protocol field.

5. Message Formats

Every Data Channel Establishment Protocol message starts with a one byte field called "Message Type" which indicates the type of the message. The corresponding values are managed by IANA (see Section 8.2.1).

5.1. DATA_CHANNEL_OPEN Message

This message is sent initially on the Stream used for user messages using the Data Channel.



Message Type: 1 byte (unsigned integer)
This field holds the IANA defined message type for the DATA_CHANNEL_OPEN message. The value of this field is 0x03 as specified in Section 8.2.1.

Channel Type: 1 byte (unsigned integer)
This field specifies the type of the Data Channel to be opened and the values are managed by IANA (see Section 8.2.2):

DATA_CHANNEL_RELIABLE (0x00): The Data Channel provides a reliable in-order bi-directional communication.

DATA_CHANNEL_RELIABLE_UNORDERED (0x80): The Data Channel provides a reliable unordered bi-directional communication.

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT (0x01): The Data Channel provides a partially-reliable in-order bi-directional communication. User messages will not be retransmitted more times than specified in the Reliability Parameter.

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT_UNORDERED (0x81): The Data Channel provides a partial reliable unordered bi-directional communication. User messages will not be retransmitted more times than specified in the Reliability Parameter.

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED (0x02): The Data Channel provides a partial reliable in-order bi-directional communication. User messages might not be transmitted or retransmitted after a specified life-time given in milli-

seconds in the Reliability Parameter. This life-time starts when providing the user message to the protocol stack.

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED (0x82): The Data Channel provides a partial reliable unordered bi-directional communication. User messages might not be transmitted or retransmitted after a specified life-time given in milli-seconds in the Reliability Parameter. This life-time starts when providing the user message to the protocol stack.

Priority: 2 bytes (unsigned integer)
The priority of the Data Channel as described in [I-D.ietf-rtcweb-data-channel].

Reliability Parameter: 4 bytes (unsigned integer)
For reliable Data Channels this field MUST be set to 0 on the sending side and MUST be ignored on the receiving side. If a partial reliable Data Channel with limited number of retransmissions is used, this field specifies the number of retransmissions. If a partial reliable Data Channel with limited lifetime is used, this field specifies the maximum lifetime in milliseconds. The following table summarizes this:

Channel Type	Reliability Parameter
DATA_CHANNEL_RELIABLE	Ignored
DATA_CHANNEL_RELIABLE_UNORDERED	Ignored
DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT	Number of RTX
DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT_UNORDERED	Number of RTX
DATA_CHANNEL_PARTIAL_RELIABLE_TIMED	Lifetime in ms
DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED	Lifetime in ms

Label Length: 2 bytes (unsigned integer)
The length of the label field in bytes.

Protocol Length: 2 bytes (unsigned integer)
The length of the protocol field in bytes.

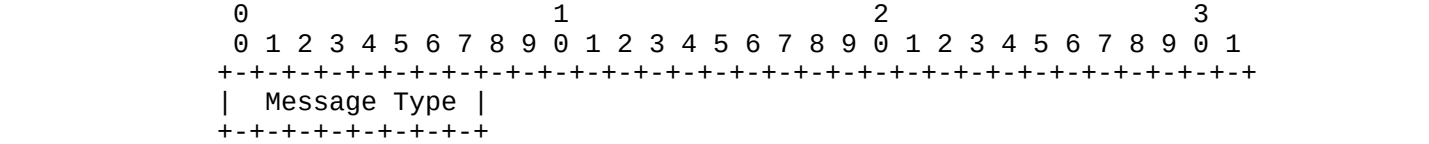
Label: Variable Length (sequence of characters)
The name of the Data Channel as a UTF-8 encoded string as specified in [RFC3629]. This may be an empty string.

Protocol: Variable Length (sequence of characters)
If this is an empty string the protocol is unspecified. If it is a non-empty string, it specifies a protocol registered in the

'WebSocket Subprotocol Name Registry' created in [RFC6455]. This string is UTF-8 encoded as specified in [RFC3629].

5.2. DATA_CHANNEL_ACK Message

This message is sent in response to a DATA_CHANNEL_OPEN_RESPONSE message on the stream used for user messages using the Data Channel. Reception of this message tells the opener that the Data Channel setup handshake is complete.



Message Type: 1 byte (unsigned integer)
This field holds the IANA defined message type for the DATA_CHANNEL_ACK message. The value of this field is 0x02 as specified in Section 8.2.1.

6. Procedures

All Data Channel Establishment Protocol messages MUST be sent using ordered delivery and reliable transmission. They MUST be sent on the same outgoing Stream as the user messages belonging to the corresponding Data Channel. Multiplexing and demultiplexing is done by using the SCTP payload protocol identifier (PPID). Therefore Data Channel Establishment Protocol message MUST be sent with the assigned PPID for the Data Channel Establishment Protocol (see Section 8.1). Other messages MUST NOT be sent using this PPID.

The peer that initiates opening a Data Channel selects a Stream Identifier for which the corresponding incoming and outgoing Streams are unused. If the side is the DTLS client, it MUST choose an even Stream Identifier, if the side is the DTLS server, it MUST choose an odd one. It fills in the parameters of the DATA_CHANNEL_OPEN message and sends it on the chosen Stream.

If a DATA_CHANNEL_OPEN message is received on an unused Stream, the Stream Identifier corresponds to the role of the peer and all parameters in the DATA_CHANNEL_OPEN message are valid, then a corresponding DATA_CHANNEL_ACK message is sent on the Stream with the same Stream Identifier as the one the DATA_CHANNEL_OPEN message was received on.

If the DATA_CHANNEL_OPEN message doesn't satisfy the conditions above, for instance if a DATA_CHANNEL_OPEN message is received on an

already used Stream or there are any problems with parameters within the DATA_CHANNEL_OPEN message, the odd/even rule is violated or the DATA_CHANNEL_OPEN message itself is not well-formed, the receiver MUST close the corresponding Data Channel using the procedure described in [I-D.ietf-rtcweb-data-channel] and MUST NOT send a DATA_CHANNEL_ACK message in response to the received message. Therefore, receiving an SCTP stream reset request for a Stream on which no DATA_CHANNEL_ACK message has been received indicates to the sender of the corresponding DATA_CHANNEL_OPEN message the failure of the Data Channel setup procedure. After also successfully resetting the corresponding outgoing Stream, which concludes the Data Channel closing initiated by the peer, a new DATA_CHANNEL_OPEN message can be sent on the Stream.

After the DATA_CHANNEL_OPEN message has been sent, the sender of the DATA_CHANNEL_OPEN MAY start sending messages containing user data without waiting for the reception of the corresponding DATA_CHANNEL_ACK message. However, before the DATA_CHANNEL_ACK message or any other message has been received on a Data Channel, all other messages containing user data and belonging to this Data Channel MUST be sent ordered, no matter whether the Data Channel is ordered or not. After the DATA_CHANNEL_ACK or any other message has been received on the Data Channel, messages containing user data MUST be sent ordered on ordered Data Channels and MUST be sent unordered on unordered Data Channels. Therefore receiving a message containing user data on an unused Stream indicates an error. The corresponding Data Channel MUST be closed as described in [I-D.ietf-rtcweb-data-channel].

7. Security Considerations

The DATA_CHANNEL_OPEN messages contains two variable length fields: the protocol and the label. A receiver must be prepared to receive DATA_CHANNEL_OPEN messages where these field have the maximum length of 65535 bytes. Error cases like the use of inconsistent lengths fields, unknown parameter values or violation the odd/even rule must also be handled by closing the corresponding Data Channel. An end-point must also be prepared that the peer open the maximum number of Data Channels.

This protocol does not provide privacy, integrity or authentication. It needs to be used as part of a protocol suite that contains all these things. Such a protocol suite is specified in [I-D.ietf-tsvwg-sctp-dtls-encaps].

For general considerations see [I-D.ietf-rtcweb-security] and [I-D.ietf-rtcweb-security-arch].

8. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

IANA is asked to update the reference of an already existing SCTP PPID assignment (Section 8.1) and to create a new standalone registry with its own URL for the DCEP (Section 8.2) containing two new registration tables (Section 8.2.1 and Section 8.2.2).

8.1. SCTP Payload Protocol Identifier

This document uses one already registered SCTP Payload Protocol Identifier (PPID) named "WebRTC Control". [RFC4960] creates the registry "SCTP Payload Protocol Identifiers" from which this identifier was assigned. IANA is requested to update the reference of this assignment to point to this document and to update the name. The corresponding date should be kept.

Therefore this assignment should be updated to read:

Value	SCTP PPID	Reference	Date
WebRTC DCEP	50	[RFCXXXX]	2013-09-20

8.2. New Standalone Registry for the DCEP

IANA is requested to create a new standalone registry (aka a webpage) with its own URL for the Data Channel Establishment Protocol (DCEP). The title should be "Data Channel Establishment Protocol (DCEP) Parameters". It will contain the two tables as described in Section 8.2.1 and Section 8.2.2.

8.2.1. New Message Type Registry

IANA is requested to create a new registration table "Message Type Registry" for the Data Channel Establishment Protocol (DCEP) to manage the one byte "Message Type" field in DCEP messages (see Section 5). This registration table should be part of the registry described in Section 8.2.

The assignment of new message types is done through an RFC required action, as defined in [RFC5226]. Documentation of the new message type MUST contain the following information:

1. A name for the new message type;

2. A detailed procedural description of the use of messages with the new type within the operation of the Data Channel Establishment Protocol.

Initially the following values need to be registered:

Name	Type	Reference
Reserved	0x00	[RFCXXXX]
Reserved	0x01	[RFCXXXX]
DATA_CHANNEL_ACK	0x02	[RFCXXXX]
DATA_CHANNEL_OPEN	0x03	[RFCXXXX]
Unassigned	0x04-0xfe	
Reserved	0xff	[RFCXXXX]

Please note that the values 0x00 and 0x01 are reserved to avoid interoperability problems, since they have been used in earlier versions of the document. The value 0xff has been reserved for future extensibility. The range of possible values is from 0x00 to 0xff.

8.2.2. New Channel Type Registry

IANA is requested to create a new registration table "Channel Type Registry" for the Data Channel Establishment Protocol to manage the one byte "Channel Type" field in DATA_CHANNEL_OPEN messages (see Section 5.1). This registration table should be part of the registry described in Section 8.2.

The assignment of new message types is done through an RFC required action, as defined in [RFC5226]. Documentation of the new Channel Type MUST contain the following information:

1. A name for the new Channel Type;

2. A detailed procedural description of the user message handling for Data Channels using this new Channel Type.

Please note that if new Channel Types support ordered and unordered message delivery, the high order bit MUST be used to indicate whether the message delivery is unordered or not.

Initially the following values need to be registered:

Name	Type	Reference
DATA_CHANNEL_RELIABLE	0x00	[RFCXXXX]
DATA_CHANNEL_RELIABLE_UNORDERED	0x80	[RFCXXXX]
DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT	0x01	[RFCXXXX]
DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT_UNORDERED	0x81	[RFCXXXX]
DATA_CHANNEL_PARTIAL_RELIABLE_TIMED	0x02	[RFCXXXX]
DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED	0x82	[RFCXXXX]
Reserved	0x7f	[RFCXXXX]
Reserved	0xff	[RFCXXXX]
Unassigned	rest	

Please note that the values 0x7f and 0xff have been reserved for future extensibility. The range of possible values is from 0x00 to 0xff.

9. Acknowledgments

The authors wish to thank Harald Alvestrand, Richard Barnes, Adam Bergkvist, Spencer Dawkins, Barry Dingle, Stefan Haekansson, Cullen Jennings, Paul Kyzivat, Doug Leonard, Alexey Melnikov, Pete Resnick, Irene Ruengeler, Randall Stewart, Peter Thatcher, Martin Thompson, Justin Uberti, and many others for their invaluable comments.

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.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

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

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[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-07 (work in progress), December 2014.

[I-D.ietf-rtcweb-data-channel] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-12 (work in progress), September 2014.

10.2. Informational References

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

[I-D.ietf-rtcweb-security] Rescorla, E., "Security Considerations for WebRTC", draft-ietf-rtcweb-security-07 (work in progress), July 2014.

[I-D.ietf-rtcweb-security-arch] Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-10 (work in progress), July 2014.

Authors' Addresses

Randell Jesup
Mozilla
US

Email: randell-ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: salvatore.loreto@ericsson.com

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
Steinfurt 48565
DE

Email: tuexen@fh-muenster.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 28, 2015

M. Tuexen
Muenster Univ. of Appl. Sciences
R. Stewart
Netflix, Inc.
R. Jesup
WorldGate Communications
S. Loreto
Ericsson

January 24, 2015

DTLS Encapsulation of SCTP Packets
draft-ietf-tsvwg-sctp-dtls-encaps-09.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a transport protocol originally defined to run on top of the network protocols IPv4 or IPv6. This document specifies how SCTP can be used on top of the Datagram Transport Layer Security (DTLS) protocol. Using the encapsulation method described in this document, SCTP is unaware of the protocols being used below DTLS; hence explicit IP addresses cannot be used in the SCTP control chunks. As a consequence, the SCTP associations carried over DTLS can only be single homed.

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 July 28, 2015.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Overview 2
2. Conventions 3
3. Encapsulation and Decapsulation Procedure 3
4. General Considerations 3
5. DTLS Considerations 4
6. SCTP Considerations 5
7. IANA Considerations 6
8. Security Considerations 6
9. Acknowledgments 7
10. References 7
Appendix A. NOTE to the RFC-Editor 9
Authors' Addresses 9

1. Overview

The Stream Control Transmission Protocol (SCTP) as defined in [RFC4960] is a transport protocol running on top of the network protocols IPv4 [RFC0791] or IPv6 [RFC2460]. This document specifies how SCTP is used on top of the Datagram Transport Layer Security (DTLS) protocol. DTLS 1.0 is defined in [RFC4347] and the latest version when this RFC was published, DTLS 1.2, is defined in [RFC6347]. This encapsulation is used for example within the WebRTC protocol suite (see [I-D.ietf-rtcweb-overview] for an overview) for transporting non-SRTP data between browsers. The architecture of this stack is described in [I-D.ietf-rtcweb-data-channel].

[NOTE to RFC-Editor:

Please ensure that the authors double check the above statement about DTLS 1.2 during AUTH48 and then remove this note before publication.

]

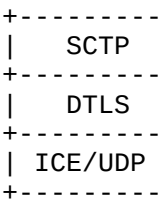


Figure 1: Basic stack diagram

This encapsulation of SCTP over DTLS over UDP or ICE/UDP (see [RFC5245]) can provide a NAT traversal solution in addition to confidentiality, source authentication, and integrity protected transfers. Please note that using ICE does not necessarily imply that a different packet format is used on the wire.

Please note that the procedures defined in [RFC6951] for dealing with the UDP port numbers do not apply here. When using the encapsulation defined in this document, SCTP is unaware about the protocols used below DTLS.

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 [RFC2119].

3. Encapsulation and Decapsulation Procedure

When an SCTP packet is provided to the DTLS layer, the complete SCTP packet, consisting of the SCTP common header and a number of SCTP chunks, is handled as the payload of the application layer protocol of DTLS. When the DTLS layer has processed a DTLS record containing a message of the application layer protocol, the payload is passed to the SCTP layer. The SCTP layer expects an SCTP common header followed by a number of SCTP chunks.

4. General Considerations

An implementation of SCTP over DTLS MUST implement and use a path maximum transmission unit (MTU) discovery method that functions without ICMP to provide SCTP/DTLS with an MTU estimate. An implementation of "Packetization Layer Path MTU Discovery" [RFC4821] either in SCTP or DTLS is RECOMMENDED.

The path MTU discovery is performed by SCTP when SCTP over DTLS is used for data channels (see Section 5 of [I-D.ietf-rtcweb-data-channel]).

5. DTLS Considerations

The DTLS implementation MUST support DTLS 1.0 [RFC4347] and SHOULD support the most recently published version of DTLS, which was DTLS 1.2 [RFC6347] when this RFC was published. In the absence of a revision to this document, the latter requirement applies to all future versions of DTLS when they are published as RFCs. This document will only be revised if a revision to DTLS or SCTP makes a revision to the encapsulation necessary.

[NOTE to RFC-Editor:

Please ensure that the authors double check the above statement about DTLS 1.2 during AUTH48 and then remove this note before publication.

]

SCTP performs segmentation and reassembly based on the path MTU. Therefore the DTLS layer MUST NOT use any compression algorithm.

The DTLS MUST support sending messages larger than the current path MTU. This might result in sending IP level fragmented messages.

If path MTU discovery is performed by the DTLS layer, the method described in [RFC4821] MUST be used. For probe packets, the extension defined in [RFC6520] MUST be used.

If path MTU discovery is performed by the SCTP layer and IPv4 is used as the network layer protocol, the DTLS implementation SHOULD allow the DTLS user to enforce that the corresponding IPv4 packet is sent with the Don't Fragment (DF) bit set. If controlling the DF bit is not possible, for example due to implementation restrictions, a safe value for the path MTU has to be used by the SCTP stack. It is RECOMMENDED that the safe value does not exceed 1200 bytes. Please note that [RFC1122] only requires end hosts to be able to reassemble fragmented IP packets up to 576 bytes in length.

The DTLS implementation SHOULD allow the DTLS user to set the Differentiated services code point (DSCP) used for IP packets being sent (see [RFC2474]). This requires the DTLS implementation to pass the value through and the lower layer to allow setting this value. If the lower layer does not support setting the DSCP, then the DTLS user will end up with the default value used by protocol stack. Please note that only a single DSCP value can be used for all packets belonging to the same SCTP association.

Using explicit congestion notifications (ECN) in SCTP requires the DTLS layer to pass the ECN bits through and its lower layer to expose access to them for sent and received packets (see [RFC3168]). The implementation of DTLS and its lower layer have to provide this support. If this is not possible, for example due to implementation restrictions, ECN can't be used by SCTP.

6. SCTP Considerations

This section describes the usage of the base protocol and the applicability of various SCTP extensions.

6.1. Base Protocol

This document uses SCTP [RFC4960] with the following restrictions, which are required to reflect that the lower layer is DTLS instead of IPv4 and IPv6 and that SCTP does not deal with the IP addresses or the transport protocol used below DTLS:

- A DTLS connection MUST be established before an SCTP association can be set up.

- Multiple SCTP associations MAY be multiplexed over a single DTLS connection. The SCTP port numbers are used for multiplexing and demultiplexing the SCTP associations carried over a single DTLS connection.

- All SCTP associations are single-homed, because DTLS does not expose any address management to its upper layer. Therefore it is RECOMMENDED to set the SCTP parameter path.max.retrans to association.max.retrans.

- The INIT and INIT-ACK chunk MUST NOT contain any IPv4 Address or IPv6 Address parameters. The INIT chunk MUST NOT contain the Supported Address Types parameter.

- The implementation MUST NOT rely on processing ICMP or ICMPv6 packets, since the SCTP layer most likely is unable to access the SCTP common header in the plain text of the packet, which triggered the sending of the ICMP or ICMPv6 packet. This applies in particular to path MTU discovery when performed by SCTP.

- If the SCTP layer is notified about a path change by its lower layers, SCTP SHOULD retest the Path MTU and reset the congestion state to the initial state. The window-based congestion control method specified in [RFC4960], resets the congestion window and slow start threshold to their initial values.

6.2. Padding Extension

When the SCTP layer performs path MTU discovery as specified in [RFC4821], the padding extension defined in [RFC4820] MUST be supported and used for probe packets (HEARTBEAT chunks bundled with PADDING chunks [RFC4820]).

6.3. Dynamic Address Reconfiguration Extension

If the dynamic address reconfiguration extension defined in [RFC5061] is used, ASCONF chunks MUST use wildcard addresses only.

6.4. SCTP Authentication Extension

The SCTP authentication extension defined in [RFC4895] can be used with DTLS encapsulation, but does not provide any additional benefit.

6.5. Partial Reliability Extension

Partial reliability as defined in [RFC3758] can be used in combination with DTLS encapsulation. It is also possible to use additional PR-SCTP policies, for example the ones defined in [I-D.ietf-tsvwg-sctp-prpolicies].

6.6. Stream Reset Extension

The SCTP stream reset extension defined in [RFC6525] can be used with DTLS encapsulation. It is used to reset SCTP streams and add SCTP streams during the lifetime of the SCTP association.

6.7. Interleaving of Large User Messages

SCTP as defined in [RFC4960] does not support the interleaving of large user messages that need to be fragmented and reassembled by the SCTP layer. The protocol extension defined in [I-D.ietf-tsvwg-sctp-ndata] overcomes this limitation and can be used with DTLS encapsulation.

7. IANA Considerations

This document requires no actions from IANA.

8. Security Considerations

Security considerations for DTLS are specified in [RFC4347] and for SCTP in [RFC4960], [RFC3758], and [RFC6525]. The combination of SCTP and DTLS introduces no new security considerations.

SCTP should not process the IP addresses used for the underlying communication since DTLS provides no guarantees about them.

It should be noted that the inability to process ICMP or ICMPv6 messages does not add any security issue. When SCTP is carried over a connection-less lower layer like IPv4, IPv6, or UDP, processing of these messages is required to protect other nodes not supporting SCTP. Since DTLS provides a connection-oriented lower layer, this kind of protection is not necessary.

9. Acknowledgments

The authors wish to thank David Black, Benoit Claise, Spencer Dawkins, Francis Dupont, Gorry Fairhurst, Stephen Farrell, Christer Holmberg, Barry Leiba, Eric Rescorla, Tom Taylor, Joe Touch and Magnus Westerlund for their invaluable comments.

10. References

10.1. Normative References

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, March 2007.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[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.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.

10.2. Informative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, August 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.

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

[RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, February 2012.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013.

[I-D.ietf-rtcweb-overview] Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-13 (work in progress), November 2014.

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ietf-tsvwg-sctp-prpolicies]
Tuexen, M., Seggelmann, R., Stewart, R., and S. Loreto, "Additional Policies for the Partial Reliability Extension of the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-prpolicies-06 (work in progress), December 2014.

[I-D.ietf-tsvwg-sctp-ndata]
Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "Stream Schedulers and a New Data Chunk for the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-ndata-02 (work in progress), January 2015.

Appendix A. NOTE to the RFC-Editor

Although the references to [I-D.ietf-tsvwg-sctp-prpolicies] and [I-D.ietf-tsvwg-sctp-ndata] are informative, put this document in REF-HOLD until these two references have been approved and update these references to the corresponding RFCs.

Authors' Addresses

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
DE

Email: tuexen@fh-muenster.de

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
US

Email: randall@lakerest.net

Randell Jesup
WorldGate Communications
3800 Horizon Blvd, Suite #103
Trevose, PA 19053-4947
US

Phone: +1-215-354-5166
Email: randell_ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: Salvatore.Loreto@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

P. Kyzivat
L. Xiao
C. Groves
Huawei
R. Hansen
Cisco Systems

February 14, 2014

CLUE Signaling
draft-kyzivat-clue-signaling-07

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.holmberg-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 August 18, 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. CLUE call establishment 5
3.1. Establishment of the CLUE data channel 5
3.2. Initial media transmission 5
3.3. Interoperability with non-CLUE devices 6
3.4. Mid-call changes to CLUE status 6
4. CLUE use of SDP O/A 6
4.1. Signalling CLUE Encodings 6
4.1.1. Alternate encoding limit syntaxes 7
4.2. Negotiating receipt of CLUE capture encodings in SDP . . . 7
4.3. Signaling CLUE control of "m" lines 7
4.4. Media line directionality 8
4.5. Multiplexing CLUE media lines 8
5. Interaction of CLUE protocol and SDP negotiations 9
5.1. Independence of SDP and CLUE negotiation 9
5.2. Recommendations for operating with non-atomic operations 9
5.3. Constraints on sending media 10
6. Example: A call between two CLUE-capable endpoints 10
7. Example: A call between a CLUE-capable and non-CLUE endpoint 18
8. CLUE requirements on SDP O/A 19
9. SIP Signaling 19
10. CLUE over RTCWEB 20
11. Open Issues 20
12. What else? 20
13. Acknowledgements 20
14. IANA Considerations 20
15. Security Considerations 20
16. Change History 21
17. References 22
17.1. Normative References 22
17.2. Informative References 23
Appendix A. CLUE Signalling and data channel concerns 24
A.1. Protocol Versioning and Options 24
A.1.1. Versioning Objectives 24
A.1.2. Versioning Overview 24
A.1.3. Version Negotiation 26
A.1.4. Option Negotiation 27
A.1.5. Option Elements 27

A.1.5.1. <mediaProvider> 28
A.1.6. Version & option negotiation errors 28
A.1.7. Definition and Use of Version Numbers 29
A.1.8. Version & Option Negotiation Examples 30
A.1.8.1. Successful Negotiation - Multi-version 30
A.1.8.2. Successful Negotiation - Consumer-Only Endpoint . 32
A.1.8.3. Successful Negotiation - Provider-Only Endpoint . 33
A.1.8.4. Version Incompatibility 33
A.1.8.5. Option Incompatibility 34
A.1.8.6. Syntax Error 35
A.2. Message Transport 35
A.2.1. CLUE Channel Lifetime 35
A.2.2. Channel Error Handling 36
A.3. Message Framing 36
Authors' Addresses 36

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.holmberg-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.

This document originally also defined the CLUE protocol itself. These details have mostly been split out into [I-D.presta-clue-protocol] and expanded, but at present some details remain in this document.

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.holmberg-clue-datachannel] for more details..

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

CLUE-enabled device: A CLUE-capable device that wishes to negotiate a CLUE data channel and send and/or receive CLUe-controlled media.

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

3. CLUE call establishment

3.1. Establishment of the CLUE data channel

The CLUE data channel [I-D.holmberg-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.

Presence of the CLUE data channel in an SDP offer or answer also served as an indication that the device supports CLUE and wishes to upgrade the call to include CLUE-controlled media. A CLUE-enabled device SHOULD include an "m" line for the CLUE channel in its initial SDP offer, and SHOULD include an "m" line in subsequent offers and answers, when allowed by [RFC3264].

In cases where both devices in an SDP negotiation are CLUE-enabled and include an "m" line for the data channel, see [I-D.holmberg-clue-datachannel] for negotiation details. If negotiation is successful, the call is now considered CLUE-enabled, and sending of CLUE protocol [I-D.presta-clue-protocol] messages can begin.

3.2. Initial media transmission

In the event that the CLUE data channel is successfully negotiated, a CLUE-enabled 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 negotiated. However, a CLUE-enabled device MUST still be prepared to receive media on non-CLUE-controlled media

lines as defined in [RFC3264].

3.3. Interoperability with non-CLUE devices

A CLUE-enabled 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.

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

3.4. Mid-call changes to CLUE status

A CLUE-enabled device that receives an initial SDP offer from a non-CLUE device with no CLUE data channel "m" line SHOULD include a new data channel "m" line in any subsequent offers it sends, to indicate that it is CLUE-enabled.

If, in an ongoing non-CLUE call, one or both sides of the call subsequently add the CLUE data channel "m" line to their SDP and the CLUE data channel is then negotiated successfully the call is then considered CLUE-enabled, and sending of CLUE protocol [I-D.presta-clue-protocol] messages can begin.

If, in an ongoing CLUE-enabled call, an SDP offer-answer negotiation completes in a fashion in which the CLUE data channel is no longer active, the call is no longer considered CLUE-enabled. Devices in the call must revert to non-CLUE behaviour or terminate the call.

4. CLUE use of SDP O/A

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.

Every "m" line representing a CLUE encoding SHOULD 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.

4.1.1. Alternate encoding limit syntaxes

Note that while the expressing of CLUE encoding limits in SDP has been discussed at some length by the working group and it has been agreed that this is the current, working assumption, formal consensus has not been agreed on this. Alternatives include placing encoding limits in the CLUE ADVERTISEMENT message, or by using alternate SDP syntax, such as is suggested in [I-D.groves-clue-latent-config].

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. As well as the normal restrictions defined in [RFC3264] media MUST NOT be sent on this stream until the sender has received a valid CLUE CONFIGURE message specifying the capture to be used for this stream.

4.3. Signaling CLUE control of "m" lines

In many cases an implementation may wish to mix media channels that are under CLUE control with those that are not. It may want to ensure that there are non-CLUE streams for purposes of interoperability, or that can provide media from the start of the call before CLUE negotiation completes, or because the implementation wants CLUE-controlled video but traditional audio, or for any other reasons.

Which "m" lines in an SDP body are under control of the CLUE channel is signalled via the SDP Grouping Framework [RFC5888]. Devices that wish to negotiate CLUE MUST support the grouping framework.

A new semantic for the "group" session-level attribute, "CLUE", is used to signal which "m" lines are under the control of a CLUE channel. As per the framework, all of the "m" lines of a session description that uses "group" MUST be identified with a "mid" attribute whether they are controlled by CLUE or not. The "mid" id of any "m" lines controlled by a CLUE channel MUST be included in the "CLUE" group attribute alongside the "mid" id of the CLUE channel controlling them.

The CLUE group MUST NOT include more than one "m" line for a CLUE data channel. If a CLUE data channel is part of the CLUE group

attribute other media "m" lines included in the group are under the control of that CLUE channel; media MUST NOT be sent or received on these "m" lines until the CLUE channel has been negotiated and media has been negotiated via the CLUE protocol. If no CLUE data channel is part of the CLUE group attribute then media MUST NOT be sent or received on these "m" lines.

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

An SDP MAY include more than one group attribute with the "CLUE" semantic. An "mid" id for a given "m" line MUST NOT be included in more than one CLUE group.

4.4. Media line directionality

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

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

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

4.5. Multiplexing CLUE media lines

There is a desire in many use-cases to be able to multiplex multiple RTP streams onto a single port. However, the syntax for doing this in a CLUE or a generic MMUSIC fashion has not yet been determined.

Because there will always also be a need for non-multiplexed operation, the decision was made to move forward with non-multiplexed syntax, and add multiplexing capabilities when syntax for that has been defined.

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 complicated state machines with the potential to reach invalid states if messages were to be lost, or be rewritten en-route by middle boxes, the current proposal is that SDP and CLUE messages are independent. 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.

CLUE messages contain an EncodingID which is used to identify a specific encoding 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 EncodingID which does not currently match an extant id in SDP.

5.2. Recommendations for operating with non-atomic operations

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.

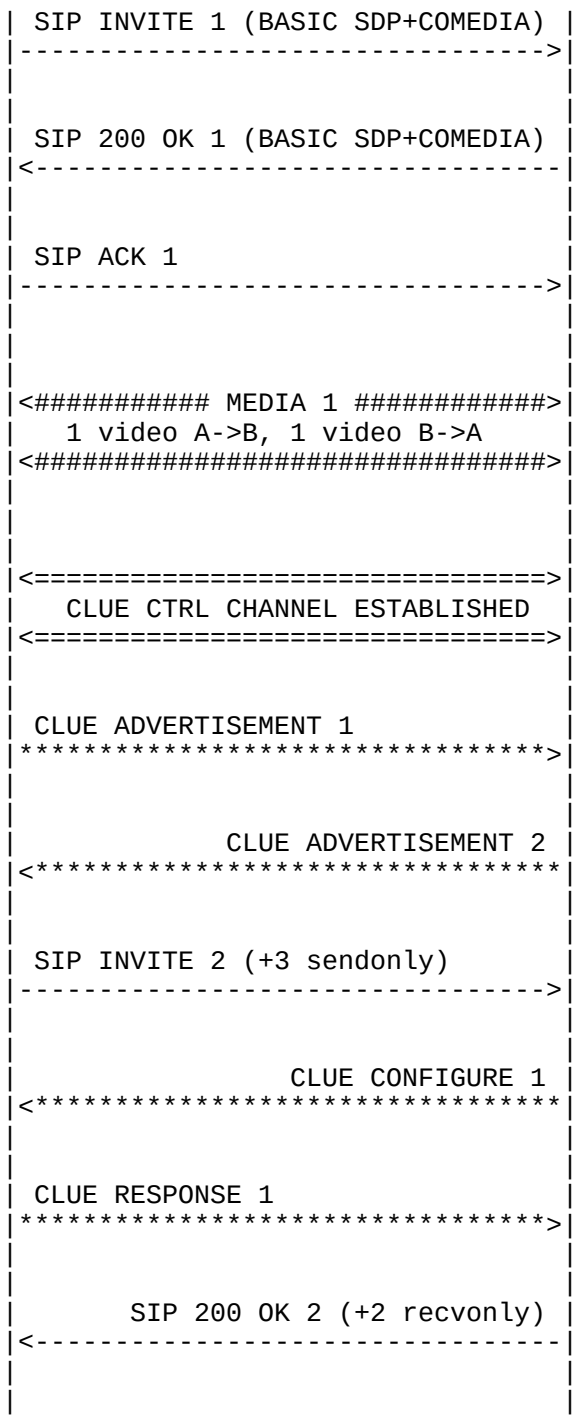
5.3. Constraints on sending media

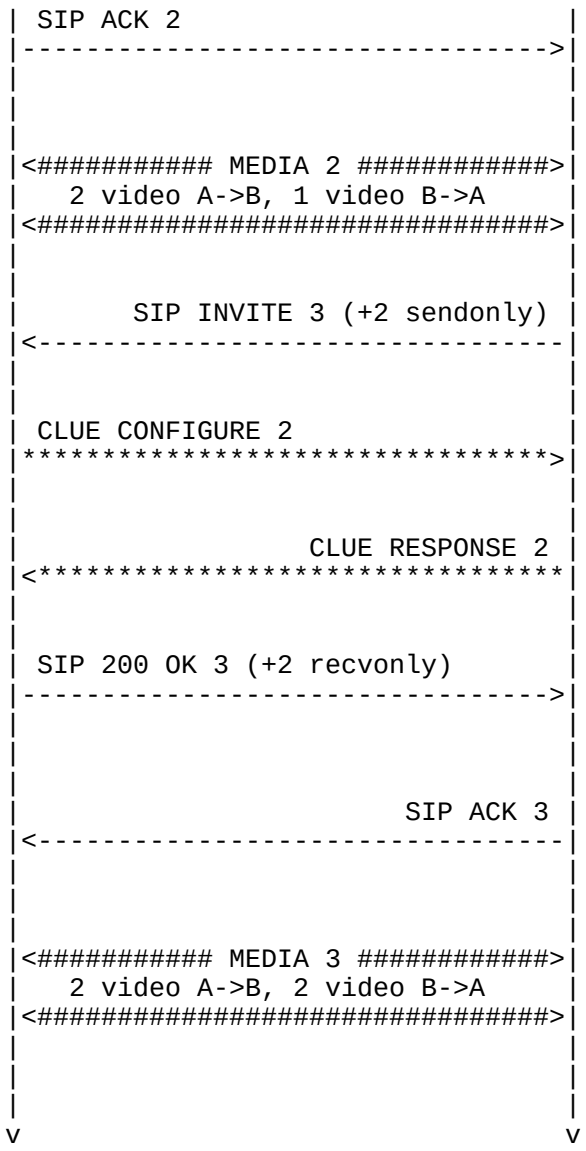
While SDP and CLUE message states do not impose constraints on each other, both impose constraints on the sending of media - media MUST NOT be sent unless it has been negotiated in both CLUE and SDP: an implementation MUST NOT send a specific CLUE capture encoding unless its most recent SDP exchange 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.

6. 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.


SIP INVITE 1 (BASIC SDP+COMEDIA)
SIP 200 OK 1 (BASIC SDP+COMEDIA)
SIP ACK 1
MEDIA 1
1 video A->B, 1 video B->A
CLUE CTRL CHANNEL ESTABLISHED
CLUE ADVERTISEMENT 1
CLUE ADVERTISEMENT 2
SIP INVITE 2 (+3 sendonly)
CLUE CONFIGURE 1
CLUE RESPONSE 1
SIP 200 OK 2 (+2 recvonly)



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

represents the CLUE channel):

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

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

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

Alice now sends her CLUE Advertisement (ADVERTISEMENT 1). She advertises three static captures representing her three cameras. She also includes switched captures suitable for two- and one-screen systems. All of these captures are in a single capture scene, with suitable capture scene entries to tell Bob that he should either subscribe to the three static captures, the two switched capture view or the one switched capture view. Alice has no simultaneity constraints, so includes all six captures in one simultaneous set. Finally, Alice includes an encoding group with three encoding IDs: "enc1", "enc2" and "enc3". These encoding ids aren’t currently valid, but will match the next SDP offer she sends.

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

Bob also sends his CLUE ADVERTISEMENT (ADVERTISEMENT 2). He advertises two static captures representing his cameras. He also includes a single composed capture for single-screen systems, in which he will composite the two camera views into a single video stream. All three captures are in a single capture scene, with suitable capture scene entries to tell Alice that she should either

subscribe to the two static captures, or the single composed capture. Bob also has no simultaneity constraints, so includes all three captures in one simultaneous set. Bob also includes a single encoding group with two encoding IDs: "foo" and "bar".

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

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

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

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

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

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

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

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

The constraints of offer/answer meant that Bob could not include his encoder information as new m-lines in 200 OK 2. As such Bob now sends INVITE 3 to generate a new offer. Along with all the streams

from 200 OK 2 Bob also includes two new sendonly streams. Each stream has a label corresponding to the encoding ids in his ADVERTISEMENT 2 message. He also adds their mid values to the grouping attribute to show they are controlled by the CLUE channel. A snippet of the SDP showing the grouping attribute and the video m-lines are shown below (mid 100 represents the CLUE channel):

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

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

Bob receives Alice’s message CONFIGURE 2 and sends RESPONSE 2 to ack its receptions. Bob does not yet send the capture encodings

specified, because Alice hasn't yet negotiated the ability to receive these streams in SDP.

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

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

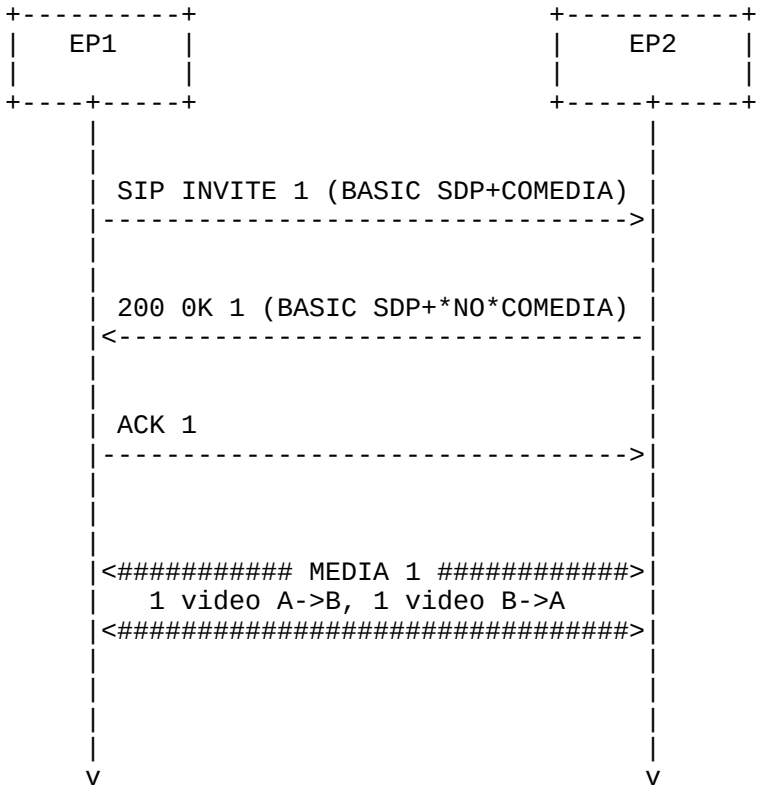
Finally, on receiving 200 OK 3 Bob is now able to send the two streams of video Alice requested - this is illustrated as MEDIA 3.

Both sides of the call are now sending multiple video streams with

their sources defined via CLUE negotiation. As the call progresses either side can send new ADVERTISEMENT or CONFIGURE or new SDP negotiation to add, remove or change what they have available or want to receive.

7. 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.

8. CLUE requirements on SDP O/A

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

Any other SDP extensions required to support CLUE signaling should also be specified here. Then we will need to take action within MMUSIC to make those happen. This section should be empty and removed before this document becomes an RFC.

NOTE: The RTP mapping document [I-D.even-clue-rtp-mapping] is also likely to call for SDP extensions. We will have to reconcile how to coordinate these two documents.

9. SIP Signaling

(Placeholder) This may be unremarkable. If so we can drop it.

10. CLUE over RTCWEB

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

11. 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.

- While the preference is to multiplex multiple capture encodings over a single RTP session, this will not always be desirable or possible. The factors that prevent multiplexing may come from either the provider or the consumer. So the extent of multiplexing must be negotiated. The decision about how to multiplex affects the number and grouping of m-lines in the SDP. The endpoint of a CLUE session that sends an offer needs to know the mapping of capture encodings to m-lines for both sides.

  AFAIK this issue hasn’t yet been considered at all.
- 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]

12. What else?

13. Acknowledgements

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

Christian Groves has contributed detailed comments and suggestions.

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

14. IANA Considerations

TBD

15. Security Considerations

TBD

16. Change History

-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.holmberg-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.holmberg-clue-datachannel] and [I-D.presta-clue-protocol], but that has not yet been resolved in those documents, into an appendix.

-06: Revisions by Rob Hansen
- Removed CLUE message XML schema and details that are now in draft-presta-clue-protocol
- Encoding limits in SDP section updated to note that this has been investigated and discussed and is the current working assumption of the WG, though consensus has not been fully achieved.
- A section has also been added on the current mandation of unidirectional "m"-lines.
- Updated CLUE messaging in example call flow to match draft-presta-clue-protocol-03

-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.)

-04: Revisions by Rob Hansen: ???

-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.

-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.

-01: Updated by roberta to include some sample call flows.

-00: Initial version by pkyzivat. Established general outline for the document, and specified a few things thought to represent wg consensus.

17. References

17.1. 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-14 (work in progress), February 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-03 (work in progress), November 2013.

[I-D.holmberg-clue-datachannel] Holmberg, C., "CLUE Protocol Data Channel", draft-holmberg-clue-datachannel-03 (work in progress), February 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.

17.2. Informative References

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

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

[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.

Appendix A. CLUE Signalling and data channel concerns

[The specifics of the CLUE signaling protocol are in the process of being defined in [I-D.presta-clue-protocol], while the negotiation of the CLUE data channel is being defined in [I-D.holmberg-clue-datachannel]. As such, considerable text originally in this section have been transitioned to these document. The following text relates to issues that are no longer the focus of this document, but remain important and unresolved, and so have been preserved here.]

A.1. Protocol Versioning and Options

A.1.1. Versioning Objectives

The CLUE versioning mechanism addresses the following needs:

- Coverage:
  - Versioning of basic behavior and options,
  - CLUE message exchange,
  - CLUE message exchange,
  - coordinated use of SIP and SDP,
  - required media behavior.
- Remain fixed for the duration of the CLUE channel
- Be extensible for configuration of new options.
- Be sufficient (with extensions) for all envisioned future versions.

A.1.2. Versioning Overview

An initial message exchange on the CLUE channel handles the negotiation of version and options.

- Dedicated message types are used for this negotiation.
- The negotiation is repeated if the CLUE channel is reestablished.

The version usage is similar in philosophy to XMPP:

- See [RFC6120] section 4.7.5.
- A version 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.
- If a common major version cannot be negotiated, then CLUE MUST NOT be used.

- o The same message exchange also negotiates options.
- o Each option is denoted by a unique XML element in the negotiation.

Figure 1 shows the negotiation in simplified form:

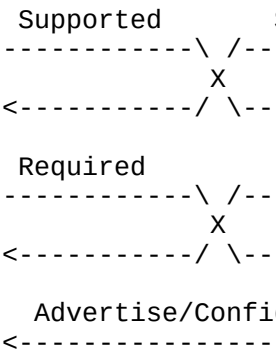


Figure 1: Basic Option Negotiation (simplified)

Dedicated message types are used for the negotiation because:

- o The protocol can then ensure that the negotiation is done first, and once. Not changing mid-session means an endpoint can plan ahead, and predict what may be used and what might be received.
- o This provides extensible framework for negotiating optional features.
- o A full option negotiation can be completed before other messages are exchanged.

Figure 2 and Figure 3 are simplified examples of the Supported and Required messages:

```
<supported>
        <version major="1" minor="0">
          <!- May repeat version if multiple
         major versions supported.      ->
   <!- Options follow ->
   <mediaProvider/>
        ...
</supported>
```

Figure 2: Supported Message (simplified)

```
<required>
        <version major="1" minor="0">
          <!- Requested options of peer follow ->
   <!- Options follow ->
   <mediaProvider/>
        ...
</required>
```

Figure 3: Required Message (simplified)

A.1.3. Version Negotiation

The Supported message includes one or more <version> elements, each denoting a major/minor version combination that the sender of the message is capable of supporting.

The <version> element contains both a major and minor version. Each is a non-negative integer. Each <version> element in the message MUST contain a unique major version number, distinct from the major version number in all the other <version> elements in the message. The minor version in a <version> element denotes the largest minor version the sender supports for the corresponding major version. (Minor versions are always backwards compatible, so support for a minor version implies support for all smaller minor versions.)

Each endpoint of the CLUE channel sends a Supported message, and receives the Supported message sent by the other end. Then each end compares the versions sent and the versions received to determine the version to be used for this CLUE session.

- If there is no major version in common between the two ends, negotiation fails.
- The <version> elements from the two ends that have the largest matching major version are selected.
- After exchange each end determines compatible version numbers to be used for encoding and decoding messages, and other behavior in the CLUE session.
  - The <version> elements from the two ends that have the largest matching major version are selected.
  - The side that sent the smaller minor version chooses the one it sent.
  - The side that sent the larger minor version may choose the minor version it received, or the one it sent, or any value between those two.
- Each end then sends a Required message with a single <version> element containing the major and minor versions it has chosen.

  [[Note: "required" is the wrong semantic for this. Might want a

better message name.]]

- o Each end then behaves in accord with the specifications denoted by the version it chose. This continues until the end of the CLUE session, or until changed as a result of another version negotiation when the CLUE channel is reestablished.

  [[Note: The version negotiation remains in effect even if the CLUE channel is lost.]]

A.1.4. Option Negotiation

Option negotiation is used to agree upon which options will be available for use within the CLUE session. (It does not say that these options must be used.) This may be used for both standard and proprietary options. (As used here, and option could be either a feature described as part of this specification that is optional to implement, or a feature defined in a separate specification that extends this one.)

Each end includes, within the Supported message it sends, elements describing those options it is willing and able to use with this CLUE session.

Each side, upon receiving a Supported message, selects from that message those option elements that it wishes the peer to use. (If/ when occasion for that use arises.) It then includes those selected elements into the Required message that it sends.

Within a received Supported message, unknown option elements MUST be ignored. This includes elements that are of a known type that is not known to denote an option.

A.1.5. Option Elements

Each option is denoted, in the Supported and Required messages, by an XML element. There are no special rules for these elements - they can be any XML element. The attributes and body of the element may carry further information about the option. The same element type is used to denote the option in the Supported message and the corresponding Required message, but the attributes and body may differ according to option-specific rules. This may be used to negotiate aspects of a particular option. The ordering of option elements is irrelevant within the Supported and Required messages, and need not be consistent in the two.

Only one option element is defined in this document: <mediaProvider>.

A.1.5.1. <mediaProvider>

The <mediaProvider> element, when placed in a Supported message, indicates that the sender is willing and able to send ADVERTISEMENT messages and receive CONFIGURE messages. When placed in a Required message, the <mediaProvider> element indicates that the sender is willing, able, and desirous of receiving ADVERTISEMENT messages and sending CONFIGURE messages. If an endpoint does not receive <mediaProvider> in a Required message, it MUST NOT send ADVERTISEMENT messages. For common cases <mediaProvider> should be supported and required by both endpoints, to enable bidirectional exchange of media. If not required by either end, the CLUE session is useless. This is an error condition, and SHOULD result in termination of the CLUE channel.

The <mediaProvider> element has no defined attributes or body.

A.1.6. Version & option negotiation errors

The following are errors that may be detected and reported during version negotiation:

- Version incompatibility

  There is no common value between the major version numbers sent in a Supported message and those in the received Supported message.

- Option incompatibility

  This can occur if options supported by one endpoint are inconsistent with those supported by the other endpoint. E.g., The <mediaProvider> option is not specified by either endpoint. Options SHOULD be specified so as to make it difficult for this problem to occur.

  This error may also be used to indicate that insufficient options have been required among the two ends for a useful session to result. This can occur with a feature that needs to be present on at least one end, but not on a specific end. E.g., The <mediaProvider> option was Supported by at least one of the endpoints, but it was not Required by either.

  This may also be used to indicate that an option element in the Required message has attributes or body content that is syntactically correct, but in inconsistent with the rules for option negotiation specified for that particular element. The definition of each option must specify the negotiation rules for that option.

- Unsupported option

  An option element type received in a Required message did not appear in the corresponding Supported element.

  (Unsupported options received in a Supported message do not trigger this error. They are ignored.)

These errors are reported using the normal message error reporting mechanism.

Other applicable error codes may also be returned in response to a Supported or Required message.

Errors that occur at this stage result in negotiation failure. When this occurs, CLUE cannot be used until the end of the SIP session, or until a new CLUE channel is negotiated and a subsequent version negotiation succeeds. The SIP session may continue without CLUE features.

A.1.7. Definition and Use of Version Numbers

[[NOTE: THIS IS AWKWARD. SUGGESTIONS FOR BETTER WAYS TO DEFINE THIS ARE WELCOME.]]

This document defines CLUE version 1.0 (major=1, minor=0). This denotes the normative behavior defined in this document and other documents upon which it normatively depends, including but is not limited to:

- the schema defined in [I-D.presta-clue-protocol];
- the schema defined in [clue-data-model];
- the protocol used to exchange CLUE messages;
- the protocol defined herein that defines valid sequence of CLUE messages;
- the specific rules defined herein for employing SIP, SDP, and RTP to realize the CLUE messages.

Given two CLUE versions Vx and Vy, then Vx is backward compatible with Vy if and only if:

- All messages valid according to the schema of Vx are also valid according to the schemas of Vy
- All messages valid according to the schema of Vy can be made valid according to the schemas of Vx by deleting elements undefined in the schemas of Vx.

  [[NOTE: THIS PROBABLY NEEDS WORK!]]

- o All normative behaviors defined for Vx are defined consistently for Vy.

  [[NOTE: SOME HAND WAVING HERE.]]

Revisions, updates, to any of the documents denoted by Version 1.0 MAY result in the definition of a new CLUE version. If they do, then this document MUST be revised to define the new version.

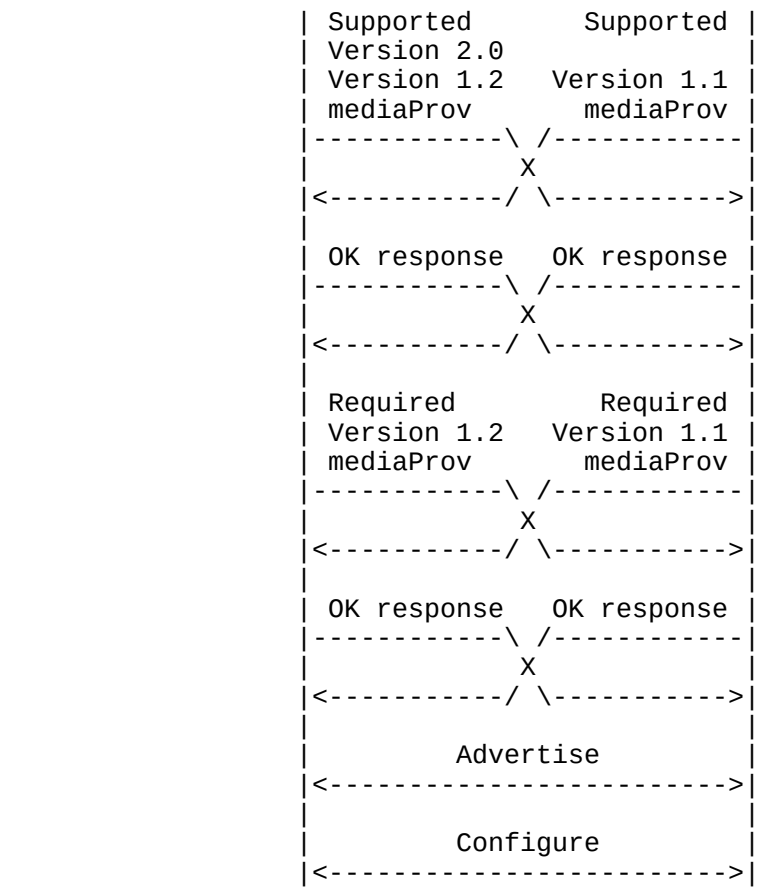
The CLUE version to be defined in a revision to this document MUST be determined as follows:

- o If the revision and the document being revised are mutually backward compatible (they are functionally equivalent), then the CLUE version MUST remain unchanged.
- o Else if the revision is backward compatible with the document being revised, then the CLUE major version MUST remain unchanged, and the CLUE minor version MUST be increased by one (1).
- o Else the CLUE major version must be increased by one (1), and the CLUE minor version set to zero (0).

When a CLUE implementation sends a Supported message, it MUST include the CLUE versions it is willing and able to conform with.

A.1.8. Version & Option Negotiation Examples

A.1.8.1. Successful Negotiation - Multi-version



The endpoint on the left can support versions 1.2 and 2.0, and because of backward compatibility can support versions 1.0 and 1.1. The endpoint on the right supports only version 2.0. Both endpoints with to both provide and consume media. They each send a Supported message indicating what they support.

The element on the left, upon receiving the Supported message, determines that it is permitted to use version 1.2 or 1.1, and decides to use 1.2. It sends a Required message containing version 1.2 and also includes the mediaProvider option element, because it wants its peer to provide media.

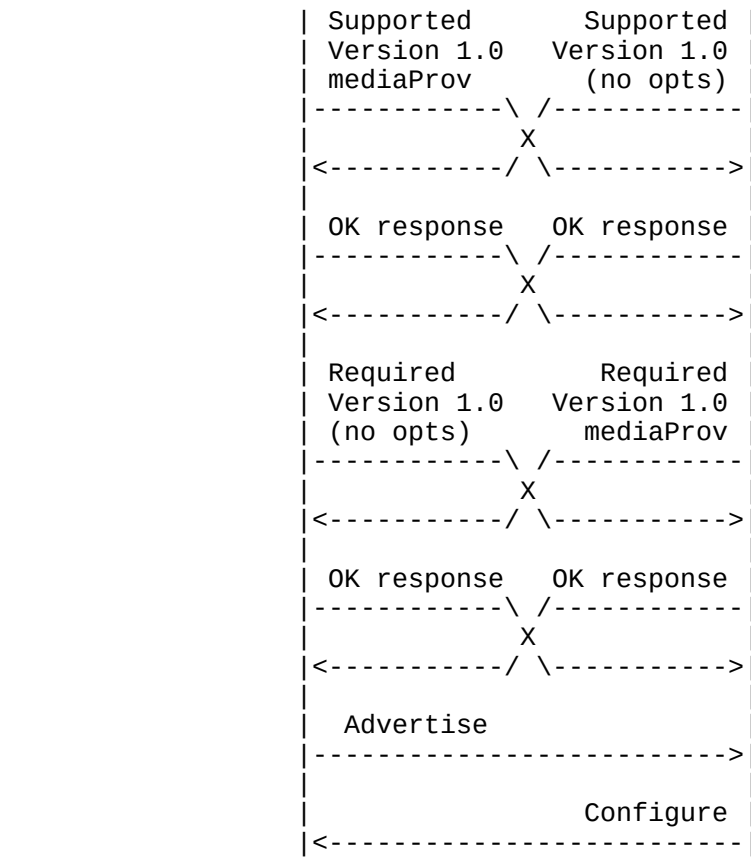
The element on the right, upon receiving the Supported message, selects version 1.1 because it is the highest version in common to the two sides. It sends a Required message containing version 1.1 because that is the highest version in common. It also includes the mediaProvider option element, because it wants its peer to provide

media.

Upon receiving the Required messages, both endpoints determine that they should send ADVERTISEMENTs.

ADVERTISEMENT and CONFIGURE messages will flow in both directions.

A.1.8.2. Successful Negotiation - Consumer-Only Endpoint

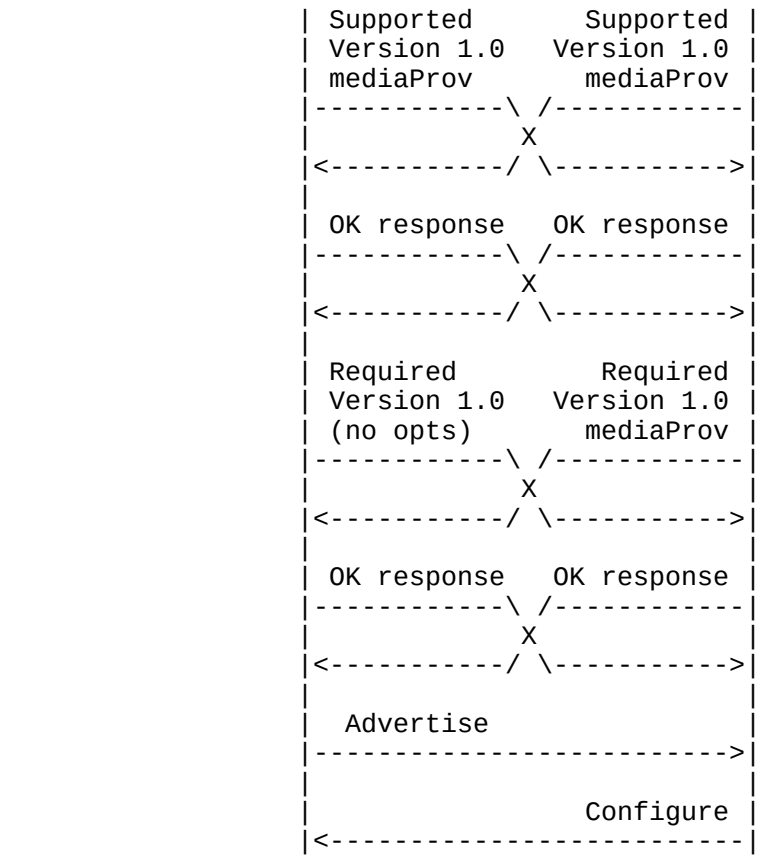


The endpoint on the right consumes media, but doesn’t provide any so it doesn’t include the mediaProvider option element in the Supported message it sends.

The element on the left would like to include a mediaProvider option element in the Requirements message it sends, but can’t because it did not receive one in the Supported message it received.

ADVERTISEMENT messages will only go from left to right, and CONFIGURE messages will only go from right to left.

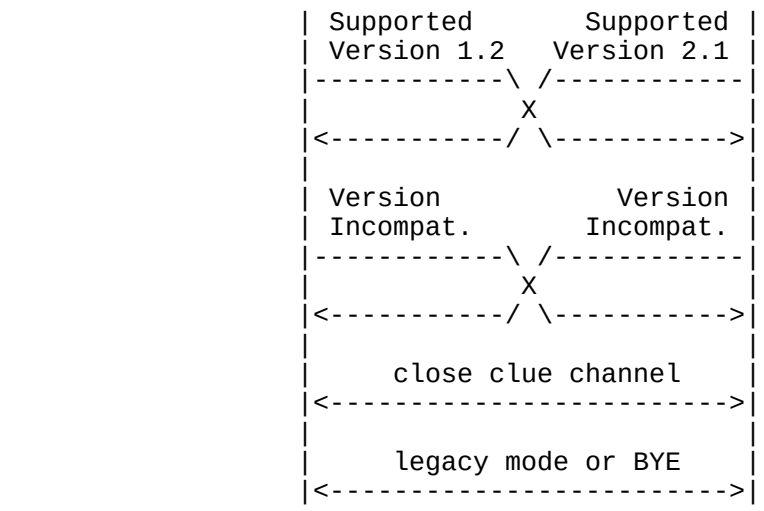
A.1.8.3. Successful Negotiation - Provider-Only Endpoint



The endpoint on the left provides media but does not consume any so it includes the mediaProvider option element in the Supported message it sends, but does't include the mediaProvider option element in the Required message it sends.

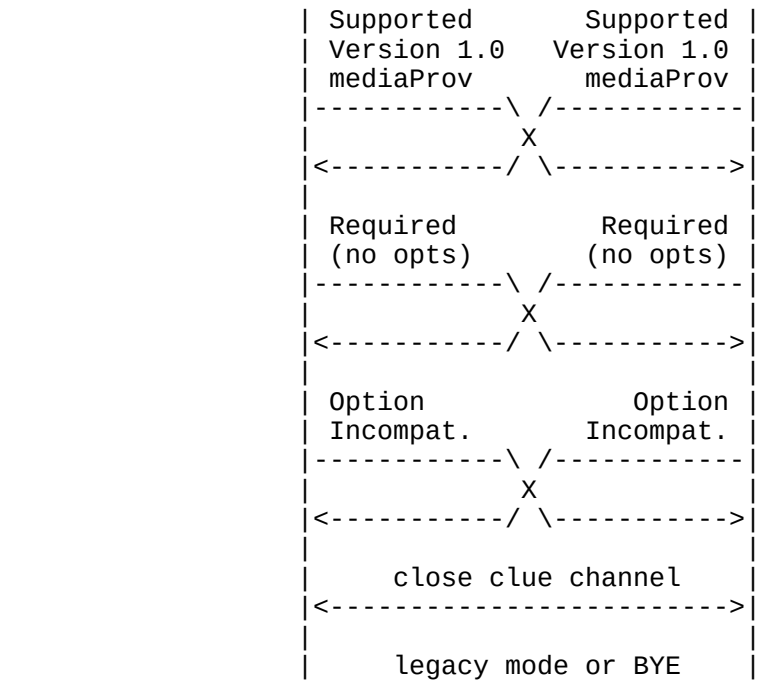
ADVERTISEMENT messages will only go from left to right, and CONFIGURE messages will only go from right to left.

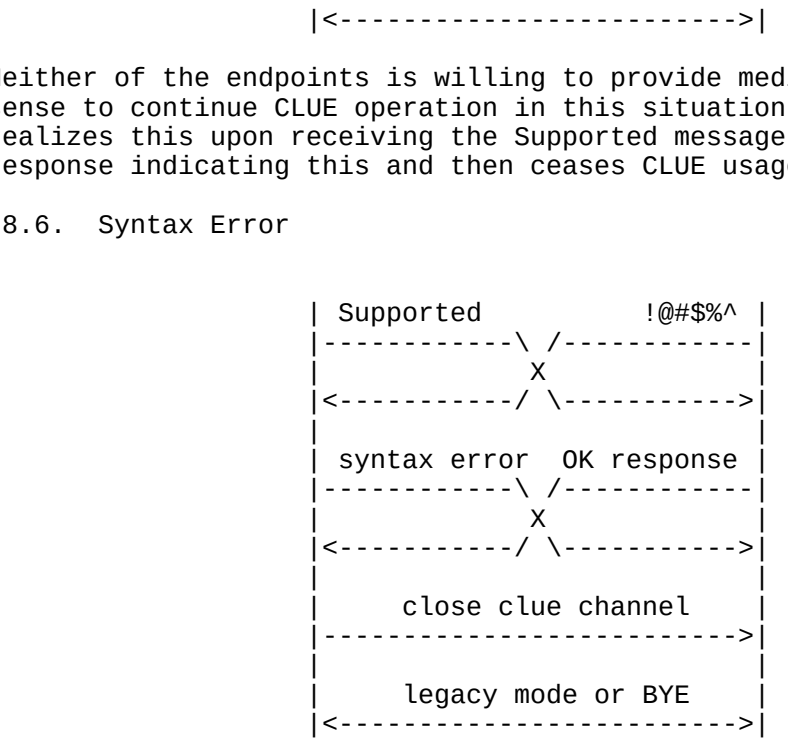
A.1.8.4. Version Incompatibility



Upon receiving the Supported message, each endpoint discovers there is no major version in common, so CLUE usage is not possible. Each sends an error response indicating this and then ceases CLUE usage.

A.1.8.5. Option Incompatibility





A.2. Message Transport

CLUE messages are transported over a bidirectional CLUE channel. In a two-party CLUE session, a CLUE channel connects the two endpoints. In a CLUE conference, each endpoint has a CLUE channel connecting it to an MCU. (In conferences with cascaded mixers [RFC4353], two MCUs will be connected by a CLUE channel.)

A.2.1. CLUE Channel Lifetime

The transport mechanism used for CLUE messages is DTLS/SCTP as specified in [I-D.tuexen-tsvwg-sctp-dtls-encaps] and [I-D.ietf-mmusic-sctp-sdp]. A CLUE channel consists of one SCTP stream in each direction over a DTLS/SCTP session. The mechanism for establishing the DTLS/SCTP session is described in Section 4.

The CLUE channel will usually be offered during the initial SIP INVITE, and remain connected for the duration of the CLUE/SIP session. However this need not be the case. The CLUE channel may be established mid-session after desire and capability for CLUE have been determined, and the CLUE channel may be dropped mid-call if the desire and/or capability to support it is lost.

There may be cases when it becomes necessary to "reset" the CLUE channel. This by be as a result of an error on the underlying SCTP association, a need to change the endpoint address of the SCTP association, loss of CLUE protocol state, or something else TBD.

The precise mechanisms used to determine when a reset is required, and how to accomplish it and return to a well defined state are TBS.

A.2.2. Channel Error Handling

We will need to specify behavior in the face of transport errors that are so severe that they can't be managed via CLUE messaging within the CLUE channel. Some errors of this sort are:

- Unable to establish the SCTP association after signaling it in SDP.
- CLUE channel setup rejected by peer.
- Error reported by transport while writing message to CLUE channel.
- Error reported by transport while reading message from CLUE channel.
- Timeout - overdue acknowledgement of a CLUE message. (Requirements for now soon a message must be responded to are TBD.)
- Application fault. CLUE protocol state lost.

The worst case is to drop the entire CLUE call. Another possibility is to fall back to legacy compatibility mode. Or perhaps a "reset" can be done on the protocol. E.g. this might be accomplished by sending a new O/A and establishing a replacement SCTP association. Or a new CLUE channel might be established within the existing SCTP association.

A.3. Message Framing

Message framing is provided by the SCTP transport protocol. Each CLUE message is carried in one SCTP message.

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 13, 2014

P. Kyzivat
L. Xiao
C. Groves
Huawei
R. Hansen
Cisco Systems

April 11, 2014

CLUE Signaling
draft-kyzivat-clue-signaling-08

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 October 13, 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. Media Feature Tag Definition 5
4. SDP Grouping Framework TELEPRESENCE Extension Semantics . . . 5
4.1. General 5
4.2. The CLUE data channel and the TELEPRESENCE grouping semantic 5
4.3. CLUE-controlled media and the TELEPRESENCE grouping semantic 6
4.4. SDP Offer/Answer Procedures 6
4.4.1. Generating the Initial Offer 6
4.4.1.1. Signalling CLUE Encodings 6
4.4.1.2. Receiving CLUE-controlled media 8
4.4.1.3. Interoperability with non-CLUE devices 8
4.4.2. Generating the Answer 8
4.4.2.1. Negotiating use of CLUE and the CLUE data channel 8
4.4.2.2. Negotiating receipt of CLUE capture encodings in SDP 8
4.4.3. Processing the initial Offer/Answer negotiation . . . 9
4.4.3.1. Successful CLUE negotiation 9
4.4.3.2. CLUE negotiation failure 9
4.4.4. Modifying the session 9
4.4.4.1. Enabling CLUE mid-call 9
4.4.4.2. Disabling CLUE mid-call 10
5. Interaction of CLUE protocol and SDP negotiations 10
5.1. Independence of SDP and CLUE negotiation 10
5.2. Recommendations for operating with non-atomic operations 11
5.3. Constraints on sending media 11
6. Multiplexing of CLUE-controlled media using BUNDLE 12
6.1. Overview 12
6.2. Usage of BUNDLE with CLUE 12
6.2.1. Generating the Initial Offer 12
6.2.2. Bundle Address Synchronization 12
6.2.3. Multiplexing of the data channel and RTP media 13
7. Example: A call between two CLUE-capable endpoints 13
8. Example: A call between a CLUE-capable and non-CLUE endpoint 20
9. CLUE requirements on SDP O/A 22
10. SIP Signaling 22

11. CLUE over RTCWEB 22
12. Open Issues 23
13. What else? 23
14. Acknowledgements 23
15. IANA Considerations 23
16. Security Considerations 23
17. Change History 23
18. References 25
18.1. Normative References 25
18.2. Informative References 26
Appendix A. CLUE Signalling and data channel concerns 27
A.1. Protocol Versioning and Options 27
A.1.1. Versioning Objectives 27
A.1.2. Versioning Overview 28
A.1.3. Version Negotiation 30
A.1.4. Option Negotiation 31
A.1.5. Option Elements 31
A.1.5.1. <mediaProvider> 31
A.1.6. Version & option negotiation errors 32
A.1.7. Definition and Use of Version Numbers 33
A.1.8. Version & Option Negotiation Examples 34
A.1.8.1. Successful Negotiation - Multi-version 34
A.1.8.2. Successful Negotiation - Consumer-Only Endpoint . 35
A.1.8.3. Successful Negotiation - Provider-Only Endpoint . 36
A.1.8.4. Version Incompatibility 37
A.1.8.5. Option Incompatibility 38
A.1.8.6. Syntax Error 39
A.2. Message Transport 39
A.2.1. CLUE Channel Lifetime 39
A.2.2. Channel Error Handling 40
A.3. Message Framing 40
Authors' Addresses 40

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.

This document originally also defined the CLUE protocol itself. These details have mostly been split out into [I-D.presta-clue-protocol] and expanded, but at present some details remain in this document.

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.

CLUE-enabled device: A CLUE-capable device that wishes to negotiate a CLUE data channel and send and/or receive CLUe-controlled media.

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. There is a corresponding "non-CLUE-controlled" media term. See Section 4 for details of how this control is signalled in SDP

3. Media Feature Tag Definition

The "sip.telepresence" 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 TELEPRESENCE Extension Semantics

4.1. General

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

The TELEPRESENCE 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 TELEPRESENCE group.

4.2. The CLUE data channel and the TELEPRESENCE 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 TELEPRESENCE group in the SDP and include the "mid" of the "m" line for the data channel in that group. A TELEPRESENCE grup MUST NOT include the "mid"s for more than one data channel, and the data channel "mid" MUST NOT be included in more than one TELEPRESENCE group.

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

4.3. CLUE-controlled media and the TELEPRESENCE 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 TELEPRESENCE group. CLUE-controlled media line "mid"s MUST NOT be included in more than one TELEPRESENCE group.

CLUE-controlled media is controlled by the CLUE protocol as negotiated on the CLUE data channel with an "mid" included in the TELEPRESENCE group. If no data channel is included in the group 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 Offer/Answer Procedures

4.4.1. Generating the Initial Offer

4.4.1.1. 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 TELEPRESENCE group as defined above.

As well as the normal restrictions defined in [RFC3264] media MUST NOT be sent on this stream until the media provider has received a valid CLUE CONFIGURE message specifying the capture to be used for this stream. In the case of RTP media this includes corresponding RTCP packets.

Every "m" line representing a CLUE encoding SHOULD 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.

4.4.1.1.1. Media line directionality

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

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

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

4.4.1.1.2. Alternate encoding limit syntaxes

Note that while the expressing of CLUE encoding limits in SDP has been discussed at some length by the working group and it has been agreed that this is the current, working assumption, formal consensus has not been agreed on this. Alternatives include placing encoding limits in the CLUE ADVERTISEMENT message, or by using alternate SDP syntax, such as is suggested in [I-D.groves-clue-latent-config].

4.4.1.2. Receiving CLUE-controlled media

As well as including sendonly media lines to send CLUE-controlled media, the sender of the initial SDP offer MAY also include "a=recvonly" media lines to preallocate "m" lines to receive media; these are described in more detail in the next section.

4.4.1.3. Interoperability with non-CLUE devices

A CLUE-enabled 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-enabled, for instance due to receiving an initial INVITE with no SDP but including a 'sip.telepresence' media feature tag, the above recommendation is waived, and the initial offer MAY contain "m" lines for CLUE-controlled media.

4.4.2. Generating the Answer

4.4.2.1. Negotiating use of CLUE and the CLUE data channel

If the recipient wishes to enable CLUE for the call, they MUST negotiate data channel support for an "m" line, and include the "mid" of that "m" line in a corresponding TELEPRESENCE group.

4.4.2.2. 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 an "mid" the corresponding TELEPRESENCE group corresponding to the encoding they wish to send.

In the case of RTCP for RTP media or any other media type that

includes a bidirectional flow of packets for unidirectional media streams, such bidirectional packets MUST NOT be sent until the media consumer has received acknowledgement that the media provider has received a valid CLUE CONFIGURE message specifying the capture to be used for this stream.

4.4.3. 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 TELEPRESENCE groups CLUE has been successfully negotiated and the call is now CLUE-enabled, otherwise the call is not CLUE enabled.

4.4.3.1. 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-enabled 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-enabled device MUST still be prepared to receive media on non-CLUE-controlled media lines 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.

4.4.3.2. 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.

4.4.4. Modifying the session

4.4.4.1. Enabling CLUE mid-call

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

to indicate that it is CLUE-enabled.

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 TELEPRESENCE groups then the call is now CLUE-enabled; negotiation of the data channel and subsequently the CLUE protocol begin.

4.4.4.2. 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 TELEPRESENCE 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 TELEPRESENCE 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 complicated state machines with the potential to reach invalid states if messages were to be lost, or be rewritten en-route by middle boxes, the current proposal is that SDP and CLUE messages are independent. 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.

CLUE messages contain an EncodingID which is used to identify a specific encoding 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 EncodingID which does not currently match an extant id in SDP.

5.2. Recommendations for operating with non-atomic operations

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.

5.3. 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 - 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.

6. Multiplexing of CLUE-controlled media using BUNDLE

6.1. Overview

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

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

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

6.2. Usage of BUNDLE with CLUE

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

6.2.1. Generating the Initial Offer

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

6.2.2. Bundle Address Synchronization

When using BUNDLE the initial offerer may be mandated to send a Bundle Address Synchronisation offer. If the initial offerer also followed the recommendation of not including CLUE-controlled media lines in their offer, they MAY choose to include them in this subsequent offer. In this circumstance the BUNDLE specification recommends that the offerer does not "modify SDP parameters that

could get the answerer to reject the BAS offer". Including new CLUE-controlled media lines using codecs and other attributes used in existing media lines should not increase the chance of the answerer rejecting the BAS offer; implementations should consider carefully before including new codecs or other new SDP attributes in these CLUE-controlled media lines.

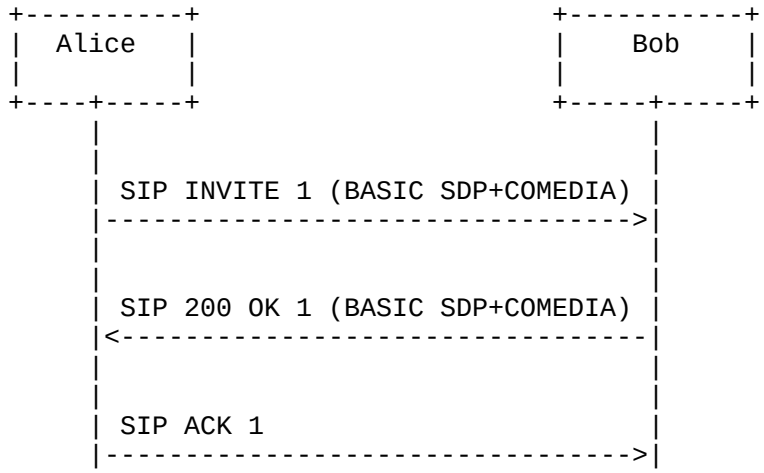
6.2.3. Multiplexing of the data channel and RTP media

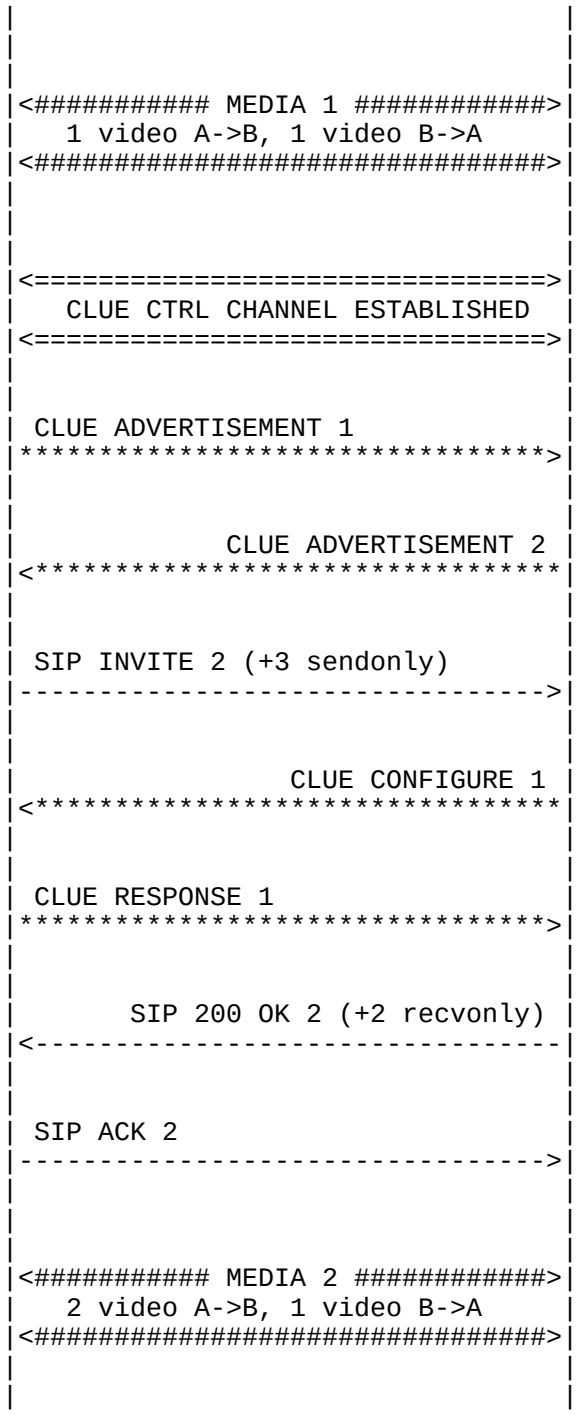
BUNDLE-supporting CLUE-enabled devices MAY include the data channel in the same BUNDLE group as RTP media. In this case the device MUST be able to demultiplex the various transports - see section 7.2 of the BUNDLE draft [I-D.ietf-mmusic-sdp-bundle-negotiation]. If the BUNDLE group includes other protocols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

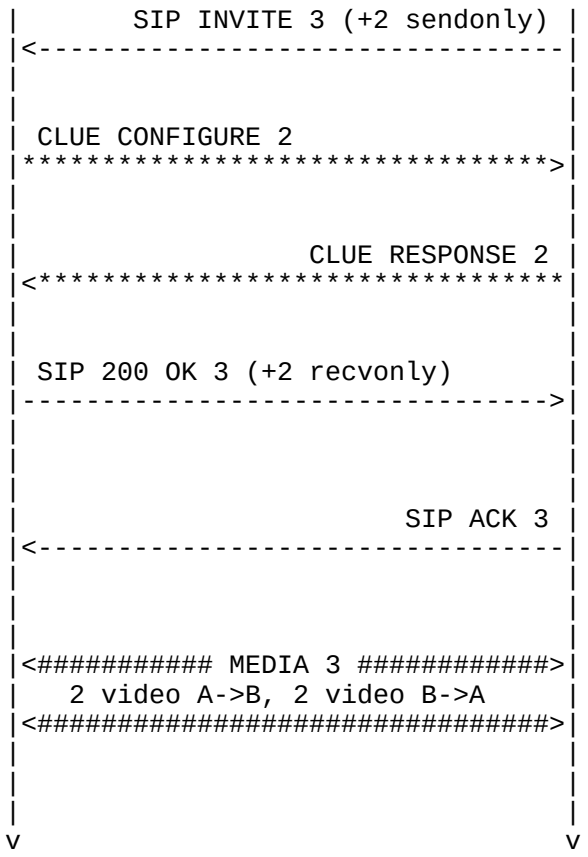
7. Example: A call between two CLUE-capable endpoints

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



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

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

```
a=mid:2
```

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

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

Alice now sends her CLUE Advertisement (ADVERTISEMENT 1). She advertises three static captures representing her three cameras. She also includes switched captures suitable for two- and one-screen systems. All of these captures are in a single capture scene, with suitable capture scene entries to tell Bob that he should either subscribe to the three static captures, the two switched capture view or the one switched capture view. Alice has no simultaneity constraints, so includes all six captures in one simultaneous set. Finally, Alice includes an encoding group with three encoding IDs: "enc1", "enc2" and "enc3". These encoding ids aren’t currently valid, but will match the next SDP offer she sends.

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

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

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

Alice now sends INVITE 2. She maintains the sendrecv audio, video

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

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

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

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

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

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

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

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

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

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

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

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

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

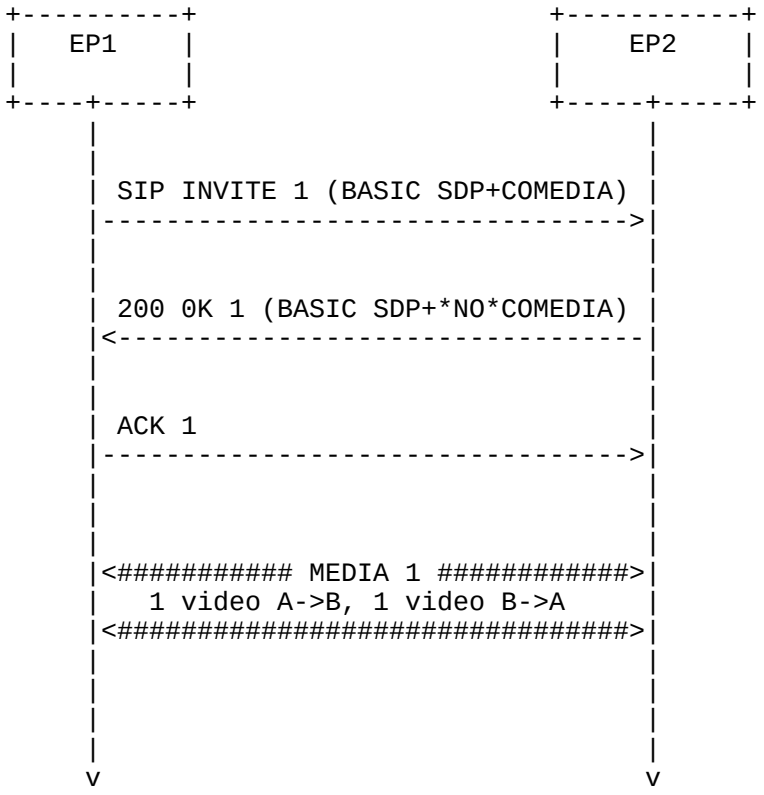
Finally, on receiving 200 OK 3 Bob is now able to send the two streams of video Alice requested - this is illustrated as MEDIA 3.

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

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

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

protocol.



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

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

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

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

9. CLUE requirements on SDP O/A

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

Any other SDP extensions required to support CLUE signaling should also be specified here. Then we will need to take action within MMUSIC to make those happen. This section should be empty and removed before this document becomes an RFC.

NOTE: The RTP mapping document [I-D.even-clue-rtp-mapping] is also likely to call for SDP extensions. We will have to reconcile how to coordinate these two documents.

10. SIP Signaling

(Placeholder) This may be unremarkable. If so we can drop it.

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.

- While the preference is to multiplex multiple capture encodings over a single RTP session, this will not always be desirable or possible. The factors that prevent multiplexing may come from either the provider or the consumer. So the extent of multiplexing must be negotiated. The decision about how to multiplex affects the number and grouping of m-lines in the SDP. The endpoint of a CLUE session that sends an offer needs to know the mapping of capture encodings to m-lines for both sides.

  AFAIK this issue hasn't yet been considered at all.
- 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 has 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

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

-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.

-06: Revisions by Rob Hansen
* Removed CLUE message XML schema and details that are now in draft-presta-clue-protocol
* Encoding limits in SDP section updated to note that this has been investigated and discussed and is the current working assumption of the WG, though consensus has not been fully achieved.
* A section has also been added on the current mandation of unidirectional "m"-lines.
* Updated CLUE messaging in example call flow to match draft-presta-clue-protocol-03

-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.)

-04: Revisions by Rob Hansen: ???
-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.
-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.

-01: Updated by roberta to include some sample call flows.
-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-14 (work in progress), February 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-03 (work in progress), November 2013.

[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, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-06 (work in progress), April 2014.

Appendix A. CLUE Signalling and data channel concerns

[The specifics of the CLUE signaling protocol are in the process of being defined in [I-D.presta-clue-protocol], while the negotiation of the CLUE data channel is being defined in [I-D.ietf-clue-datachannel]. As such, considerable text originally in this section have been transitioned to these document. The following text relates to issues that are no longer the focus of this document, but remain important and unresolved, and so have been preserved here.]

A.1. Protocol Versioning and Options

A.1.1. Versioning Objectives

The CLUE versioning mechanism addresses the following needs:

- Coverage:
  - Versioning of basic behavior and options,
  - CLUE message exchange,
  - CLUE message exchange,
  - coordinated use of SIP and SDP,
  - required media behavior.
- Remain fixed for the duration of the CLUE channel
- Be extensible for configuration of new options.
- Be sufficient (with extensions) for all envisioned future versions.

A.1.2. Versioning Overview

An initial message exchange on the CLUE channel handles the negotiation of version and options.

- Dedicated message types are used for this negotiation.
- The negotiation is repeated if the CLUE channel is reestablished.

The version usage is similar in philosophy to XMPP:

- See [RFC6120] section 4.7.5.
- A version 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.
- If a common major version cannot be negotiated, then CLUE MUST NOT be used.
- The same message exchange also negotiates options.
- Each option is denoted by a unique XML element in the negotiation.

Figure 1 shows the negotiation in simplified form:

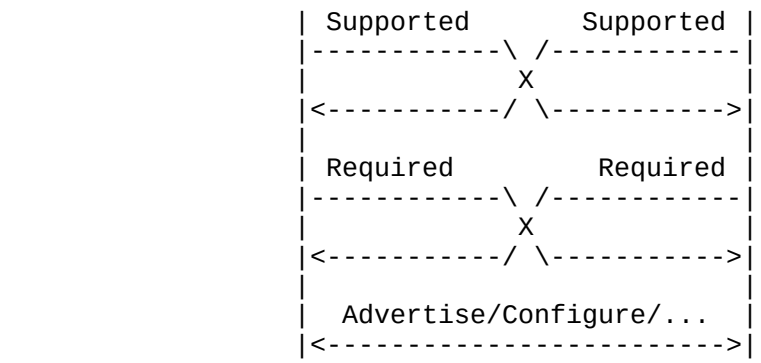


Figure 1: Basic Option Negotiation (simplified)

Dedicated message types are used for the negotiation because:

- The protocol can then ensure that the negotiation is done first, and once. Not changing mid-session means an endpoint can plan ahead, and predict what may be used and what might be received.
- This provides extensible framework for negotiating optional features.
- A full option negotiation can be completed before other messages are exchanged.

Figure 2 and Figure 3 are simplified examples of the Supported and Required messages:

```
<supported>
        <version major="1" minor="0">
          <!- May repeat version if multiple
         major versions supported.      ->
   <!- Options follow ->
   <mediaProvider/>
        ...
</supported>
```

Figure 2: Supported Message (simplified)

```
<required>
        <version major="1" minor="0">
          <!- Requested options of peer follow ->
   <!- Options follow ->
   <mediaProvider/>
        ...
</required>
```

Figure 3: Required Message (simplified)

A.1.3. Version Negotiation

The Supported message includes one or more <version> elements, each denoting a major/minor version combination that the sender of the message is capable of supporting.

The <version> element contains both a major and minor version. Each is a non-negative integer. Each <version> element in the message MUST contain a unique major version number, distinct from the major version number in all the other <version> elements in the message. The minor version in a <version> element denotes the largest minor version the sender supports for the corresponding major version. (Minor versions are always backwards compatible, so support for a minor version implies support for all smaller minor versions.)

Each endpoint of the CLUE channel sends a Supported message, and receives the Supported message sent by the other end. Then each end compares the versions sent and the versions received to determine the version to be used for this CLUE session.

- If there is no major version in common between the two ends, negotiation fails.
- The <version> elements from the two ends that have the largest matching major version are selected.
- After exchange each end determines compatible version numbers to be used for encoding and decoding messages, and other behavior in the CLUE session.
  - The <version> elements from the two ends that have the largest matching major version are selected.
  - The side that sent the smaller minor version chooses the one it sent.
  - The side that sent the larger minor version may choose the minor version it received, or the one it sent, or any value between those two.
- Each end then sends a Required message with a single <version> element containing the major and minor versions it has chosen.

  [[Note: "required" is the wrong semantic for this. Might want a better message name.]]
- Each end then behaves in accord with the specifications denoted by the version it chose. This continues until the end of the CLUE session, or until changed as a result of another version negotiation when the CLUE channel is reestablished.

  [[Note: The version negotiation remains in effect even if the CLUE channel is lost.]]

A.1.4. Option Negotiation

Option negotiation is used to agree upon which options will be available for use within the CLUE session. (It does not say that these options must be used.) This may be used for both standard and proprietary options. (As used here, and option could be either a feature described as part of this specification that is optional to implement, or a feature defined in a separate specification that extends this one.)

Each end includes, within the Supported message it sends, elements describing those options it is willing and able to use with this CLUE session.

Each side, upon receiving a Supported message, selects from that message those option elements that it wishes the peer to use. (If/when occasion for that use arises.) It then includes those selected elements into the Required message that it sends.

Within a received Supported message, unknown option elements MUST be ignored. This includes elements that are of a known type that is not known to denote an option.

A.1.5. Option Elements

Each option is denoted, in the Supported and Required messages, by an XML element. There are no special rules for these elements - they can be any XML element. The attributes and body of the element may carry further information about the option. The same element type is used to denote the option in the Supported message and the corresponding Required message, but the attributes and body may differ according to option-specific rules. This may be used to negotiate aspects of a particular option. The ordering of option elements is irrelevant within the Supported and Required messages, and need not be consistent in the two.

Only one option element is defined in this document: <mediaProvider>.

A.1.5.1. <mediaProvider>

The <mediaProvider> element, when placed in a Supported message, indicates that the sender is willing and able to send ADVERTISEMENT messages and receive CONFIGURE messages. When placed in a Required message, the <mediaProvider> element indicates that the sender is willing, able, and desirous of receiving ADVERTISEMENT messages and sending CONFIGURE messages. If an endpoint does not receive <mediaProvider> in a Required message, it MUST NOT send ADVERTISEMENT messages. For common cases <mediaProvider> should be supported and

required by both endpoints, to enable bidirectional exchange of media. If not required by either end, the CLUE session is useless. This is an error condition, and SHOULD result in termination of the CLUE channel.

The <mediaProvider> element has no defined attributes or body.

A.1.6. Version & option negotiation errors

The following are errors that may be detected and reported during version negotiation:

- Version incompatibility

  There is no common value between the major version numbers sent in a Supported message and those in the received Supported message.
- Option incompatibility

  This can occur if options supported by one endpoint are inconsistent with those supported by the other endpoint. E.g., The <mediaProvider> option is not specified by either endpoint. Options SHOULD be specified so as to make it difficult for this problem to occur.

  This error may also be used to indicate that insufficient options have been required among the two ends for a useful session to result. This can occur with a feature that needs to be present on at least one end, but not on a specific end. E.g., The <mediaProvider> option was Supported by at least one of the endpoints, but it was not Required by either.

  This may also be used to indicate that an option element in the Required message has attributes or body content that is syntactically correct, but in inconsistent with the rules for option negotiation specified for that particular element. The definition of each option must specify the negotiation rules for that option.
- Unsupported option

  An option element type received in a Required message did not appear in the corresponding Supported element.

  (Unsupported options received in a Supported message do not trigger this error. They are ignored.)

These errors are reported using the normal message error reporting mechanism.

Other applicable error codes may also be returned in response to a Supported or Required message.

Errors that occur at this stage result in negotiation failure. When this occurs, CLUE cannot be used until the end of the SIP session, or until a new CLUE channel is negotiated and a subsequent version negotiation succeeds. The SIP session may continue without CLUE features.

A.1.7. Definition and Use of Version Numbers

[[NOTE: THIS IS AWKWARD. SUGGESTIONS FOR BETTER WAYS TO DEFINE THIS ARE WELCOME.]]

This document defines CLUE version 1.0 (major=1, minor=0). This denotes the normative behavior defined in this document and other documents upon which it normatively depends, including but is not limited to:

- the schema defined in [I-D.presta-clue-protocol];
- the schema defined in [clue-data-model];
- the protocol used to exchange CLUE messages;
- the protocol defined herein that defines valid sequence of CLUE messages;
- the specific rules defined herein for employing SIP, SDP, and RTP to realize the CLUE messages.

Given two CLUE versions Vx and Vy, then Vx is backward compatible with Vy if and only if:

- All messages valid according to the schema of Vx are also valid according to the schemas of Vy
- All messages valid according to the schema of Vy can be made valid according to the schemas of Vx by deleting elements undefined in the schemas of Vx.

  [[NOTE: THIS PROBABLY NEEDS WORK!]]
- All normative behaviors defined for Vx are defined consistently for Vy.

  [[NOTE: SOME HAND WAVING HERE.]]

Revisions, updates, to any of the documents denoted by Version 1.0 MAY result in the definition of a new CLUE version. If they do, then this document MUST be revised to define the new version.

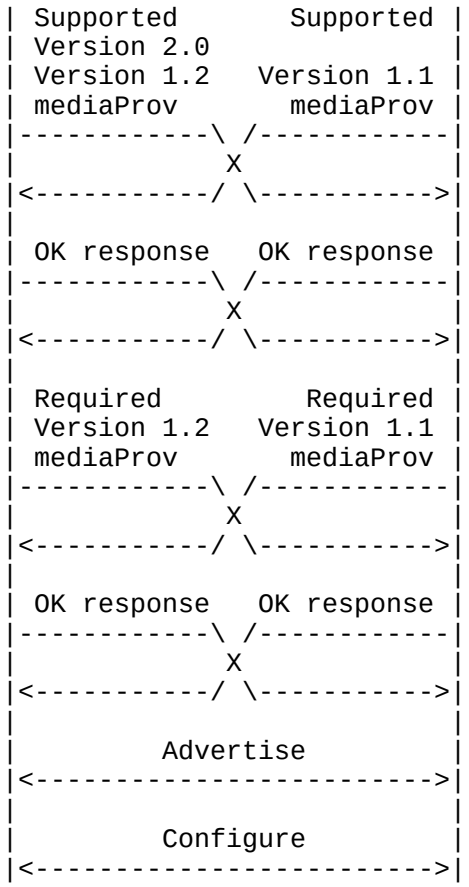
The CLUE version to be defined in a revision to this document MUST be determined as follows:

- o If the revision and the document being revised are mutually backward compatible (they are functionally equivalent), then the CLUE version MUST remain unchanged.
- o Else if the revision is backward compatible with the document being revised, then the CLUE major version MUST remain unchanged, and the CLUE minor version MUST be increased by one (1).
- o Else the CLUE major version must be increased by one (1), and the CLUE minor version set to zero (0).

When a CLUE implementation sends a Supported message, it MUST include the CLUE versions it is willing and able to conform with.

A.1.8. Version & Option Negotiation Examples

A.1.8.1. Successful Negotiation - Multi-version



The endpoint on the left can support versions 1.2 and 2.0, and because of backward compatibility can support versions 1.0 and 1.1. The endpoint on the right supports only version 2.0. Both endpoints with to both provide and consume media. They each send a Supported message indicating what they support.

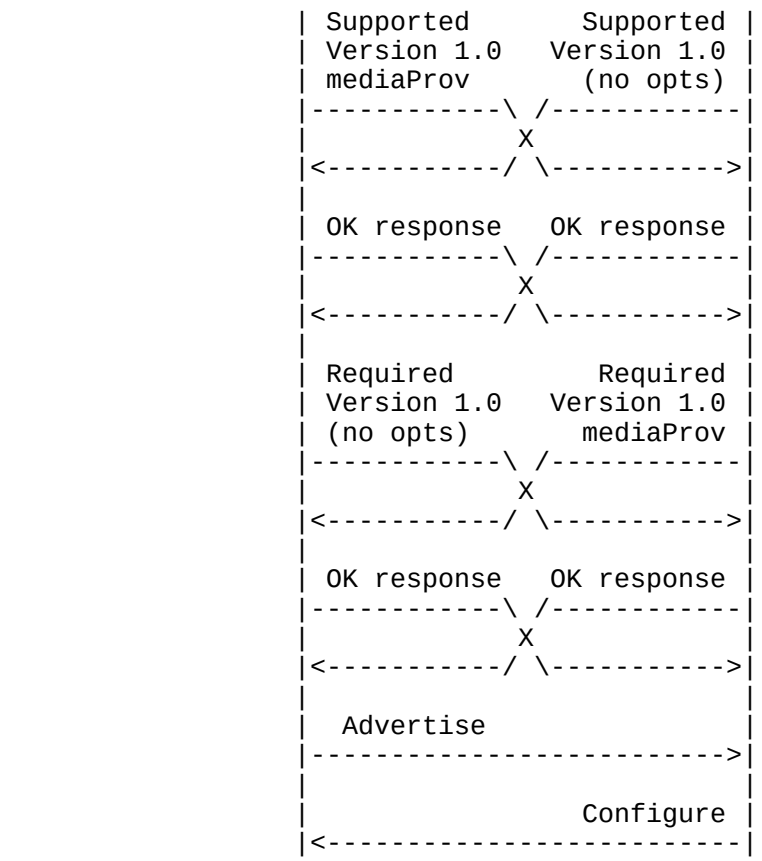
The element on the left, upon receiving the Supported message, determines that it is permitted to use version 1.2 or 1.1, and decides to use 1.2. It sends a Required message containing version 1.2 and also includes the mediaProvider option element, because it wants its peer to provide media.

The element on the right, upon receiving the Supported message, selects version 1.1 because it is the highest version in common to the two sides. It sends a Required message containing version 1.1 because that is the highest version in common. It also includes the mediaProvider option element, because it wants its peer to provide media.

Upon receiving the Required messages, both endpoints determine that they should send ADVERTISEMENTs.

ADVERTISEMENT and CONFIGURE messages will flow in both directions.

A.1.8.2. Successful Negotiation - Consumer-Only Endpoint

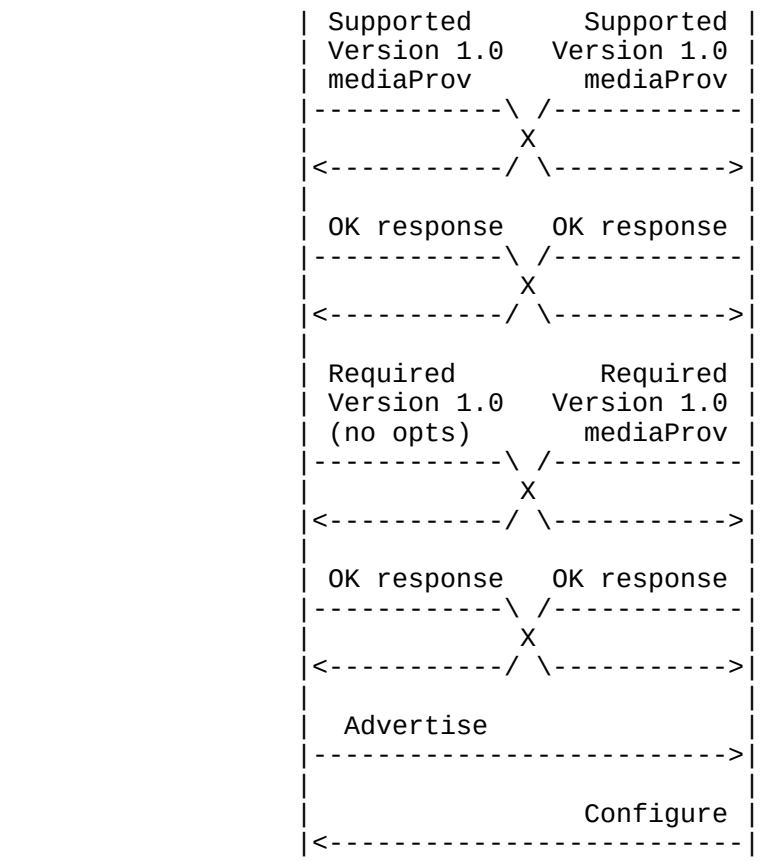


The endpoint on the right consumes media, but doesn't provide any so it doesn't include the mediaProvider option element in the Supported message it sends.

The element on the left would like to include a mediaProvider option element in the Requirements message it sends, but can't because it did not receive one in the Supported message it received.

ADVERTISEMENT messages will only go from left to right, and CONFIGURE messages will only go from right to left.

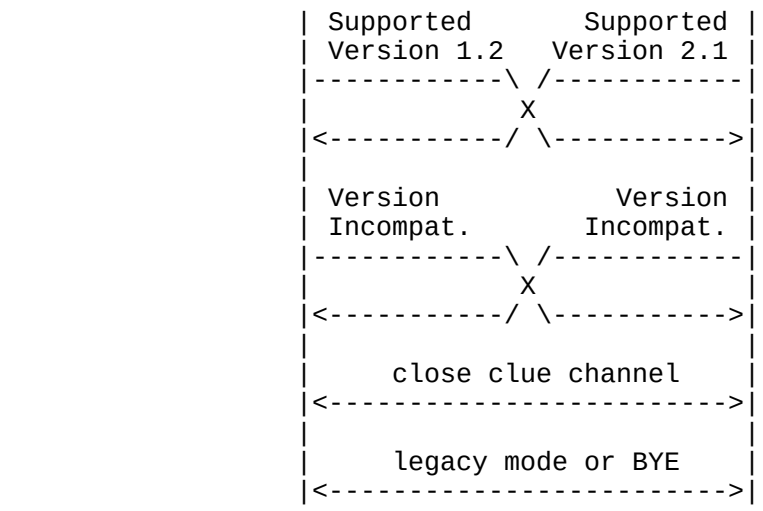
A.1.8.3. Successful Negotiation - Provider-Only Endpoint



The endpoint on the left provides media but does not consume any so it includes the mediaProvider option element in the Supported message it sends, but does't include the mediaProvider option element in the Required message it sends.

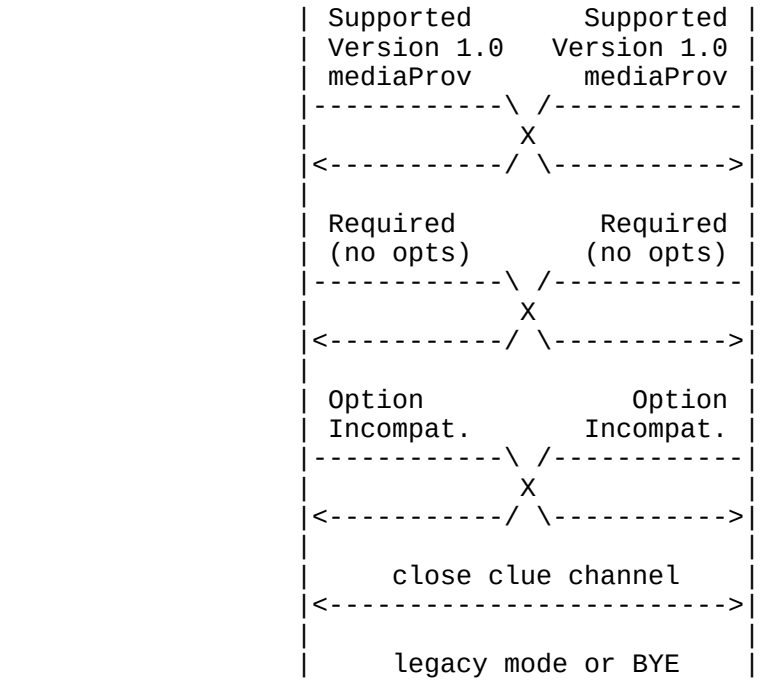
ADVERTISEMENT messages will only go from left to right, and CONFIGURE messages will only go from right to left.

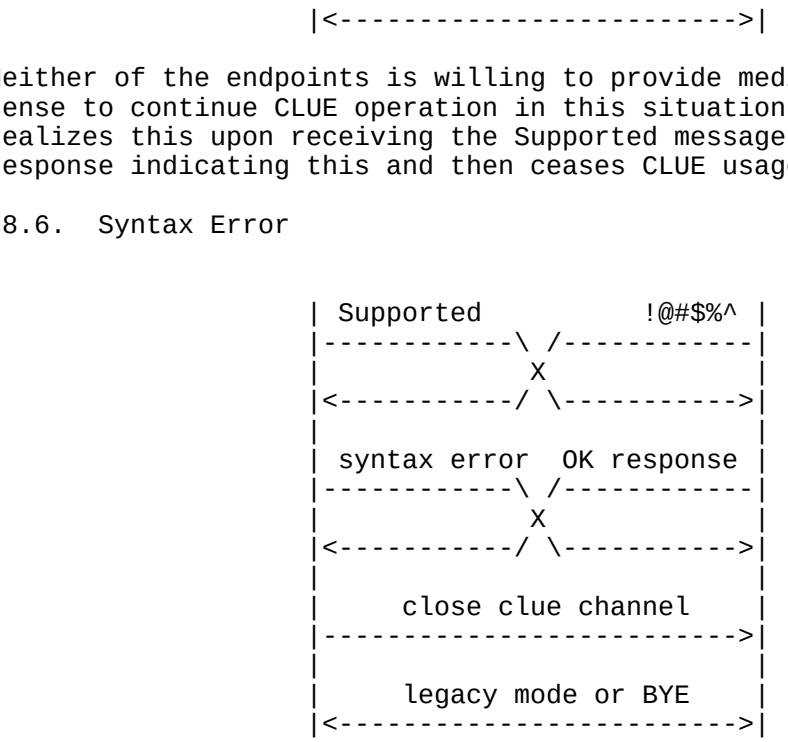
A.1.8.4. Version Incompatibility



Upon receiving the Supported message, each endpoint discovers there is no major version in common, so CLUE usage is not possible. Each sends an error response indicating this and then ceases CLUE usage.

A.1.8.5. Option Incompatibility





Neither of the endpoints is willing to provide media. It makes no sense to continue CLUE operation in this situation. Each endpoint realizes this upon receiving the Supported message, sends an error response indicating this and then ceases CLUE usage.

A.1.8.6. Syntax Error

A.2. Message Transport

CLUE messages are transported over a bidirectional CLUE channel. In a two-party CLUE session, a CLUE channel connects the two endpoints. In a CLUE conference, each endpoint has a CLUE channel connecting it to an MCU. (In conferences with cascaded mixers [RFC4353], two MCUs will be connected by a CLUE channel.)

A.2.1. CLUE Channel Lifetime

The transport mechanism used for CLUE messages is DTLS/SCTP as specified in [I-D.tuexen-tsvwg-sctp-dtls-encaps] and [I-D.ietf-mmusic-sctp-sdp]. A CLUE channel consists of one SCTP stream in each direction over a DTLS/SCTP session. The mechanism for establishing the DTLS/SCTP session is described in [I-D.ietf-clue-datachannel].

The CLUE channel will usually be offered during the initial SIP INVITE, and remain connected for the duration of the CLUE/SIP session. However this need not be the case. The CLUE channel may be established mid-session after desire and capability for CLUE have been determined, and the CLUE channel may be dropped mid-call if the

desire and/or capability to support it is lost.

There may be cases when it becomes necessary to "reset" the CLUE channel. This by be as a result of an error on the underlying SCTP association, a need to change the endpoint address of the SCTP association, loss of CLUE protocol state, or something else TBD.

The precise mechanisms used to determine when a reset is required, and how to accomplish it and return to a well defined state are TBS.

A.2.2. Channel Error Handling

We will need to specify behavior in the face of transport errors that are so severe that they can't be managed via CLUE messaging within the CLUE channel. Some errors of this sort are:

- Unable to establish the SCTP association after signaling it in SDP.
- CLUE channel setup rejected by peer.
- Error reported by transport while writing message to CLUE channel.
- Error reported by transport while reading message from CLUE channel.
- Timeout - overdue acknowledgement of a CLUE message. (Requirements for now soon a message must be responded to are TBD.)
- Application fault. CLUE protocol state lost.

The worst case is to drop the entire CLUE call. Another possibility is to fall back to legacy compatibility mode. Or perhaps a "reset" can be done on the protocol. E.g. this might be accomplished by sending a new O/A and establishing a replacement SCTP association. Or a new CLUE channel might be established within the existing SCTP association.

A.3. Message Framing

Message framing is provided by the SCTP transport protocol. Each CLUE message is carried in one SCTP message.

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

CLUE Working Group R. Presta
Internet-Draft S P. Romano
Intended status: Informational University of Napoli
Expires: April 11, 2014 October 8, 2013

CLUE protocol
draft-presta-clue-protocol-02

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. We herein focus on the application level perspective. Message details, together with the behavior of both the Media Provider and the Media Consumer are 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 April 11, 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 3
3. Overview of the CLUE protocol messages 3
3.1. ADVERTISEMENT 4
3.2. CONFIGURE 5
3.3. RESPONSE 6
3.4. RE-ADV 9
3.5. OPTIONS 10
4. Protocol state machines 10
5. Media Consumer's state machine 11
6. Media Provider's state machine 13
7. About CLUE protocol XML schema versioning 15
8. Extensibility issues 16
8.1. Aspect 1 - new information within existing messages . . . 16
8.2. Aspect 2 - new messages 17
9. Managing protocol version negotiation and extensions: the OPTIONS request 17
9.1. An example using OPTIONS 19
10. XML Schema of CLUE protocol messages 20
11. Examples 24
12. Handling channel errors 24
13. Diff with the -01 version 24
14. Diff with -00 version 25
15. Informative References 25

1. Introduction

The CLUE protocol is an application protocol used by a Media Provider (MP) and a Media Consumer (MC) to establish a CLUE multimedia telepresence session. The main goals of the CLUE protocol are:

1. enabling a MP to fully announce its current telepresence capabilities to the MC in terms of available media captures, encodings, and simultaneity constraints;

2. enabling a MC to request the desired multimedia streams from the offering MP.

CLUE protocol messages flow upon a DTLS/SCTP/UDP channel established as depicted in [I-D.kyzivat-clue-signaling]. While [I-D.kyzivat-clue-signaling] focuses on protocol signaling details and on its interaction with the SIP/SDP session establishment phase, we herein investigate the protocol in action and try to define the behavior of both the MP and the MC at the CLUE application level. We assume the DTLS/SCTP/UDP channel is established and discuss how the CLUE dialogue between the MP and the MC can be exploited to successfully setup the telepresence session according to the principles and concepts pointed out in in [I-D.ietf-clue-framework].

In Section 3 we provide an overview of the CLUE protocol and describe CLUE messages along with their features and functionality. MC's and MP's state machines are introduced in Section 4 and further described in Section 5 and Section 6 respectively. Versioning, extensions and options management mechanisms are discussed in Section 7, Section 8 and Section 9, respectively. The XML schema defining the CLUE messages is reported in Section 10.

2. Terminology

This document refers to the same terminology used in [I-D.ietf-clue-framework] and in [I-D.kyzivat-clue-signaling].

3. Overview of the CLUE protocol messages

The CLUE protocol, as defined in the following, is a stateful, client-server, XML-based application protocol. The server side of the protocol is represented by a Media Provider, which produces media streams, while the client side is represented by the Media Consumer, which requests media streams.

The MP first advertises the media captures and associated encodings to the MC, as well as possible simultaneity constraints. The description of such telepresence features is made according to the

information defined in the CLUE framework and data model ([I-D.ietf-clue-framework] and [I-D.ietf-clue-data-model-schema]). The CLUE message conveing the MP's multimedia offer is the ADVERTISEMENT message. Such message leverages the XML definitions provided in [I-D.ietf-clue-data-model-schema] for the description of media captures, encodings, and simultaneity constraints features.

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

In the following, a bird's-eye view of the CLUE protocol messages is provided. For each message it is indicated who sends it, who receives it, a brief description of the information it carries, and how/when it is used. Besides ADVERTISEMENT and CONFIGURE, new messages have been conceived in order to provide all the mechanisms and operations envisaged in [I-D.kyzivat-clue-signaling].

- ADVERTISEMENT (ADV)
- CONFIGURE (CONF)
- RESPONSE
- RE-ADV
- OPTIONS

3.1. ADVERTISEMENT

FROM	MP
TO	MC
TYPE	Notification
DESCRIPTION	This message is used by the MP to advertise the available media captures and related information to the MC. The ADV contains elements compliant with the CLUE data model and other information like the CLUE protocol version and a sequence number.
USAGE	The MP sends this message as soon as the CLUE channel is ready. The MP sends an ADV to the MC each time there is a modification of the MP's telepresence capabilities. The ADV message is also sent back to the MC when the MP receives a RE-ADV request.

The ADV message is considered a notification since, during the session, it can be sent from the MP also on a per-event basis, i.e. when the CLUE capabilities of the MP change with respect to the last issued ADV. It is still to be discussed if 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].

3.2. CONFIGURE

FROM	MC
TO	MP
TYPE	Request
DESCRIPTION	This message allows a MC to ask for the desired (advertised) capture. It contains capture encodings and other information like the CLUE protocol version and a sequence number.
USAGE	The MC can send a CONF after the reception of an ADV or each time it wants to request other advertised captures from the MP.

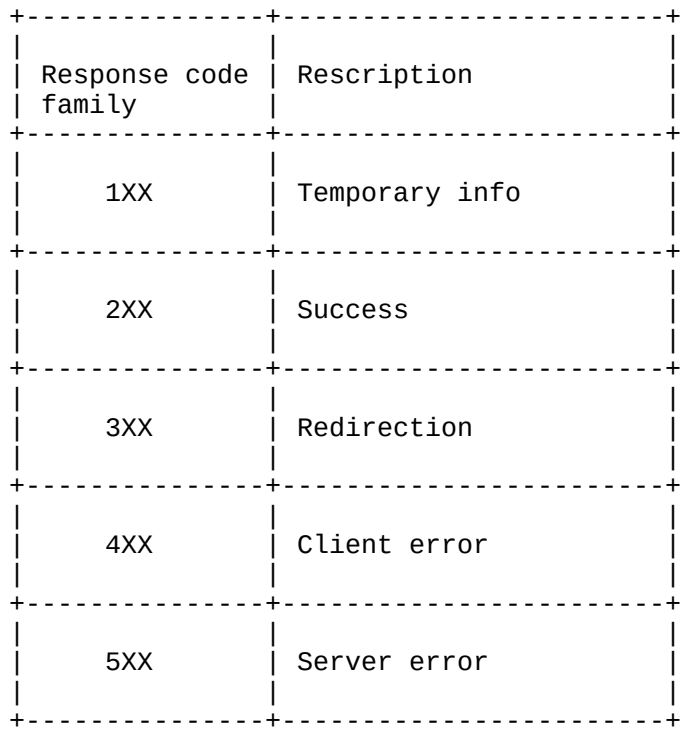
3.3. RESPONSE

FROM	MP
TO	MC
TYPE	Response
DESCRIPTION	This message allows a MP to answer to a CONF message. Besides the protocol version and a sequence number, it contains a response code with a response string indicating either the success or the failure (along with failure details) of a CONF request elaboration. Example response codes and strings are provided in the following table.
USAGE	The MP sends this message in response to a CONF message.

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	Rescription
1XX	Temporary info
2XX	Success
3XX	Redirection
4XX	Client error
5XX	Server error

3.4. RE-ADV

FROM	MC
TO	MP
TYPE	Request
DESCRIPTION	This message allows a MC to request a MP to issue a new copy of the ADV. This message can contain a reason string indicating the motivation for the request (e.g., refresh, missing elements in the received ADV, wrong syntax in the received ADV, invalid capture area, invalid line of capture point, etc).
USAGE	The MC sends this message to the MP when the timeout for the ADV is fired, or when the ADV is not compliant with the CLUE specifications (this can be useful for interoperability testing purposes)

3.5. OPTIONS

ToDo. See Section 9.

4. Protocol state machines

The CLUE protocol is an application protocol used between a Media Provider (MP) and a Media Consumer (MC) in order to establish a multimedia telepresence session. CLUE protocol messages flow upon 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 MC process and with the MP process.

5. Media Consumer's state machine

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

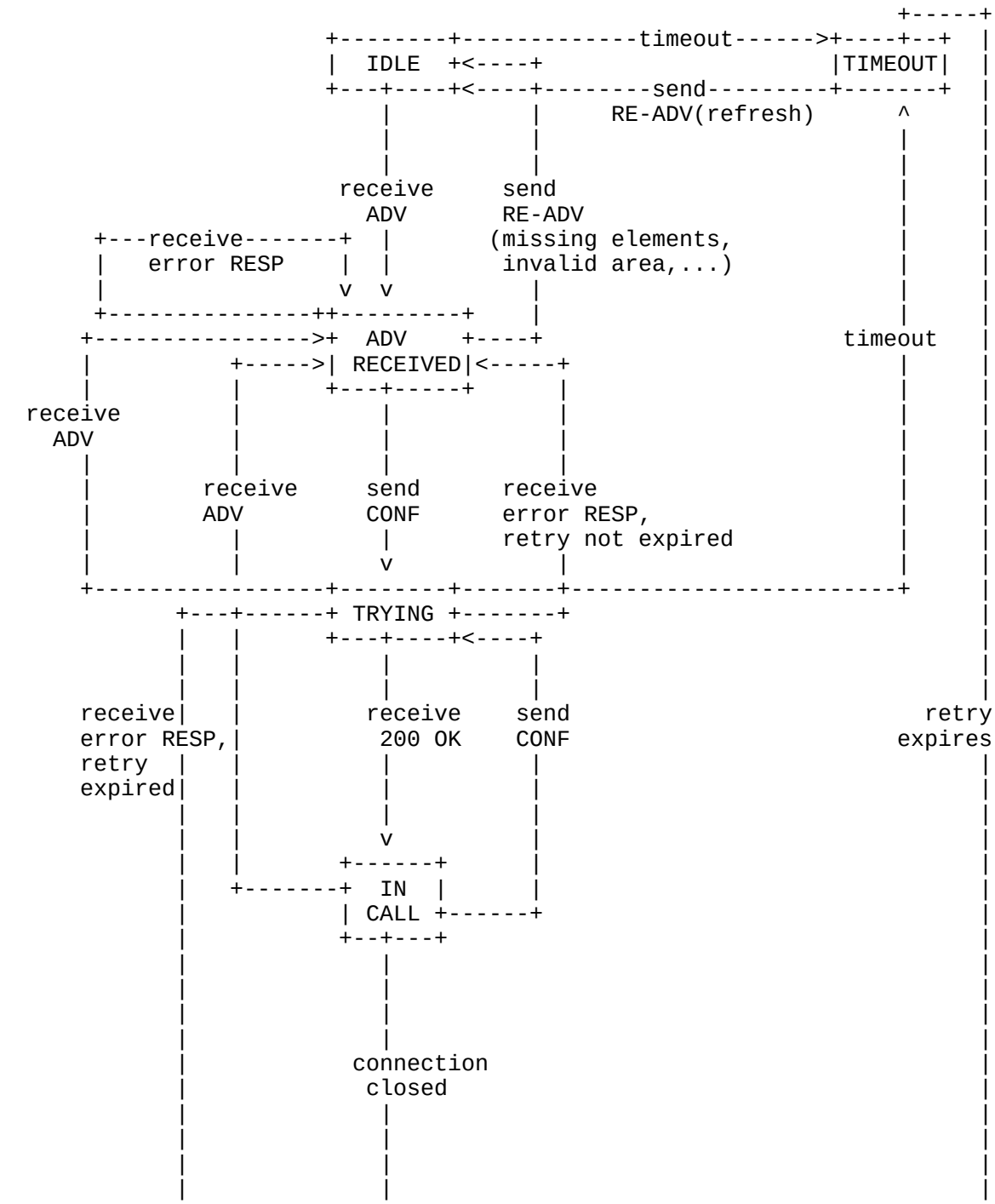
In the TIMEOUT state, if the number of trials is below the retry threshold, the MC sends a RE-ADV/refresh message to the MP ("send RE-ADV"), switching back to the IDLE state. Otherwise, the MC moves to the TERMINATED state.

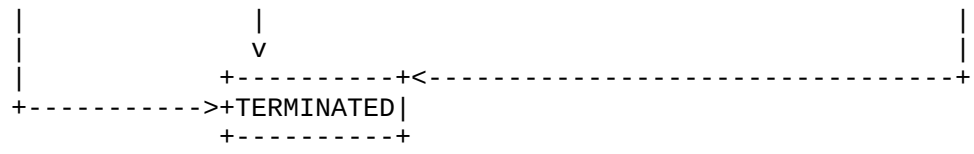
When the ADV has been received ("receive ADV"), the MC goes into the ADV RECEIVED state. The ADV is then parsed. If something goes wrong with the ADV (bad syntax, missing XML elements, etc.), the MC sends a RE-ADV message to the MP specifying the encountered problem via a proper reason phrase. In this way, the MC switches back to the IDLE state, waiting for a new copy of the ADV. If the ADV is successfully processed, the MC issues a CONF message towards the MP ("send CONF") and switches to the TRYING state.

While in the TRYING state, the MC is waiting for a RESPONSE message (to the issued CONF) from the MP. If the timeout expires ("timeout"), the MC moves to the TIMEOUT state and sends a RE-ADV in order to solicit a new ADV from the MP. If a RESPONSE with an error code is received ("receive 4xx, 5xx not supported"), then the MC moves back to the ADV-RCVD state and produces a new CONF message to be sent to the MP. If a successful RESPONSE arrives ("receive 200 OK"), the MC gets into the IN CALL state. If a new ADV arrives in the meanwhile, it is ignored. Indeed, as soon as the timeout expires, the MC switches to the TIMEOUT state and then sends a RE-ADV to the MP.

When the MC is in the IN CALL state, it means that the telepresence session 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-RCV 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 transitions have not been reported for the sake of simplicity. This termination condition is a temporary solution.


IDLE
timeout
TIMEOUT
send
RE-ADV(refresh)
receive
ADV
send
RE-ADV
(missing elements,
invalid area,...)
receive
error RESP
ADV
RECEIVED
timeout
receive
ADV
receive
ADV
send
CONF
receive
error RESP,
retry not expired
TRYING
receive
error RESP,
retry
expired
receive
200 OK
send
CONF
retry
expires
IN
CALL
connection
closed



6. Media Provider's state machine

In the IDLE 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 CONF state.

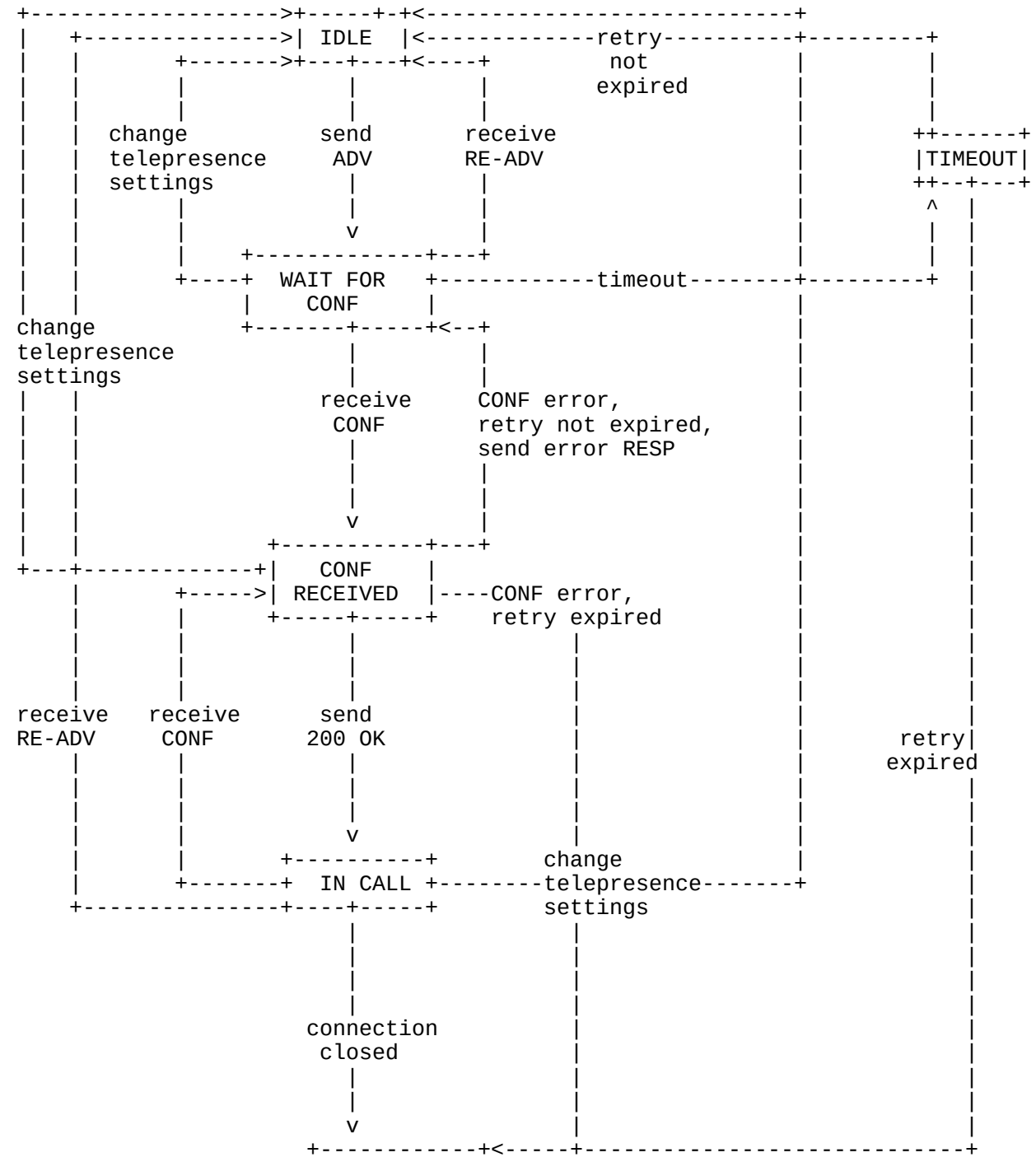
When in the WAIT FOR CONF state, the MP is listening to the channel for a CONF coming from the MC. If a RE-ADV is received, the MP goes back to the IDLE state and issues an ADV again. If telepresence settings change in the meanwhile, it moves back to the IDLE state too, and prepares a new ADV to be sent 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 IDLE 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 RESPONSE message. If the MP is fine with the MC's configuration, then it sends back a 200 OK successful RESPONSE and moves to the IN CALL state. If there are errors duting CONF processing, then the MC returns a RESPONSE carrying an error response code. Finally, if there are changes in the telepresence settings, it goes back to the IDLE state to issue an updated ADV.

When in the IN CALL state, the MP has successfully set up 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 IDLE 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.


IDLE
retry
not expired
change telepresence settings
send ADV
receive RE-ADV
TIMEOUT
WAIT FOR CONF
timeout
change telepresence settings
receive CONF
CONF error, retry not expired, send error RESP
CONF RECEIVED
CONF error, retry expired
receive RE-ADV
receive CONF
send 200 OK
retry expired
IN CALL
change telepresence settings
connection closed

7. About CLUE protocol XML schema 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.

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 subsequet versions of the XML schema should be backward compatible. 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 5.0 of the protocol while the MC only understands version 4.0, the MP should have no problem in reverting the dialogue to version 4.0 without exploiting 5.0 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.

Even though strongly discouraged, if a future version of the protocol is designed which breaks the backward compatibility constraint, this ascpect MUST be explicitely advertised in the corresponding new RFC document. In such a case, it would up to developers to update their systems accordingly.

8. Extensibility issues

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 in the future. Such needs may relate to two main aspects of the protocol:

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.

8.1. Aspect 1 - new information within existing messages

CLUE messages are envelopes carrying two types of information:

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

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. If brand new data model elements are needed, then there are three options:

1. writing down a new version of the data model schema, with the new elements added after the existing ones. This is a possible solution. However, we must state that telepresence applications are forced to check the version attribute of the schema they use.

2. putting all the new elements inside a brand new schema to be linked to a new URN that the most up to date telepresence system must be aware of.

3. designing a wildcard envelope for future data model elements.

8.2. Aspect 2 - new messages

New CLUE protocol messages, not envisioned in the standard version of the schema, are needed. Also in that case we have three chances:

writing down a new version of the protocol schema, with the new messages added after the existing ones. The same considerations of the first option above hold here.

putting all the new messages inside a brand new schema to be linked to a new URN that the most up to date telepresence system must be aware of. [Editors' note: we strongly dislike this option!!]

designing a wildcard envelope for future messages. This is an approach used also within the CCMP protocol (Centralized Conferencing Manipulation Protocol, [RFC6503]). In that case, a mechanism for the extension negotiation is also envisioned.

9. Managing protocol version negotiation and extensions: the OPTIONS request

In this section we provide a mechanism for handling protocol extension matters as those pointed out in the previous section, as well as version negotiation issues.

We propose a new request message issued by the MC to the MP as soon as the CLUE channel is instatiated: the OPTIONS request. This message carries:

the CLUE protocol version spoken by the MC

the data model extensions supported by the MC

the protocol extensions supported by the MC

When the MP receives the OPTIONS message, it reads the CLUE protocol

version of the MC (the highest protocol version of the MC). If the MC's version is higher than the MP's one, then the MP responds to the MC by using in the RESPONSE message its version. The MC has to downgrade the CLUE dialogue to the version specified by the MP in the subsequent CLUE messages. If the MC's version is equal to (case i) or lower than (case ii) the MP version, then the MP will use in the RESPONSE message the same version as the one in hthe OPTIONS message and all subsequent CLUE messages must carry that version number. In the (ii) case, it is the MP who has to to downgrade the CLUE dialogue in order to be understood by the MC.

A data model extension is a set of XML definitions related to the description of telepresence capabilities that is contained in an XML schema and which is different from the normative CLUE data model schema. Such XML definitions can represent further entities not envisioned in the CLUE framework at the time of writing of the data model draft. The entities defined in a data model extension can appear in place of the <any> and <anyAttribute> elements included in the data model document. A data model extension is then represented by a reference to the defining XML schema. The schema reference is represented by a URI defining the schema location. [TBC] If a data model extension is supported by both a MP and a MC, it means that both are aware of the associated XML schema and of the meanings of the elements defined within it.

A protocol extension is a set of XML definitions related to the CLUE protocol that is contained in an XML schema which is different from the normative CLUE protocol schema. Such definitions can represent: (i) information to be carried within the existing messages in place of <any> and <anyAttribute> elements; (ii) new messages designed for the CLUE telepresence control. Such XML definitions refer to information not envisioned during the CLUE protocol design phase. A protocol extension is then represented by a reference to the defining XML schema. If a protocol extension is supported by both a MP and a MC, it means that both are aware of the associated XML schema and of the meanings of the elements defined within it.

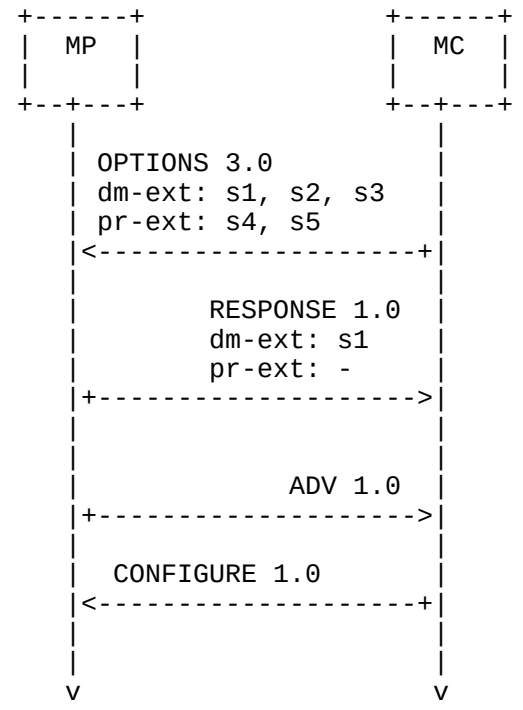
When the MP receives the MC's OPTIONS message, it selects the data model extensions and the protocol extensions that it is able to support, and then provides them into the RESPONSE message back to the MC. Only the extensions included in the RESPONSE message can be used during the telepresence session.

The XML schema definition of the OPTIONS message is provided in the following.

```
<!-- CLUE OPTIONS REQUEST -->
<xs:complexType name="optionsMessageType">
<xs:complexContent>
<xs:extension base="clueRequestMessageType">
<xs:sequence>
<!-- optional fields -->
<xs:element ref="options" minOccurs="0"/>
<xs:any namespace="##other"
processContents="lax" minOccurs="0"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- CLUE OPTIONS -->
<xs:element name="options" type="optionsType"/>

<xs:complexType name="optionsType">
<xs:sequence>
<xs:element name="dm-exts" type="schemaRefList" minOccurs="0" maxOccurs="1"/>
<xs:element name="protocol-exts" type="schemaRefList" minOccurs="0"
            maxOccurs="1"/>
</xs:sequence>
</xs:complexType>

<!-- SCHEMA REF LIST TYPE -->
<xs:complexType name="schemaRefList">
<sequence>
<element name="schemaRef" type="xs:anyURI" maxOccurs="unbounded"/>
</sequence>
</xs:complexType>
```

9.1. An example using OPTIONS

An example of OPTIONS dialogue is provided in the following.



When the CLUE channel is ready, the MC issues an OPTIONS request to the MP. The MC uses the 3.0 version of the CLUE protocol, and supports schemas s1, s2, s3 as data model extensions and schemas s4, s5 as protocol extensions.

The MP speaks the 1.0 version of the CLUE protocol and supports only the first data model extension among those indicated by the MC. It then issues a v. 1.0 RESPONSE to the MC copying only the supported option. The MC is able to understand that it can use only the 1.0 version of the protocol and the s1 extension.

10. XML Schema of CLUE protocol messages

In this section we paste the XML schema defining the ADVERTISEMENT, CONFIGURE and RESPONSE messages contained in [I-D.kyzivat-clue-signaling]. At the time of writing, it assumes that encodings are described in SDP as m-lines with a text identifier, and that the identifier has the same value as the encodingIDs embedded in the <encodingGroups>. However, that assumption is still under discussion in the context of the CLUE-SDP coupling issues.

```
<?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="clue-data-model-01.xsd"/>

<!-- ELEMENT DEFINITIONS -->
<xs:element name="response" type="responseMessageType"/>
<xs:element name="advertisement" type="advertisementMessageType"/>
<xs:element name="configure" type="configureMessageType"/>
<xs:element name="readv" type="readvMessageType"/>
<xs:element name="options" type="optionsMessageType"/>

<!-- CLUE MESSAGE TYPE -->
<xs:complexType name="clueMessageType" abstract="true">
<xs:sequence>
<!-- mandatory fields -->
<!-- TBS: version info -->
</xs:sequence>
</xs:complexType>

<!-- CLUE REQUEST MESSAGE TYPE -->
<xs:complexType name="clueRequestMessageType" abstract="true">
<xs:complexContent>
<xs:extension base="clueMessageType">
<xs:sequence>
<!-- mandatory fields -->
<xs:element name="requestNumber" type="xs:integer"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- CLUE OPTIONS REQUEST -->
<xs:complexType name="optionsMessageType">
<xs:complexContent>
<xs:extension base="clueRequestMessageType">
<xs:sequence>
<!-- optional fields -->
```

```
<xs:element ref="options" minOccurs="0"/>
<xs:any namespace="##other"
processContents="lax" minOccurs="0"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- CLUE OPTIONS -->
<xs:element name="options" type="optionsType"/>

<xs:complexType name="optionsType">
<xs:sequence>
<xs:element name="dm-exts" type="schemaRefList" minOccurs="0" maxOccurs="1"/>
<xs:element name="protocol-exts" type="schemaRefList" minOccurs="0"
           maxOccurs="1"/>
</xs:sequence>
</xs:complexType>

<!-- SCHEMA REF LIST TYPE -->
<xs:complexType name="schemaRefList">
<sequence>
<element name="schemaRef" type="xs:anyURI" maxOccurs="unbounded"/>
</sequence>
</xs:complexType>

<!-- CLUE NOTIFICATION MESSAGE TYPE -->
<xs:complexType name="clueNotificationMessageType" abstract="true">
<xs:complexContent>
<xs:extension base="clueMessageType">
<xs:sequence>
<!-- mandatory fields -->
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>


<!-- CLUE RESPONSE MESSAGE TYPE -->
<xs:complexType name="clueResponseMessageType">
<xs:complexContent>
<xs:extension base="clueMessageType">
<xs:sequence>
<!-- mandatory fields -->
<xs:element name="requestNumber" type="xs:integer"/>
<xs:element name="reason" type="reasonType" minOccurs="1"/>
<!-- optional fields -->
<xs:any namespace="##other"
```

```
processContents="lax" minOccurs="0"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- RESPONSE MESSAGE TYPE -->
<xs:complexType name="responseMessageType">
<xs:complexContent>
<xs:extension base="clueRequestMessageType">
<xs:sequence>
<!-- mandatory fields -->
<!-- TBD. -->
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>


<!-- CLUE ADVERTISEMENT MESSAGE TYPE -->
<xs:complexType name="advertisementMessageType">
<xs:complexContent>
<xs:extension base="clueNotificationMessageType">
<xs:sequence>
<!-- mandatory fields -->
<xs:element name="advNumber" type="xs:unsignedInt"/>
<xs:element name="mediaCaptures"
type="dm:mediaCapturesType"/>
<xs:element name="encodingGroups"
type="dm:encodingGroupsType"/>
<!-- The encodings are defined via identifiers in the SDP,
referenced in encodingGroups -->
<xs:element name="captureScenes"
type="dm:captureScenesType"/>
<!-- optional fields -->
<xs:element name="simultaneousSets"
type="dm:simultaneousSetsType" minOccurs="0"/>
<xs:any namespace="##other"
processContents="lax" minOccurs="0"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>


<!-- CLUE CONFIGURE MESSAGE TYPE -->
<xs:complexType name="configureMessageType">
<xs:complexContent>
```

```
<xs:extension base="clueRequestMessageType">
<xs:sequence>
<!-- mandatory fields -->
<xs:element name="advNumber" type="xs:unsignedInt"/>
<!-- optional fields -->
<xs:element name="captureEncodings"
type="dm:captureEncodingsType" minOccurs="0"/>
<xs:any namespace="##other"
processContents="lax" minOccurs="0"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- REASON TYPE -->
<xs:complexType name="reasonType">
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute type="xs:short" name="code" use="required"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>


</xs:schema>
```

11. Examples

TBD

12. Handling channel errors

TBD

13. Diff with the -01 version

XML Schema moved here from [I-D.kyzivat-clue-signaling]

advNumber introduced to couple a configure with an advertisement message

added introductory text

added a version and extension negotiation proposal

14. Diff with -00 version

MC and MP state diagrams have been updated for discussion in http://www.ietf.org/mail-archive/web/clue/current/msg02650.html

Consideration about protocol extension and versioning have been added to foster discussion.

15. 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-00 (work in progress), July 2013.

[I-D.ietf-clue-framework] Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-11 (work in progress), July 2013.

[I-D.ietf-clue-telepresence-requirements] Romanow, A., Botzko, S., and M. Barnes, "Requirements for Telepresence Multi-Streams", draft-ietf-clue-telepresence-requirements-05 (work in progress), August 2013.

[I-D.kyzivat-clue-signaling] Kyzivat, P., Xiao, L., Groves, C., and R. Hansen, "CLUE Signaling", draft-kyzivat-clue-signaling-05 (work in progress), September 2013.

[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

CLUE Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 10, 2014

R. Presta
S. Romano
University of Napoli

May 9, 2014

CLUE protocol
draft-presta-clue-protocol-04

Abstract

The CLUE protocol is an application protocol conceived for the description and negotiation of a CLUE telepresence session. The design of the CLUE protocol takes into account the requirements and the framework defined, respectively, in [I-D.ietf-clue-framework] and [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 November 10, 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. Diff with the -03 version 33
11. Diff with the -02 version 34
12. Acknowledgments 34
13. Informative References 34

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 realiable 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 backword 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 conveing 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:

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

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

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

Each CP should manage uo 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:

- mediaProvider: a mandatory boolean field set to "true" if the CP is able to act as a MP
- mediaConsumer: a mandatory boolean field set to "true" if the CP is able to act as a MC
- supportedVersions: the list of the supported versions
- 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. 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 whithin 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. Finally, the highest commonly supported version number is expressed in the <version> field and just the commonly supported options in 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:string"/>
<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 piggibacking 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="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>
```

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.

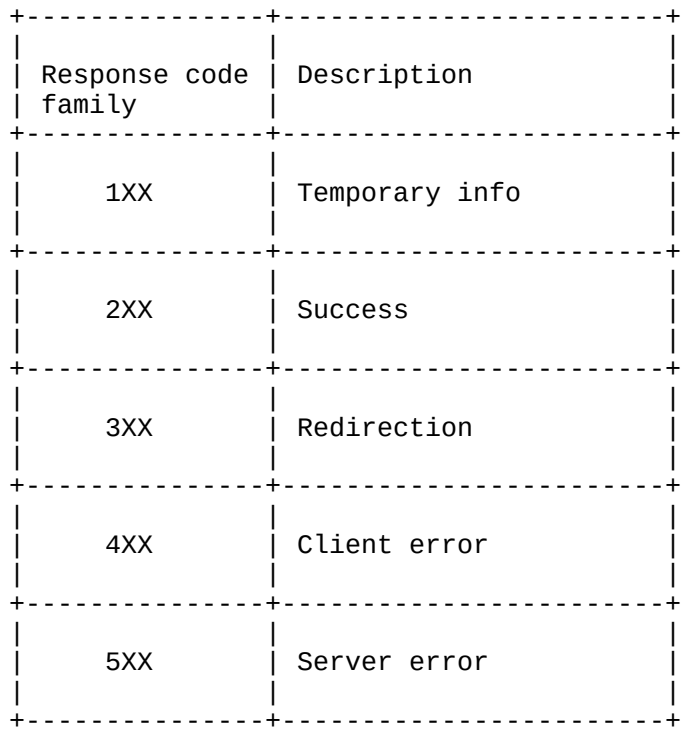
```
<!-- CLUE READV RESPONSE MESSAGE TYPE -->
<xs:complexType name="readvResponseMessageType">
<xs:complexContent>
<xs:extension base="clueMessageType">
<xs:sequence>
<xs:element name="responseCode" type="xs:short"/>
<xs:element name="reasonString" type="xs:string"/>
<xs:element name="readvSequenceNr" 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" minOc
curs="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

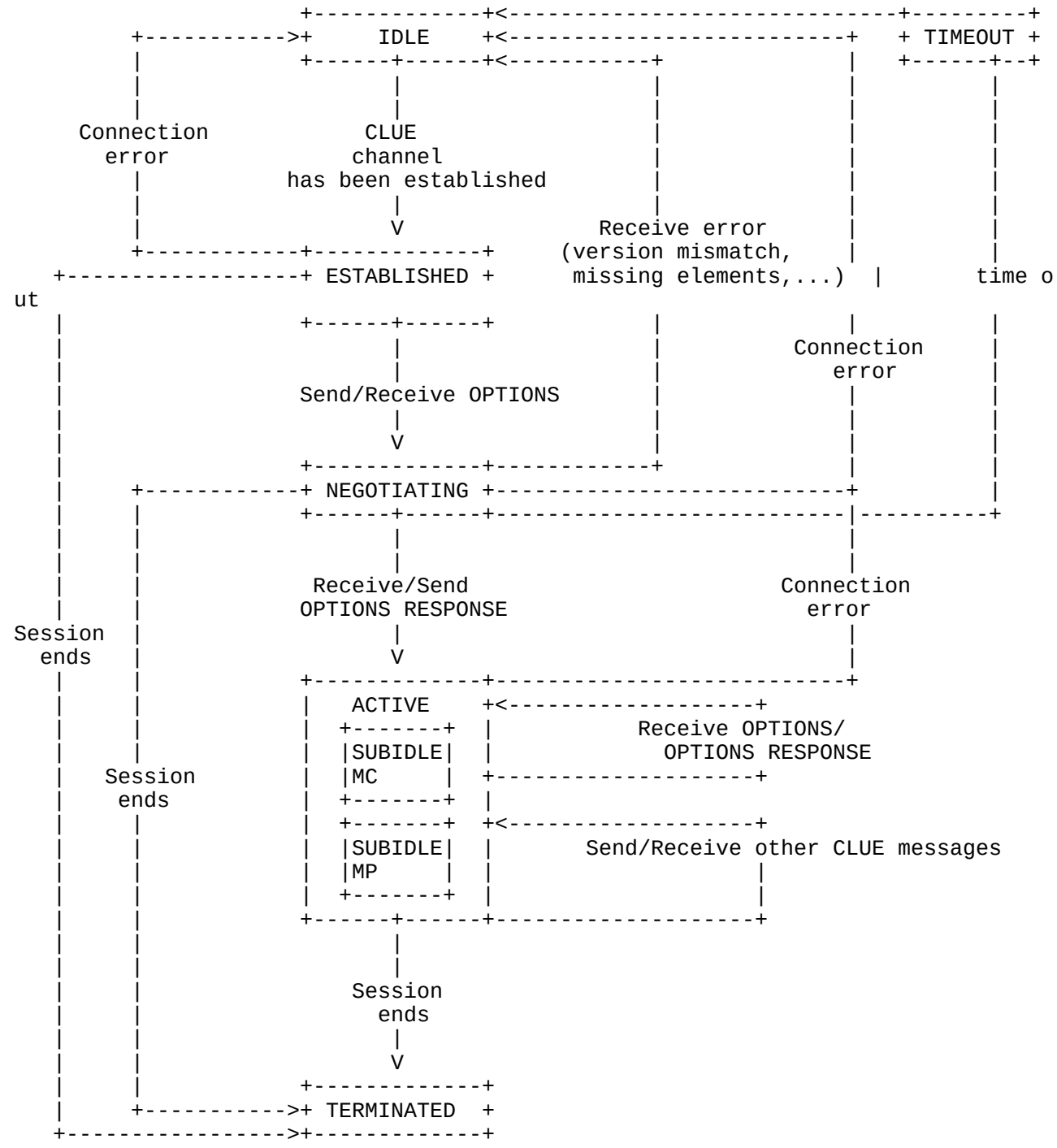
5. Protocol state machines

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]


IDLE
TIMEOUT
Connection error
CLUE channel has been established
Receive error (version mismatch, missing elements,...)
ESTABLISHED
time out
Connection error
Send/Receive OPTIONS
NEGOTIATING
Receive/Send OPTIONS RESPONSE
Connection error
Session ends
ACTIVE
SUBIDLE MC
SUBIDLE MP
Receive OPTIONS/ OPTIONS RESPONSE
Session ends
Send/Receive other CLUE messages
Session ends
TERMINATED

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.

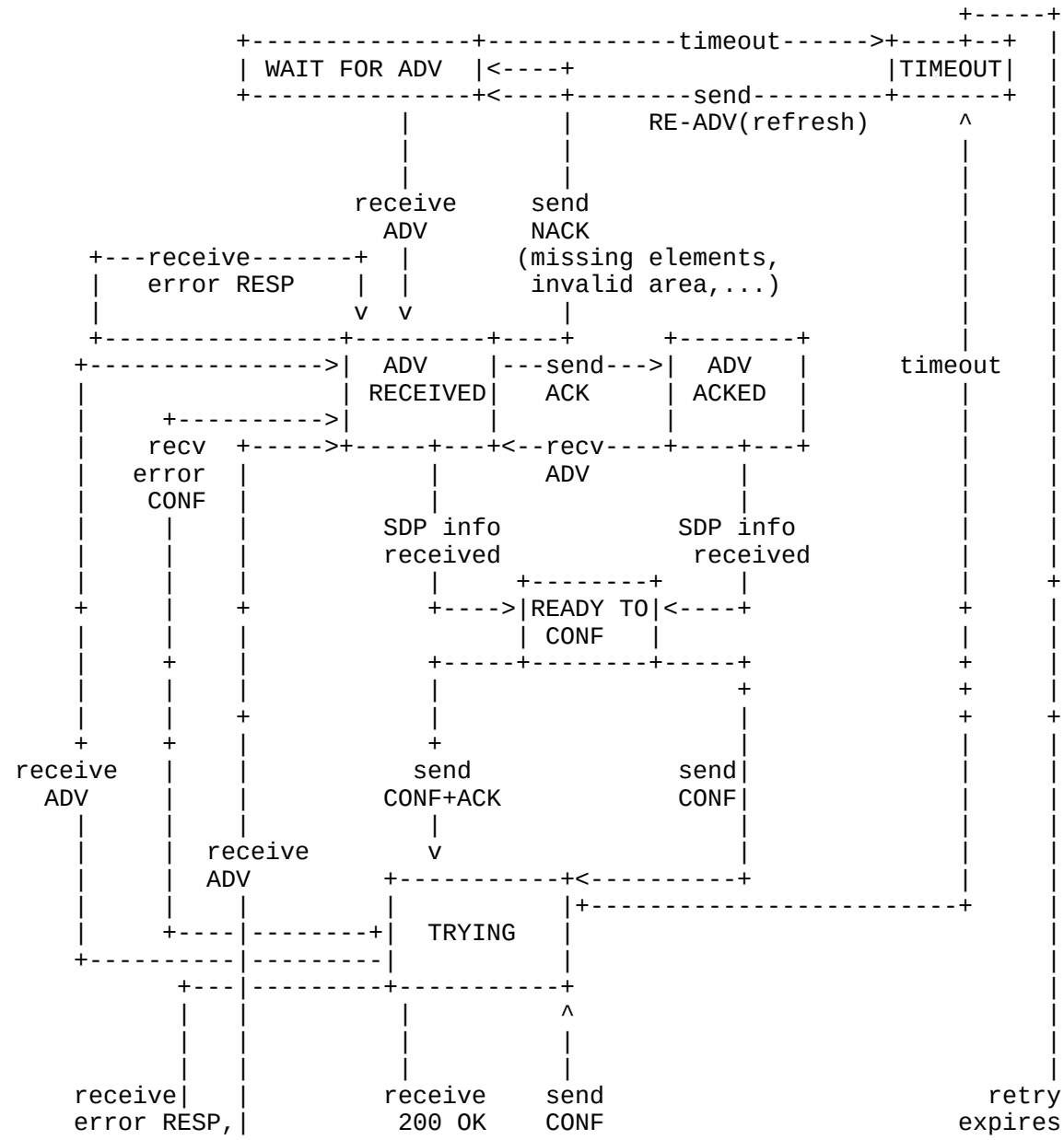
When the ADV has been received ("receive ADV"), the MC goes into the ADV RECEIVED state. The ADV is then parsed. If something goes wrong with the ADV (bad syntax, missing XML elements, etc.), the MC sends a NACK message (an ACK with an error response code) to the MP specifying the encountered problem 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.

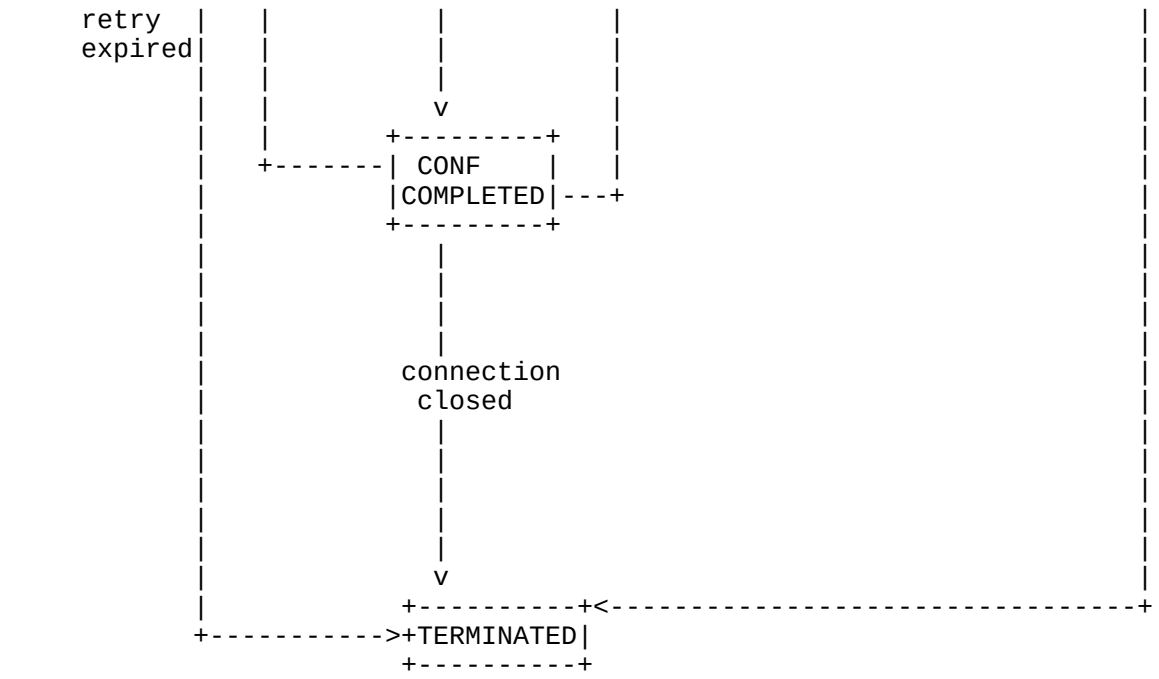
While in the TRYING state, the MC is waiting for a CONF RESPONSE message (to the issued CONF) from the MP. If the timeout expires ("timeout"), the MC moves to the TIMEOUT state and sends a READV in order to solicit a new ADV from the MP. If a CONF RESPONSE with an error code is received ("receive 4xx, 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. 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

transitions have not been reported for the sake of simplicity. This termination condition is a temporary solution.





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.

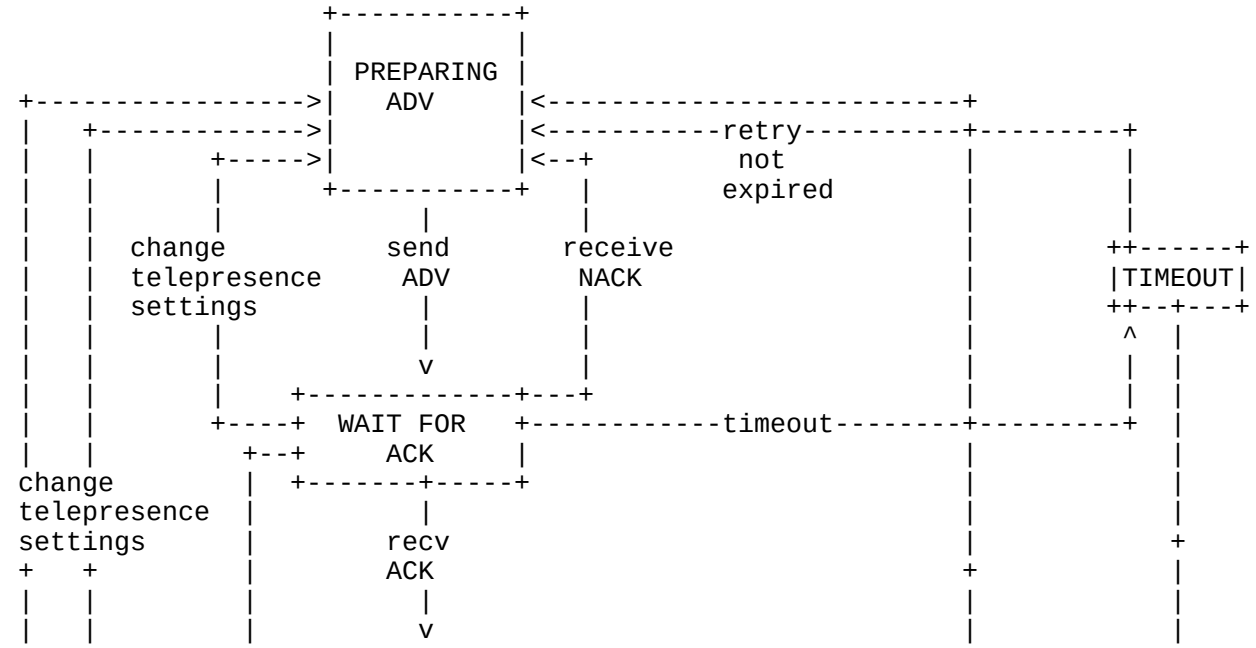
When in the WAIT FOR CONF state, the MP is listening to the channel for a CONF coming from the MC. If a RE-ADV is received, the MP goes back to the IDLE state and issues an ADV again. If telepresence settings change in the meanwhile, it moves back to the PREPARING ADV state and prepares a new ADV to 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.

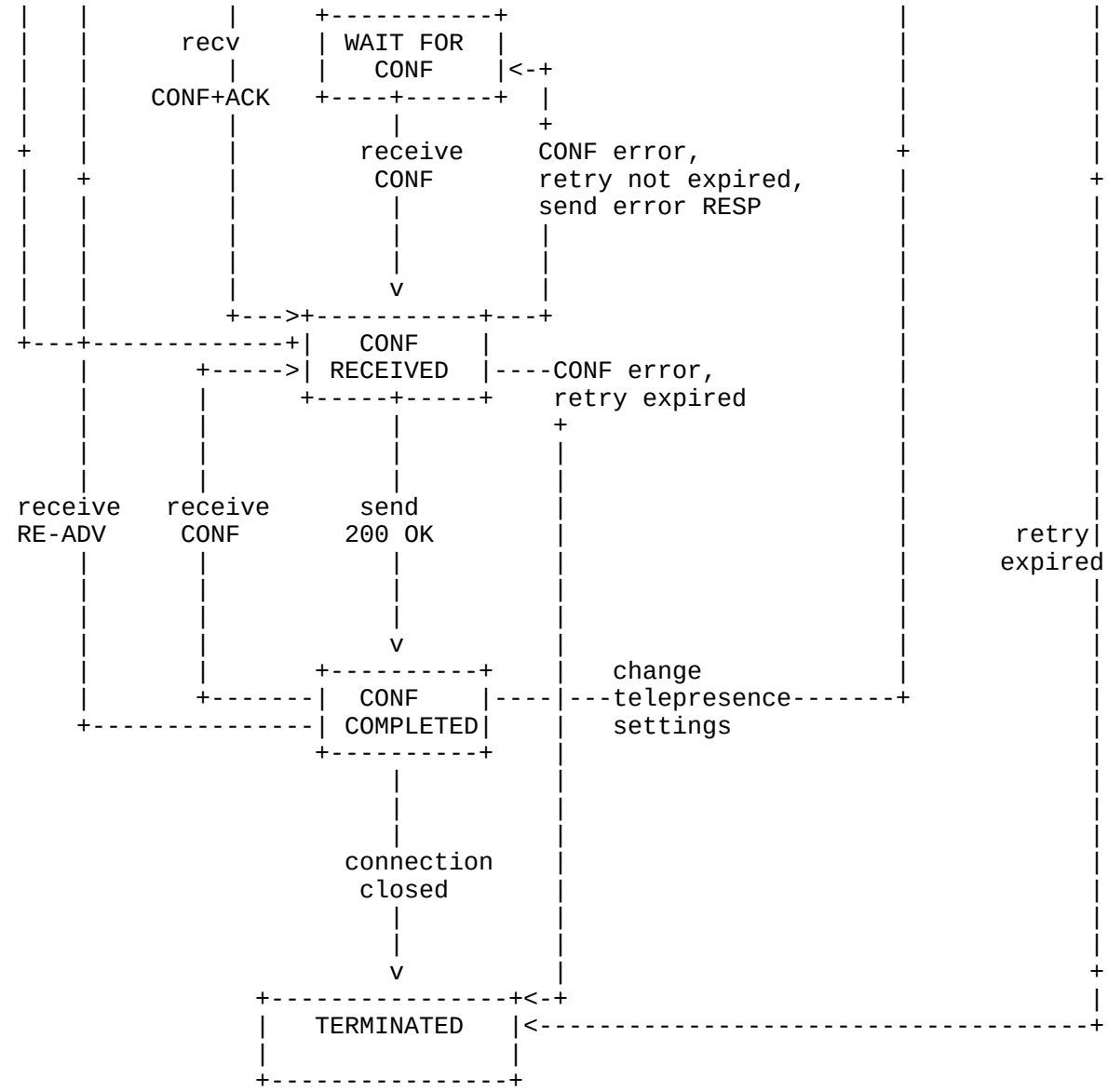
In the TIMEOUT state, if the number of trials does not exceed the retry threshold, the MC comes back to the PREPARING ADV state for sending a new ADV. Otherwise, it goes to the TERMINATED state.

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 duting 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:string"/>
<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:string"/>
<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>

<!-- 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>

<!-- 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" minOc
curs="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. Diff with the -03 version

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.

11. Diff with the -02 version

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.

12. Acknowledgments

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

13. 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-04 (work in progress), March 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-14 (work in progress), February 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