

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 media content 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 as 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 only 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,
Israel

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 dependencies 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 dependencies 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 not 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

- o Reordered main sections, as in the discussion about interdependence of SDP and CLUE is useful to reference the split between CLUE and SDP.
- o Added more detail to the argument of why dependencies should be CLUE->SDP and not the other way around or in both directions.
- o 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 WG
Internet Draft
Intended status: Standards Track
Expires: April 19, 2014

M. Duckworth, Ed.
Polycom
A. Pepperell
Acano
S. Wenger
Vidyo
October 19, 2013

Framework for Telepresence Multi-Streams
draft-ietf-clue-framework-12.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 April 19, 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. Definitions.....	4
4. Overview & Motivation.....	6
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. Capture Scene.....	22
7.2.1. Capture Scene attributes.....	25
7.2.2. Capture Scene Entry attributes.....	25
7.3. Simultaneous Transmission Set Constraints.....	26
8. Encodings.....	28
8.1. Individual Encodings.....	28
8.2. Encoding Group.....	29
9. Associating Captures with Encoding Groups.....	30
10. Consumer's Choice of Streams to Receive from the Provider....	31
10.1. Local preference.....	33
10.2. Physical simultaneity restrictions.....	33
10.3. Encoding and encoding group limits.....	33
11. Extensibility.....	34
12. Examples - Using the Framework (Informative).....	34
12.1. Provider Behavior.....	34
12.1.1. Three screen Endpoint Provider.....	35
12.1.2. Encoding Group Example.....	42
12.1.3. The MCU Case.....	42
12.2. Media Consumer Behavior.....	43
12.2.1. One screen Media Consumer.....	44
12.2.2. Two screen Media Consumer configuring the example..	44

12.2.3. Three screen Media Consumer configuring the example	45
13. Acknowledgements.....	45
14. IANA Considerations.....	45
15. Security Considerations.....	46
16. Changes Since Last Version.....	46
17. Authors' Addresses.....	50

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.

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.

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 7.3.

(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 8.

(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 4.

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.)

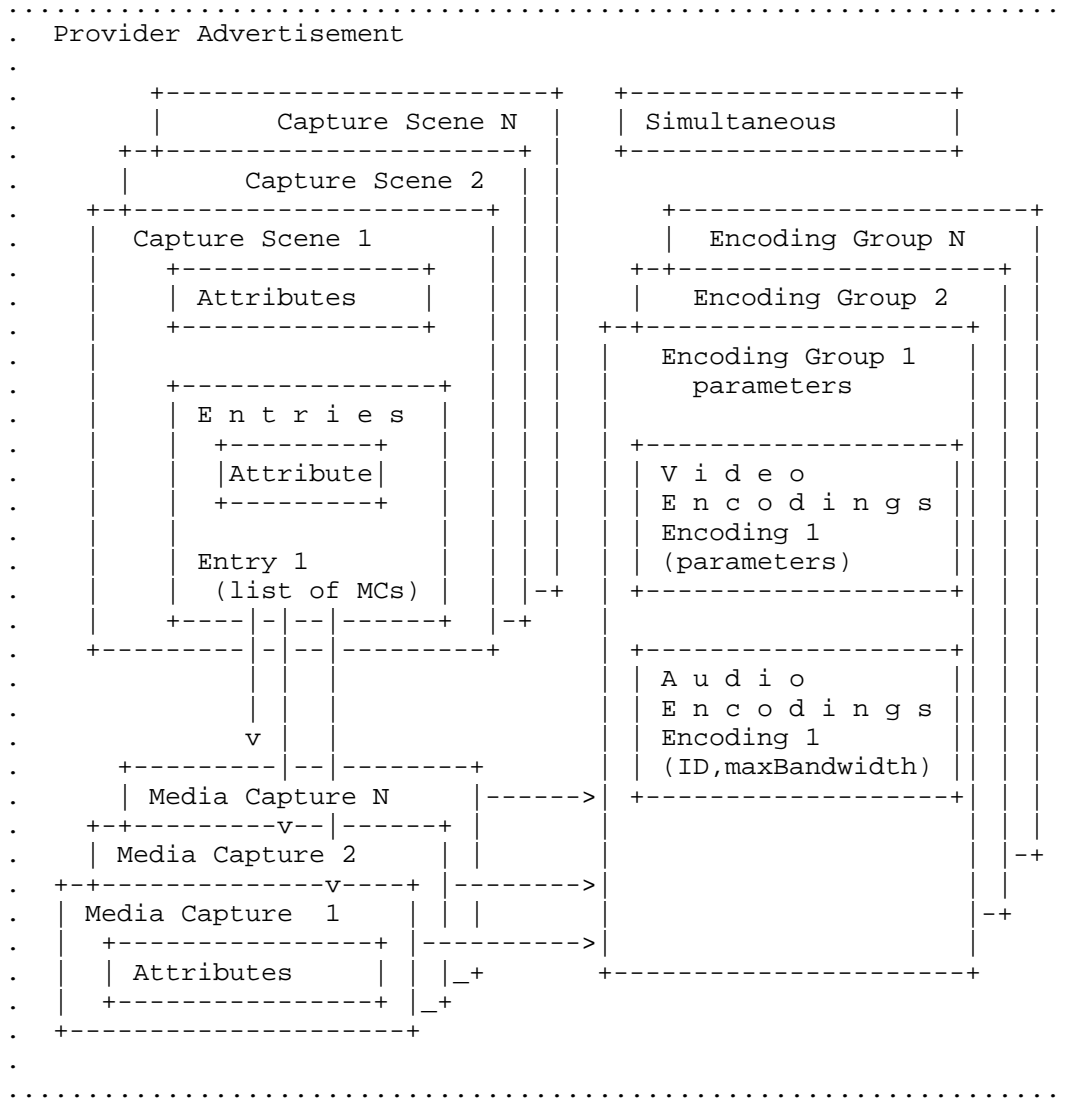
The data model is based around two main concepts: a Capture and an Encoding. A Media Capture (MC), such as audio or video, describes the content a Provider can send. Media Captures are described in terms of CLUE-defined attributes, such as spatial relationships and purpose of the capture. Providers tell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

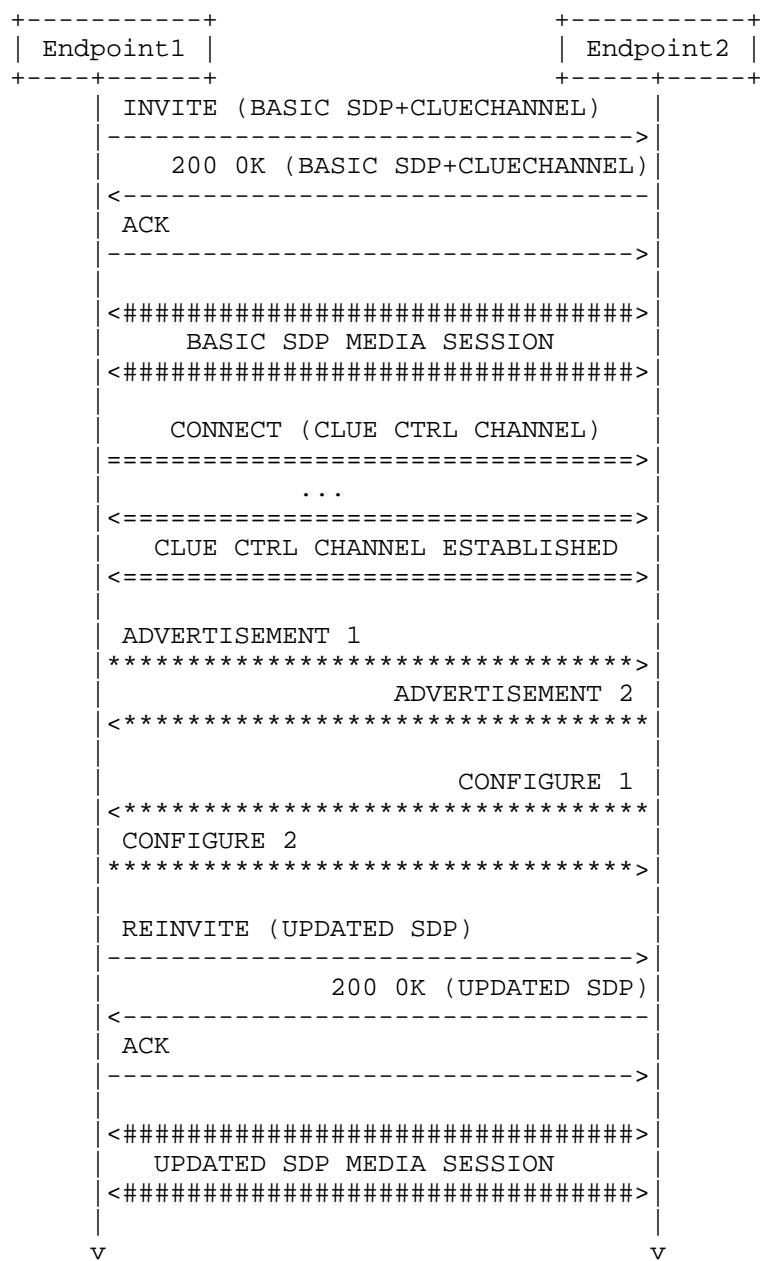
In addition, the Provider can send the Consumer a description of the Individual Encodings it can send in terms of the media attributes of the Encodings, in particular, audio and video parameters such as bandwidth, frame rate, macroblocks per second. Note that this is OPTIONAL, and intended to minimize 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.



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.



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.

Numerous optimizations are possible, and are the implementer's choice. For example, it can be sensible to establish one or more initial media channels during the initial offer/answer exchange, which would allow, for example, for a fast startup of audio. Depending on the system design, it can be possible to re-use this established channel for more advanced media negotiated only by CLUE mechanisms, thereby avoiding further offer/answer exchanges.

Edt. note: The editors are not sure whether the mentioned overloading of established RTP channels using only CLUE messages is possible, or desired by the WG. If it were, certainly there is need for specification work. One possible issue: a Provider which thinks that it can switch, say, a audio codec algorithm by CLUE only, talks to a Consumer which thinks that it has to faithfully answer the Providers Advertisement through a Configure, but does not dare setting up its internal resource until such time it has received its authoritative O/A exchange. Working group input is solicited.

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.

Edt. note (StW): what's written below is likely correct, but is not the result of the introduction of CLUE, but rather the result of a generational overhaul of RTP usage that would have happened with or without CLUE. Suggest to delete the sentences below until begin of section 6. Is having a CLUE RTP Mapping document still the plan? If yes, we should have a real draft and a real reference.

However, some form of RTP multiplexing is likely to be used by CLUE devices. More information about relating RTP flows to CLUE entities is in the CLUE RTP Mapping document.

6. Spatial Relationships

In order for a Consumer to perform a proper rendering, it is often necessary or at least helpful for the Consumer to have received spatial information about the streams it is receiving. CLUE defines a coordinate system that allows Media Providers to describe the spatial relationships of their Media Captures to enable proper scaling and spatially sensible rendering of their streams. The coordinate system is based on a few principles:

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

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

The direction of increasing coordinate values is:

X increases from Camera-Left to Camera-Right

Y increases from Front to back

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

7. Media Captures and Capture Scenes

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

7.1. Media Captures

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

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

To identify and distinguish between multiple instances, video and audio captures are labeled. 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 associated with exactly one Capture Scene
- . 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, role).
- . Control information for use inside the CLUE protocol suite.

Point of Capture:

A field with a single Cartesian (X, Y, Z) point value which describes the spatial location of the capturing device (such as camera).

Point on Line of Capture:

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.

Area of Capture:

A field with a set of four (X, Y, Z) points as a value which describe 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.

Mobility of Capture:

This 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.

Composed:

A boolean field which indicates whether or not the Media Capture is a mix (audio) or composition (video) of streams.

This attribute is useful for a media consumer to avoid nesting a composed video capture into another composed capture or rendering. This attribute is not intended to describe the layout a media provider uses when composing video streams.

Switched:

A boolean field which indicates whether or not the Media Capture represents the (dynamic) most appropriate subset of a 'whole'. What is 'most appropriate' is up to the provider and could be the active speaker, a lecturer or a VIP.

Audio Channel Format:

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.

Max Capture Encodings:

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.

Description:

Human-readable description of the Capture, which could be in multiple languages.

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.

View:

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

Language:

This 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.

Role:

Edt. Note -- this is a placeholder for a role attribute, as discussed in draft-groves-clue-capture-attr. We expect to continue discussing the role attribute in the context of that draft, and follow-on drafts, before adding it to this framework document.

Priority:

This 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.

Embedded Text:

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.

Related To:

This attribute indicates the capture contains additional complementary information related to another capture. The value indicates the other capture to which this capture is providing additional information.

For example, a conferences 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. 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 7.3.

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 7.3.

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.

7.2.1. 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;
- . Scale information (millimeters, unknown, no scale), as described in Section 5.

7.2.2. 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;
- . Scene-switch-policy: {site-switch, segment-switch}

A media provider uses this scene-switch-policy attribute to indicate its support for different switching policies. If a provider supports both policies, it MAY advertise separate Capture Scene Entries containing separate Captures, each entry with a separate scene-switch-policy value. If the provider does not support any of these policies, it MUST omit this attribute.

The "site-switch" policy means all captures are switched at the same time to keep captures from the same endpoint site together. Let's say the speaker is at site A and everyone else is at a "remote" site.

When the room at site A shown, all the camera images from site A are forwarded to the remote sites. Therefore at each receiving remote site, all the screens display camera images from site A.

This can be used to preserve full size image display, and also provide full visual context of the displayed far end, site A. In site switching, there is a fixed relation between the cameras in each room and the displays in remote rooms. The room or participants being shown can be switched from time to time based on, for example, who is speaking or by manual control.

The "segment-switch" policy means different captures can switch at different times, and can be coming from different endpoints. Still using site A as where the speaker is, and "remote" to refer to all the other sites, in segment switching, rather than sending all the images from site A, only the image containing the speaker at site A is shown. The camera images of the current speaker and previous speakers (if any) are forwarded to the other sites in the conference.

Therefore the screens in each site are usually displaying images from different remote sites - the current speaker at site A and the previous ones. This strategy can be used to preserve full size image display, and also capture the non-verbal communication between the speakers. In segment switching, the display depends on the activity in the remote rooms - generally, but not necessarily based on audio / speech detection.

7.3. 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). 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 1: 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.

8. 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.

8.1. 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;
- . 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.

8.2. 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.

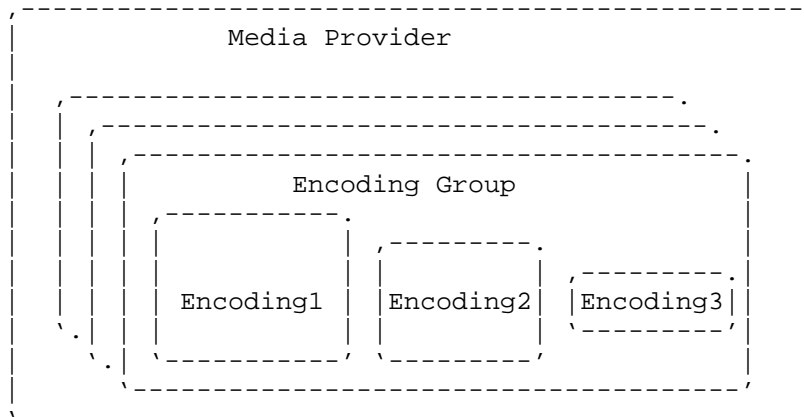


Figure 1: Encoding Group Structure

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. Associating Captures with Encoding Groups

Every Capture **MUST** be associated with at least one Encoding Group, which is used to instantiate that Capture into one or more Capture Encodings. 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. Each Capture has an Encoding Group ID attribute which specifies which Individual Encodings are available to be used for that Capture.

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.

Edt. Note (StW): is the sentence below still correct?

Even more advanced devices MAY choose to establish media streams without an offer-answer exchange, for example by overloading existing 5 tuple connections with the negotiated media.

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

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

When choosing which Media Streams to receive from the Provider, and the encoding characteristics of those Media Streams, the Consumer advantageously takes several things into account: its local preference, simultaneity restrictions, and encoding limits.

10.1. Local preference

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

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

10.2. Physical simultaneity restrictions

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

10.3. Encoding and encoding group limits

Each of the Provider's encoding groups has limits on bandwidth and computational complexity, and the constituent potential encodings have limits on the bandwidth, computational complexity, video frame rate, and resolution that can be provided. When choosing the Captures to be received from a Provider, a Consumer device MUST ensure that the encoding characteristics requested for each individual Capture fits within the capability of the encoding it is being configured to use, as well as ensuring that the combined encoding characteristics for Captures fit within the capabilities of their associated encoding groups. In some cases, this could cause an otherwise "preferred" choice of capture encodings to be passed over in favor of different Capture Encodings--for instance,

if a set of three Captures could only be provided at a low resolution then a three screen device could switch to favoring a single, higher quality, Capture Encoding.

11. Extensibility

One important characteristics of the Framework is its extensibility. 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:

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

12.1.1.1. Three screen Endpoint Provider

Consider an Endpoint with the following description:

3 cameras, 3 displays, a 6 person table

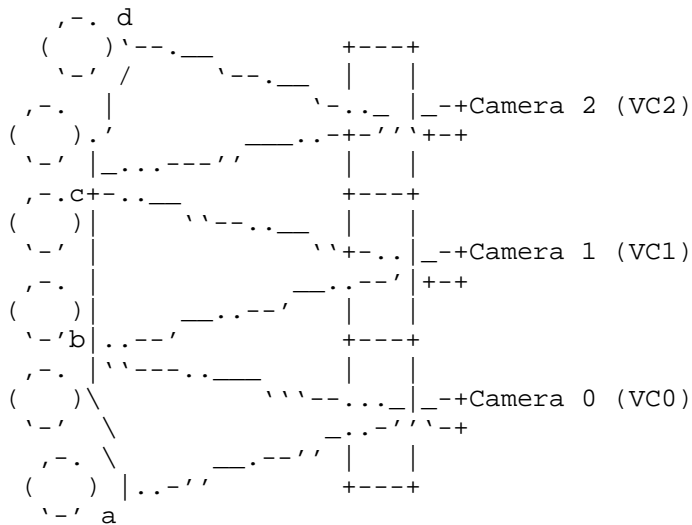
- o Each camera can provide one Capture for each 1/3 section of the table
- o A single Capture representing the active speaker can be provided (voice activity based camera selection to a given encoder input port implemented locally in the Endpoint)
- o A single Capture representing the active speaker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

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

Video Captures:

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



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:

- o AC0 (camera-left), encoding group=EG3, content=main, channel format=mono
- o AC1 (camera-right), encoding group=EG3, content=main, channel format=mono
- o AC2 (center) encoding group=EG3, content=main, channel format=mono
- o AC3 being a simple pre-mixed audio stream from the room (mono), encoding group=EG3, content=main, channel format=mono
- o 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 can not 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 2: 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 3: Example Encoding Group for Audio

Capture Scenes:

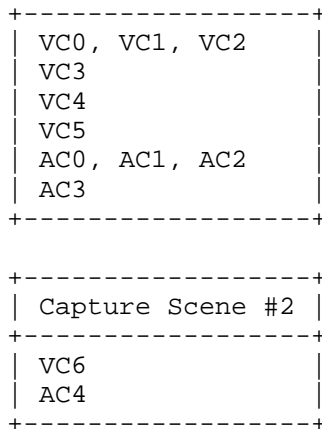
The following table represents the capture scenes for this provider. Recall that a capture scene is composed of alternative capture scene entries covering the same spatial region. Capture Scene #1 is for the main people captures, and Capture Scene #2 is for presentation.

Each row in the table is a separate Capture Scene Entry

```

+-----+
| Capture Scene #1 |

```



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).

12.1.1.2. 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
```

12.1.1.3. 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	note
VC0	video capture for single screen consumer
VC1, VC2	video capture for 2 screen consumer
VC3, VC4, VC5	video capture for 3 screen consumer
VC6, VC7, VC8, VC9	video capture for 4 screen consumer
AC0	audio capture representing all participants

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

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).

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

TBD

16. Changes Since Last Version

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

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
Acana

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 example	55
12.3. Multipoint Conference utilizing Multiple Content Captures	55
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.

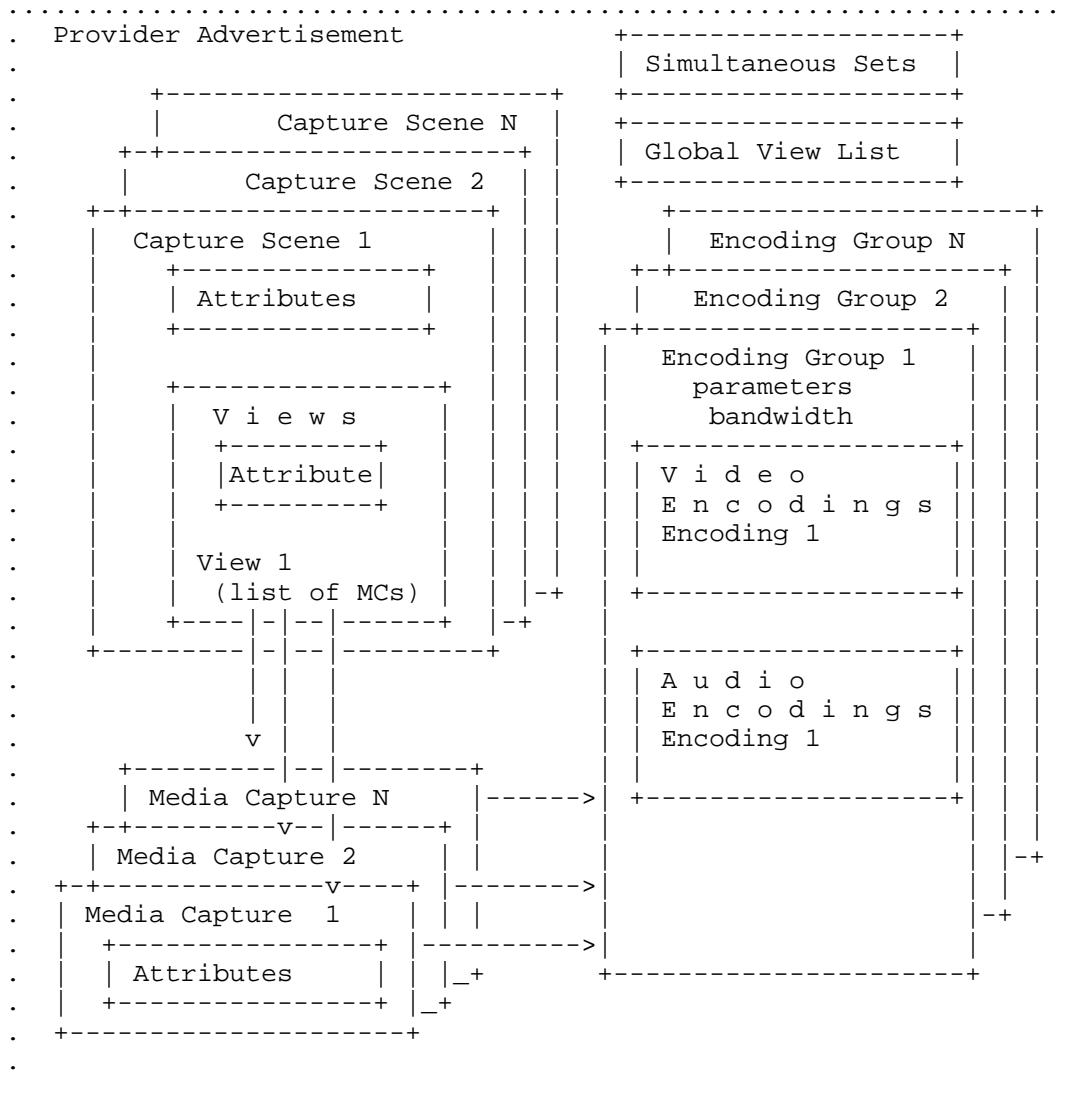


Figure 1: Advertisement Structure

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.

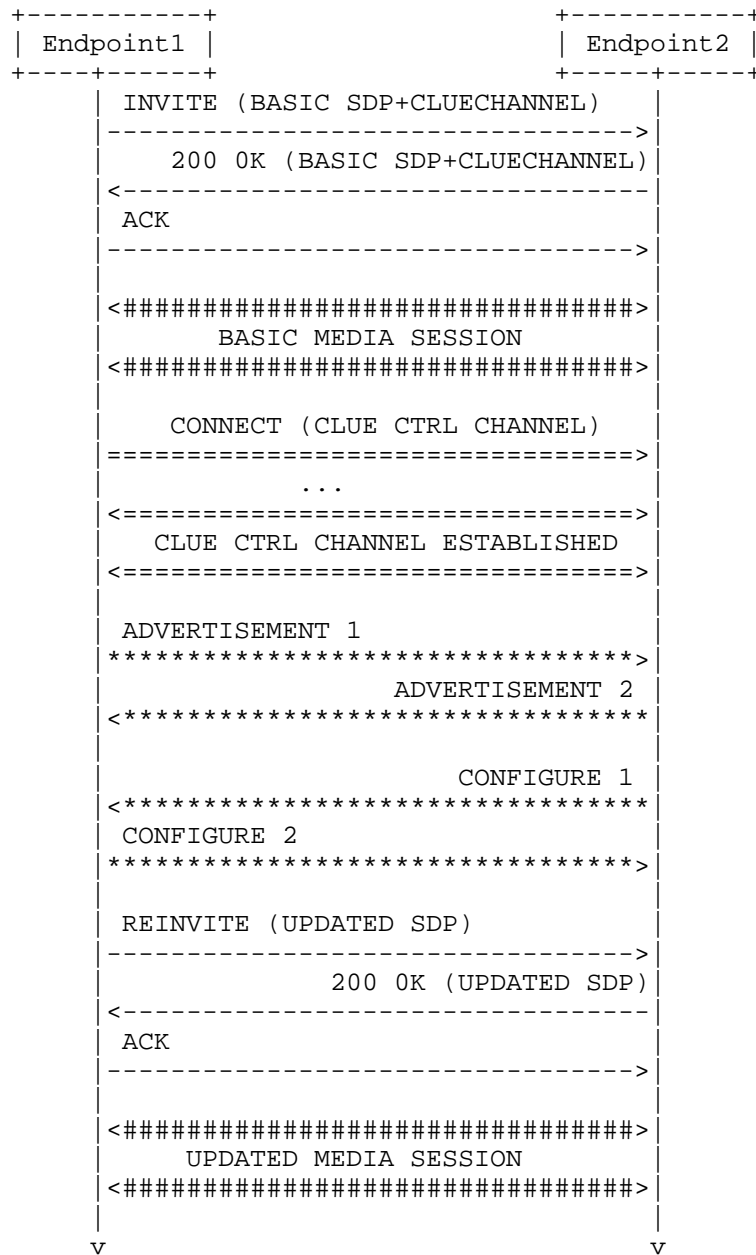


Figure 2: Basic Information Flow

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:

- o Each Capture Scene has a distinct coordinate system, unrelated to the coordinate systems of other scenes.

- o 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.
- o Coordinates can be either in real, physical units (millimeters), have an unknown scale or have no physical scale. Systems which know their physical dimensions (for example professionally installed Telepresence room systems) MUST 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.
- o 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:

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

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

Some key points about Media Captures:

- . A Media Capture is of a single media type (e.g. audio or video)
- . A Media Capture is defined in a Capture Scene and is given an Advertisement unique identity. The identity may be referenced outside the Capture Scene that defines it through a Multiple Content Capture (MCC)
- . A Media Capture 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.

7.1.1.1. 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.

7.1.1.1.1. Point of Capture

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

7.1.1.1.2. Point on Line of Capture

The Point on Line of Capture attribute is a field with a single Cartesian (X, Y, Z) point value which describes a position in space of a second point on the axis of the capturing device, 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 certainty that the capture contains that person type. The capture may contain other person types but the Advertiser has not been able to determine that this is the case.

The types of Captured people include:

- . 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.

7.2.1.1. 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 \leq to the number of Captures in the MCC this is a switched and composed case. The Capture Encoding may contain media switched between any number (up to the MaxCaptures) of composed sources.
- . When MaxCaptures is set to $=$ to the number of Captures in the MCC this is a composed case. All the sources are composed into a single Capture Encoding.

If this attribute is not set then as default it is assumed that all source 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 Configuremessage) 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.

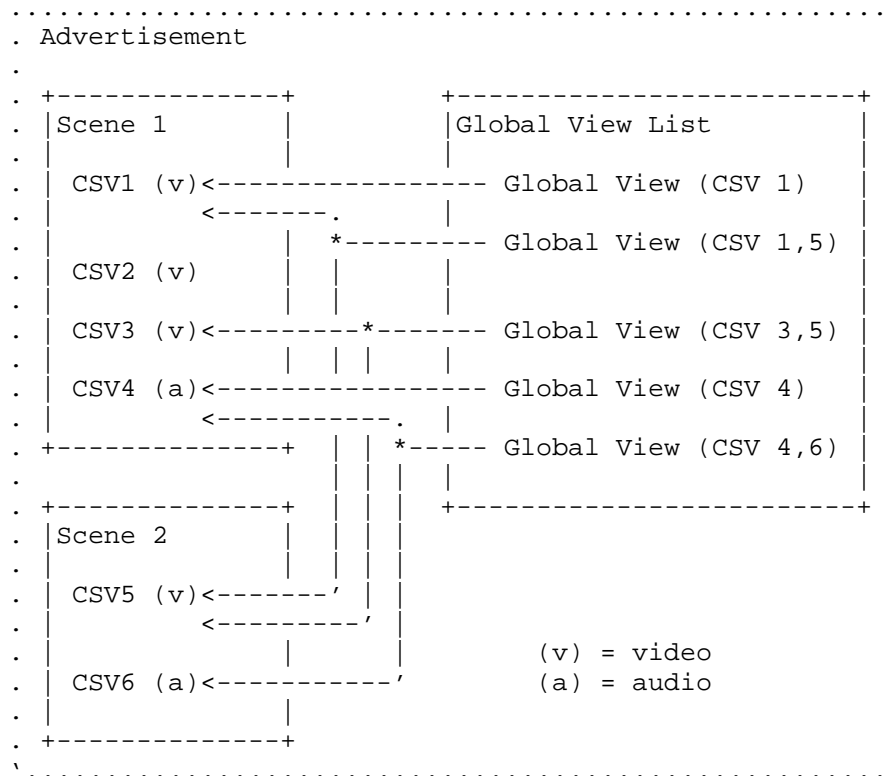


Figure 3: Global View List Structure

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.

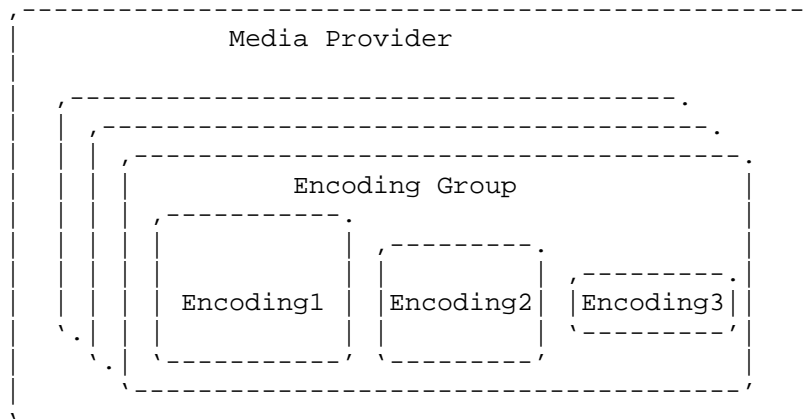


Figure 4: Encoding Group Structure

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:

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

10.2. Physical simultaneity restrictions

Often there are physical simultaneity constraints of the Provider that affect the Provider's ability to simultaneously send all of the captures the Consumer would wish to receive. For instance, 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:

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

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

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

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

Video Captures:

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

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

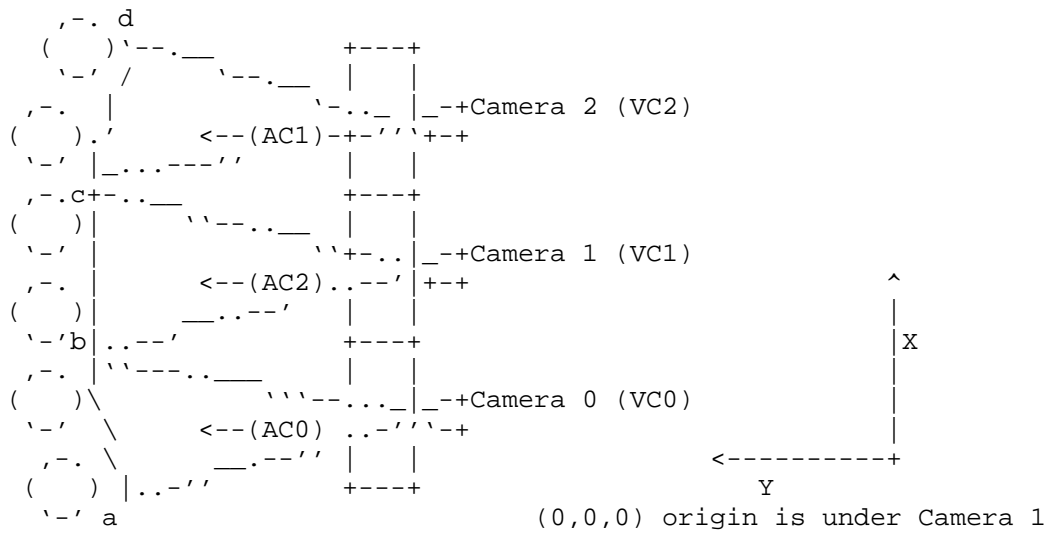


Figure 5: Room Layout Top View

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.

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

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

The physical simultaneity information is:

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

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

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

shows up in the encoding groups, not in the simultaneous transmission sets.

In this example there are no restrictions on which Audio Captures can be sent simultaneously.

Encoding Groups:

This example has three encoding groups associated with the video captures. Each group can have 3 encodings, but with each potential encoding having a progressively lower specification. In this example, 1080p60 transmission is possible (as ENC0 has a maxPps value compatible with that). Significantly, as up to 3 encodings are available per group, it is possible to transmit some video Captures simultaneously that are not in the same 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, maxWidht=1920, maxHeight=1088, maxFrameRate=60,
    maxPps=124416000, maxBandwidth=4000000
  encodeID=ENC1, maxWidht=1280, maxHeight=720, maxFrameRate=30,
    maxPps=27648000, maxBandwidth=4000000
  encodeID=ENC2, maxWidht=960, maxHeight=544, maxFrameRate=30,
    maxPps=15552000, maxBandwidth=4000000
encodeGroupID=EG1, maxGroupBandwidth=6000000
  encodeID=ENC3, maxWidht=1920, maxHeight=1088, maxFrameRate=60,
    maxPps=124416000, maxBandwidth=4000000
  encodeID=ENC4, maxWidht=1280, maxHeight=720, maxFrameRate=30,
    maxPps=27648000, maxBandwidth=4000000
  encodeID=ENC5, maxWidht=960, maxHeight=544, maxFrameRate=30,
    maxPps=15552000, maxBandwidth=4000000
encodeGroupID=EG2, maxGroupBandwidth=6000000
  encodeID=ENC6, maxWidht=1920, maxHeight=1088, maxFrameRate=60,
    maxPps=124416000, maxBandwidth=4000000
  encodeID=ENC7, maxWidht=1280, maxHeight=720, maxFrameRate=30,
    maxPps=27648000, maxBandwidth=4000000
  encodeID=ENC8, maxWidht=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).

12.1.1.2. 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 MCC1, MCC2 MCC3, MCC4, MCC5 MCC6, MCC7, MCC8, MCC9 AC0 CSV(MCC0) CSV(MCC1,MCC2) CSV(MCC3,MCC4,MCC5) CSV(MCC6,MCC7, MCC8,MCC9) CSV(AC0)	for a single screen consumer for a two screen consumer for a three screen consumer for a four screen consumer AC representing all participants

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	Description=Audience
CSV(VC1)	EncodeGroupID=1

Table 10: Advertisement received from Endpoint A

Capture Scene #1	Description=ChinaConfRoom
VC1	Description=Speaker EncodeGroupID=1
VC2	Description=Audience EncodeGroupID=1
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	Description=Speaker
VC3	EncodingGroup=1
CSV(VC2, VC3)	Description=Audience
Capture Scene #3	Description=USAConfRoom
VC4 CSV(VC4)	Description=Audience
Capture Scene #4	
MCC1(VC1,VC3,VC4)	Policy=RoundRobin:1
	MaxCaptures=1
	EncodingGroup=1
MCC2(VC2)	AllowSubset=True
	MaxCaptures=1
CSV2(MCC1,MCC2)	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.

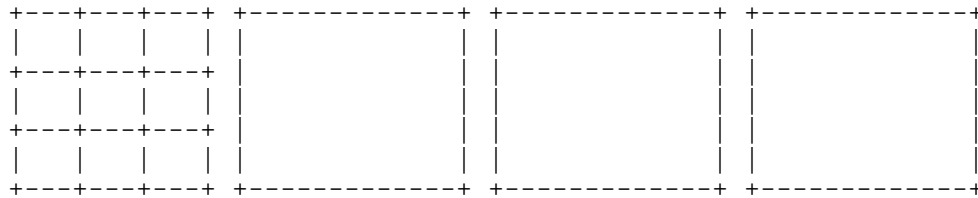


Figure 8: Endpoint A - 4 Screen Display

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.

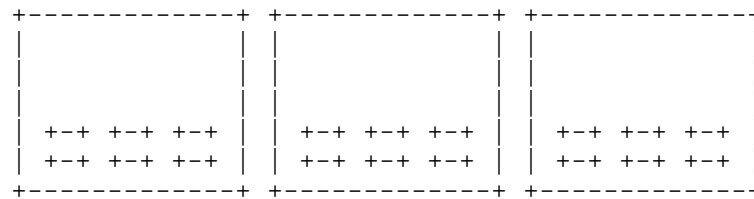


Figure 9: Endpoint B - 3 Screen Display with PiPs

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 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)	MaxCaptures=9 EncodingGroup=1
CSV(MCC13)	

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

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 19, 2014

P. Kyzivat
L. Xiao
C. Groves
Huawei
R. Hansen
Cisco Systems
September 15, 2013

CLUE Signaling
draft-kyzivat-clue-signaling-05

Abstract

This document specifies how signaling is conducted in the course of CLUE sessions. This includes how SIP/SDP signaling is applied to CLUE sessions as well as defining a CLUE-specific signaling protocol that complements SIP/SDP and supports negotiation of CLUE application level data.

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 March 19, 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. CLUE-Specific Signaling Protocol	4
3.1. CLUE Messages	4
3.1.1. Request Messages	4
3.1.2. Response Messages	5
3.1.3. CLUE Requests	5
3.1.3.1. ADVERTISEMENT Message	5
3.1.3.2. CONFIGURE Message	5
3.1.3.3. SUPPORTED Message	5
3.1.3.4. REQUIRED Message	5
3.1.4. CLUE Response Reasons	6
3.2. Message Sequencing	7
3.2.1. Pairing of Requests and Responses	8
3.2.2. Version Negotiation Happens First	8
3.2.3. Provider and Consumer Roles	8
3.2.4. Independent Sequencing of Provider and Consumer Roles	8
3.2.5. Signaling Changes in Provider State	8
3.2.6. Signaling Changes in Consumer State	9
3.2.7. Dangling Text [[TODO: FIX]]	9
3.3. Protocol Versioning and Options	10
3.3.1. Versioning Objectives	10
3.3.2. Versioning Overview	11
3.3.3. Version Negotiation	13
3.3.4. Option Negotiation	14
3.3.5. Option Elements	14
3.3.5.1. <mediaProvider>	14
3.3.6. Version & option negotiation errors	15
3.3.7. Definition and Use of Version Numbers	16
3.3.8. Version & Option Negotiation Examples	17
3.3.8.1. Successful Negotiation - Multi-version	17
3.3.8.2. Successful Negotiation - Consumer-Only Endpoint	18
3.3.8.3. Successful Negotiation - Provider-Only Endpoint	19
3.3.8.4. Version Incompatibility	20
3.3.8.5. Option Incompatibility	21
3.3.8.6. Syntax Error	22
3.4. Message Syntax	22
3.5. Message Transport	25
3.5.1. CLUE Channel Lifetime	26
3.5.2. Channel Error Handling	26
3.6. Message Framing	27
4. CLUE use of SDP O/A	27

4.1.	Establishing the CLUE channel	27
4.2.	Representing CLUE Encodings in SDP	27
4.3.	Representing CLUE Encoding Groups in SDP	28
4.4.	Signaling CLUE control of "m" lines	28
4.5.	Ensuring interoperability with non-CLUE devices	29
5.	Interaction of CLUE and SDP negotiations	29
5.1.	Independence of SDP and CLUE negotiation	30
5.2.	Recommendations for operating with non-atomic operations	30
5.3.	Constraints on sending media	31
6.	Example: A call between two CLUE-capable endpoints	31
7.	Example: A call between a CLUE and non-CLUE-capable endpoint	38
8.	CLUE requirements on SDP O/A	40
9.	SIP Signaling	40
10.	Interoperation with Legacy SIP Devices	40
11.	CLUE over RTCWEB	40
12.	Open Issues	41
13.	What else?	41
14.	Acknowledgements	41
15.	IANA Considerations	41
16.	Security Considerations	41
17.	Change History	41
18.	References	43
18.1.	Normative References	43
18.2.	Informative References	43
	Authors' Addresses	44

1. Introduction

This document specifies how signaling is conducted in the course of CLUE sessions. This includes how SIP/SDP signaling is applied to CLUE sessions as well as defining a CLUE-specific signaling protocol that complements SIP/SDP and supports negotiation of CLUE application level data.

[Yes, this is a dup of the abstract for now. Eventually it should say more.]

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 Channel: A reliable, bidirectional, transport mechanism used to convey CLUE messages. A CLUE channel consists of one SCTP stream in each direction over a DTLS/SCTP session.

3. CLUE-Specific Signaling Protocol

The CLUE Framework [I-D.ietf-clue-framework] mentions a CLUE-specific protocol for the exchange of ADVERTISEMENT and CONFIGURE messages, but gives little detail. The Data Model [I-D.presta-clue-data-model-schema] specifies a model and XML representation for CLUE-related data, but doesn't currently specify exactly what data belongs in each message, or how messages are sequenced. This document provides the detail missing from those documents.

3.1. CLUE Messages

CLUE messages have the following characteristics:

- o They are encoded in XML, following the schema in section Section 3.4].
- o There are two kinds of messages - requests and responses. (This is similar to SIP.)
- o Every request message expects exactly one response message.
- o Every request message carries a sequence number that identifies it.
- o Each end of the connection assigns sequential sequence numbers to the requests it sends.
- o Every response message carries the sequence number of the message to which it responds.
- o Responses are to be sent promptly upon the receipt of a request. (Needs more detail.)
- o Responses also carry info describing the error.

3.1.1. Request Messages

A request message specifies an action the sender is requesting the recipient to perform. There are a number of request types, each of which describes a different action. Each carries parameters qualifying the action.

3.1.2. Response Messages

The recipient of a request message must send a response message that acknowledges that the request has been received. The response indicates if the request was processed successfully, and if not, then the reason for failure. A special reason (OK) indicates that the request was processed successfully.

3.1.3. CLUE Requests

3.1.3.1. ADVERTISEMENT Message

This message contains XML representations of captures, capture scenes, encoding groups, and simultaneous sets using the types defined for those in the Data Model [I-D.presta-clue-data-model-schema].

The XML definition for this is element <advertisement> in section Section 3.4.

[[Currently this does not contain any representation of encodings. It assumes those will be defined in SDP.]]

3.1.3.2. CONFIGURE Message

This message optionally contains an XML representations of captureEncodings using the type defined in the Data Model [I-D.presta-clue-data-model-schema]. A configure message with no captureEncodings indicates that no captures are requested.

[[It currently also contains a reference to the request number of the advertisement it is based upon. Whether this should be present, or if it should implicitly reference the most recently acknowledged advertisement is TBD.]]

The XML definition for this is element <configure> in section Section 3.4

3.1.3.3. SUPPORTED Message

The Supported message describes the CLUE versions, and options, supported by the sender. The details of how use this message are presented in section Section 3.2.2.

3.1.3.4. REQUIRED Message

The Required message describes the CLUE version, and options, that have been negotiated by the sender and the receiver. The details of how use this message are presented in section Section 3.2.2.

3.1.4. CLUE Response Reasons

The following reasons are defined:

OK: The message was successfully processed.

Syntax Error: Message has failed due to a syntax error detected at the message level. The message does not conform to the schema. Used when the message cannot be parsed.

Sequencing Error: Sequence number has already been used, or is greater than the expected number. (Details of possible errors depend upon the specific sequence numbering mechanism.)

Version incompatibility: There is no common value between the major version numbers supported by the two endpoints of the CLUE channel.

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 is 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: Unsupported option

An option element type received in a Required message did not appear in the corresponding Supported element.

(This code is never received in response to a Supported message.)

Unknown capture identity: The received Configure message contains an unknown capture identity not previously declared by an Advertisement. The message is ignored.

Invalid identity: The received Configure message contains an invalid capture identity. For example a duplicated Capture scene identity or some other semantically incorrect usage. The message has ignored.

Invalid value: The received message contains an invalid parameter value. The value is not according to the specification for the containing element.

Missing element: The received message is missing an element. Certain parameters require multiple values, e.g. Point of capture requires X,Y,Z co-ordinates if one or more elements are missing this error code is used.

Conflicting parameters or values: The received message contains multiple values that may not be used together.

Invalid capture area: The received message defines a capture area that cannot be rendered in a sensible manner. For example the capture area does not define a quadrilateral region.

Invalid point of line of capture: The indicated co-ordinate for the point on line of capture is invalid. For example: does not lie between the point of capture and the area of capture or it is the same as the point of capture.

Invalid capture scene entry: The message contains an invalid capture scene entry. For example the capture scene entry contains more than one media type.

Invalid Simultaneous Set: The simultaneous set contained in the message is invalid. For example the simultaneous set refers to an undefined capture set or does not match the specified capture scene entries.

Invalid Configuration: The Configure message requests a configuration that the provider cannot support.

Invalid Advertisement reference: The Configure message refers to an invalid Advertisement. The message refers-to/depends-upon out-of-date ADVERTISEMENT message or provides an invalid reference.

3.2. Message Sequencing

[[NOTE: Should have some state machines formalizing this sequencing.]]

3.2.1. Pairing of Requests and Responses

Each endpoint of a CLUE channel sends both requests and responses. For each request sent, a prompt response is required. Each response identifies the request to which it responds. After sending a request, one or more requests may be received before a response is received.

3.2.2. Version Negotiation Happens First

Upon establishment of the CLUE channel, version and option negotiation, using Supported and Required requests, MUST complete before any other requests are sent on the channel. Refer to section Section 3.3 for details of this process.

3.2.3. Provider and Consumer Roles

Each CLUE endpoint has two roles that it potential performs: provider and consumer. In common cases an endpoint can perform both roles. Which roles are actually performed is determined during option negotiation.

3.2.4. Independent Sequencing of Provider and Consumer Roles

In the provider role, an endpoint sends Advertisement messages and receives Configure messages.

In the consumer role, an endpoint receives Advertisement messages and sends Configure messages.

The messages (requests and responses) used by each role are exchanged over the same CLUE channel, and may be interleaved in an arbitrary manner, constrained only by the sequencing rules for each role.

3.2.5. Signaling Changes in Provider State

Once a CLUE session has been established, ADVERTISEMENTS and CONFIGURES exchanged, and media is flowing, a provider may experience a change in state that has an effect on what it wishes or is able to provide. In this case it may need to alter what it is sending and/or send a new ADVERTISEMENT. In some cases it will be necessary to alter what is being sent without first sending a new ADVERTISEMENT and waiting for a CONFIGURE conforming to it.

The following is a non-exhaustive list of situations and recommended actions:

- o An advertised capture, that is not currently configured, is no longer available.

To recover from this: Send a new ADVERTISEMENT that omits this capture.

- o An advertised capture, that has been configured, is no longer available.

To recover from this: (1) stop transmitting the configured encoding of this capture. (2) Send a new ADVERTISEMENT that omits this capture.

- o The provider loses some resource and must reduce the frame rate, frame size, or resolution of a capture encoding.

If the reduced values still fall within the advertised values for the capture then the change may be made without any further signaling.

If the change must be outside the range of what was advertised, then the provider must cease transmitting the capture encoding. It then must send a new ADVERTISEMENT reflecting what it is now capable of delivering.

- o New or changed scenes or scene geometry. For instance, the addition of a new scene containing presentation captures. Also, an MCU may make significant changes in what it advertises as new endpoints join a conference.
- o [Add more]

3.2.6. Signaling Changes in Consumer State

If the Consumer for some reason loses the CLUE state information how does it ask for an Advertisement from the provider? There could be multiple possibilities. A error code approach? However error codes would typically be associated with a NACK so it may not be good for a Config message. Maybe send a message which means "send me a complete update". An alternative may be to release the connection or just do new signaling to establish a new CLUE session.

3.2.7. Dangling Text [[TODO: FIX]]

There is a very basic introduction to this topic in section 4 (Overview) of the CLUE Framework [I-D.ietf-clue-framework]. After removing extraneous material it would look like:

```

+-----+                               +-----+
| Endpoint1 |                           | Endpoint2 |
+-----+                               +-----+

|
|  ADVERTISEMENT 1
|  *****>
|                ADVERTISEMENT 2
|  <*****
|
|                CONFIGURE 1
|  <*****
|  CONFIGURE 2
|  *****>
|

```

But we need much more than this, to show multiple CONFIGUREs per ADVERTISEMENT, interleaving of ADVERTISEMENTS and CONFIGUREs in both directions, etc.

Message sequencing needs to be described at two levels:

- o Basic sequencing of the CLUE messages themselves, without regard for the SIP/SDP signaling that may be going on at the same time. This is useful to cover the basic concepts. That should be covered in this section. It provides context for understanding the more detailed treatment later.

This could include some simple state machines.

- o In reality there is a complex dependency between CLUE signaling and SDP Offer/Answer exchanges carried in SIP signaling. So there is a need to describe the valid ways in which these two forms of signaling interact. That is covered in Section 5.

3.3. Protocol Versioning and Options

3.3.1. Versioning Objectives

The CLUE versioning mechanism addresses the following needs:

- o Coverage:
 - * Versioning of basic behavior and options,

- * CLUE message exchange,
 - * CLUE message exchange,
 - * coordinated use of SIP and SDP,
 - * required media behavior.
- o Remain fixed for the duration of the CLUE channel
 - o Be extensible for configuration of new options.
 - o Be sufficient (with extensions) for all envisioned future versions.

3.3.2. Versioning Overview

An initial message exchange on the CLUE channel handles the negotiation of version and options.

- o Dedicated message types are used for this negotiation.
- o The negotiation is repeated if the CLUE channel is reestablished.

The version usage is similar in philosophy to XMPP:

- o See [RFC6120] section 4.7.5.
- o A version has major and minor components. (Each a non-negative integer.)
- o Major version changes denote non-interoperable changes.
- o Minor version changes denote schema changes that are backward compatible by ignoring unknown XML elements, or other backward compatible changes.
- o 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:

```

| Supported      Supported |
|-----\ /-----|

```

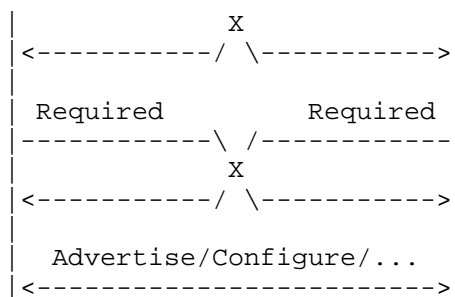


Figure 1: Basic Option Negotiation (simplified)

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)

3.3.3. 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.

- o If there is no major version in common between the two ends, negotiation fails.
- o The <version> elements from the two ends that have the largest matching major version are selected.
- o 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.
- o 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.]]

3.3.4. 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, an 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.

3.3.5. 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>.

3.3.5.1. <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.

3.3.6. Version & option negotiation errors

The following are errors that may be detected and reported during version negotiation:

- o Version incompatibility

There is no common value between the major version numbers sent in a Supported message and those in the received Supported message.

- o 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 is 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.

- o 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.

3.3.7. 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:

- o the schema defined in Section 3.4 of this document;
- o the schema defined in [clue-data-model];
- o the protocol used to exchange CLUE messages;
- o the protocol defined herein that defines valid sequence of CLUE messages;
- o 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:

- o All messages valid according to the schema of Vx are also valid according to the schemas of Vy

- o 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:

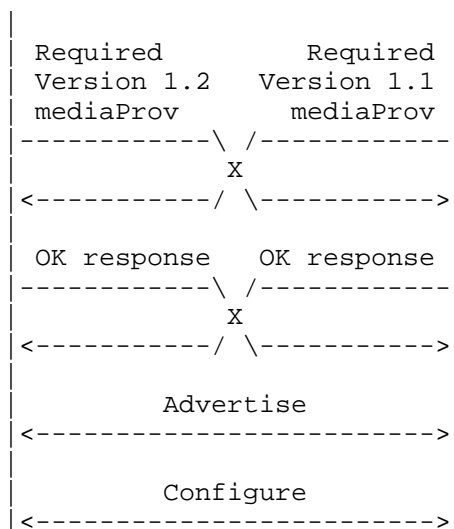
- o If the revision and the document being revised are mutually backward compatible (they are functionally equivalent), then the CLUE version MUST remain unchanged.
- o Else if the revision is backward compatible with the document being revised, then the CLUE major version MUST remain unchanged, and 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.

3.3.8. Version & Option Negotiation Examples

3.3.8.1. Successful Negotiation - Multi-version

Supported	Supported
Version 2.0	
Version 1.2	Version 1.1
mediaProv	mediaProv
-----\	/-----
	X
<-----/	\----->
OK response	OK response
-----\	/-----
	X
<-----/	\----->



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 wish 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.

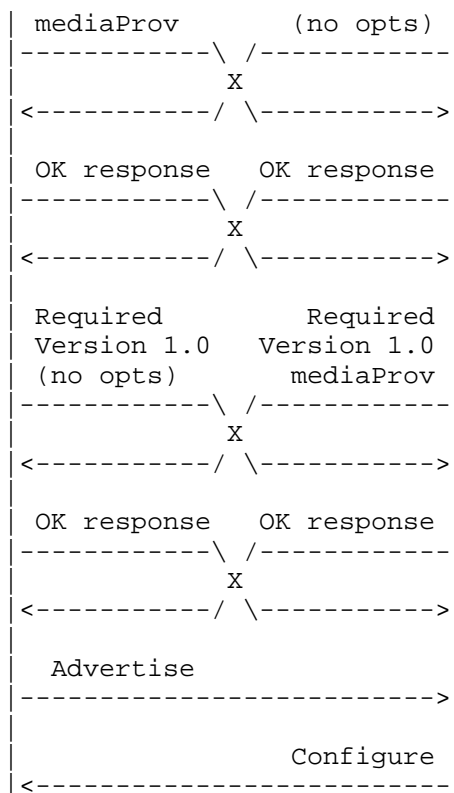
The element on the right, upon receiving the Supported message, selects version 1.1 because it is the highest version in common to the two sides. It sends a Required message containing version 1.1 because that is the highest version in common. It also includes the mediaProvider option element, because 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.

3.3.8.2. Successful Negotiation - Consumer-Only Endpoint



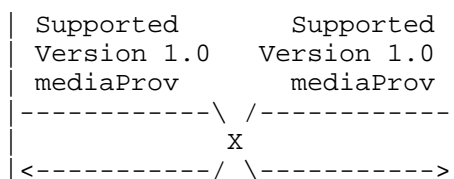


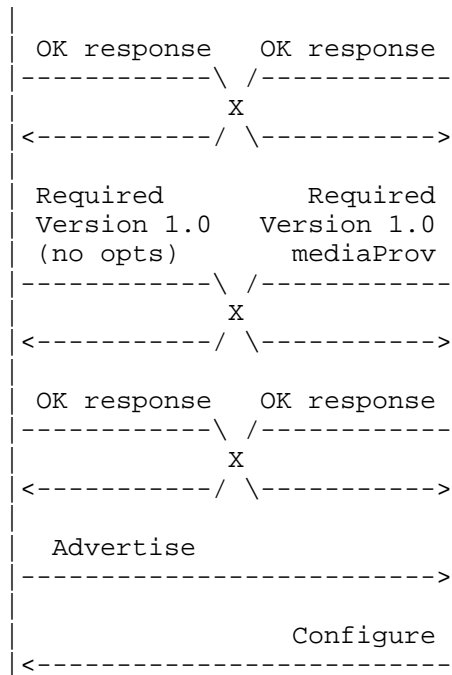
The endpoint on the right consumes media, but doesn't provide any so it doesn't include the mediaProvider option element in the Supported message it sends.

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.

3.3.8.3. Successful Negotiation - Provider-Only Endpoint

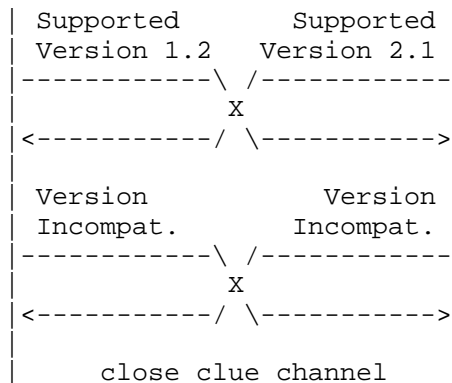




The endpoint on the left provides media but does not consume any so it includes the mediaProvider option element in the Supported message it sends, but doesn't include the mediaProvider option element in the Required message it sends.

Advertisement messages will only go from left to right, and Configure messages will only go from right to left.

3.3.8.4. Version Incompatibility



```

|<----->|
| legacy mode or BYE |
|<----->|

```

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.

3.3.8.5. Option Incompatibility

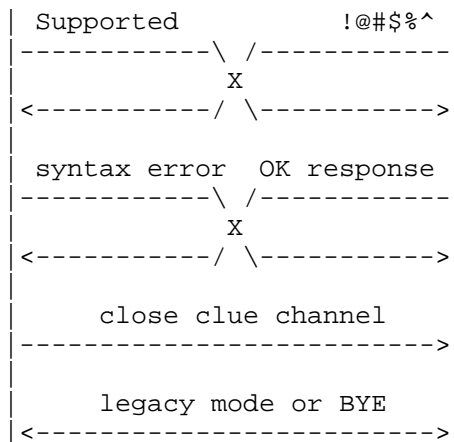
```

| Supported      Supported |
| Version 1.0    Version 1.0 |
| mediaProv      mediaProv |
|-----\ /-----|
|                X          |
|<-----/ \----->|
| Required      Required |
| (no opts)     (no opts) |
|-----\ /-----|
|                X          |
|<-----/ \----->|
| Option        Option |
| Incompat.     Incompat. |
|-----\ /-----|
|                X          |
|<-----/ \----->|
| close clue channel |
|<----->|
| legacy mode or BYE |
|<----->|

```

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.

3.3.8.6. Syntax Error



3.4. Message Syntax

[[The following is a first cut at a schema for the actual messages in the clue protocol. It uses <encodingGroups> from the data model but not <encodings>. Rather, 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>. If we stick with this the data model should be adjusted to agree, but until then it should "work". The SDP encoding of the identifier is proposed to be 'a=label:ID', though 'a=mid:ID' is another candidate.]]

For now there only <advertisement> and <configure> are defined. More messages will be needed for acknowledgment.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  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-04-wip.xsd"/>
```

```
<!-- ELEMENT DEFINITIONS -->
<xs:element name="response" type="responseMessageType"/>
<xs:element name="advertisement" type="advertisementMessageType"/>
<xs:element name="configure" type="configureMessageType"/>
<xs:element name="supported" type="supportedMessageType"/>
<xs:element name="required" type="requiredMessageType"/>

<!-- 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 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>

<!-- CLUE ADVERTISEMENT MESSAGE TYPE -->
<xs:complexType name="advertisementMessageType">
  <xs:complexContent>
    <xs:extension base="clueRequestMessageType">
      <xs:sequence>
```

```
<!-- mandatory fields -->
<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="advertisementNumber" type="xs:integer"/>
        <!-- advertisementNumber is requestNumber
              of the advertisement-->
        <!-- 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>

<!-- CLUE SUPPORTED MESSAGE TYPE -->
<xs:complexType name="supportedMessageType">
  <xs:complexContent>
    <xs:extension base="clueRequestMessageType">
      <xs:sequence>
        <!-- mandatory fields -->
        <xs:element name="version"
                    type="versionType"
                    maxOccurs="unbounded"/>
        <!-- optional fields -->
```

```
<xs:element name="Options"
            type="OptionsType" minOccurs="0"/>
<xs:any namespace="##other"
        processContents="lax" minOccurs="0"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- CLUE REQUIRED MESSAGE TYPE -->
<xs:complexType name="requiredMessageType">
  <xs:complexContent>
    <xs:extension base="clueRequestMessageType">
      <xs:sequence>
        <!-- mandatory fields -->
        <xs:element name="version"
                    type="versionType"
                    maxOccurs="1"/>
        <!-- optional fields -->
        <xs:element name="Options"
                    type="OptionsType" minOccurs="0"/>
        <xs:any namespace="##other"
            processContents="lax" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- OPTIONS TYPE -->
<xs:complexType name="optionsType">
  <!-- each element represents one option -->
  <xs:any processContents="lax" minOccurs="0"/>
</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>
```

3.5. 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.)

3.5.1. 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 may 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 TBD.

3.5.2. 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:

- o Unable to establish the SCTP association after signaling it in SDP.
- o CLUE channel setup rejected by peer.
- o Error reported by transport while writing message to CLUE channel.
- o Error reported by transport while reading message from CLUE channel.
- o Timeout - overdue acknowledgement of a CLUE message.
(Requirements for how soon a message must be responded to are TBD.)

- o 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.

3.6. Message Framing

Message framing is provided by the SCTP transport protocol. Each CLUE message is carried in one SCTP message.

4. CLUE use of SDP O/A

4.1. Establishing the CLUE channel

The CLUE channel is usually offered in the first SIP O/A exchange between two parties in an intended CLUE session. The offer of the CLUE channel is the indicator that this SIP session is proposing to establish a CLUE session.

(However it is also acceptable to start with a non-CLUE SIP session and upgrade it to a CLUE session later.)

The mechanism for negotiating a DTLS/SCTP connection is specified in [I-D.ietf-mmusic-sctp-sdp]. We need to specify how to select the specific pair of SCTP streams that comprise the CLUE channel.

The presence of an active m-line for the CLUE channel in an SDP offer is an indication that the offer that the sender is CLUE-capable and hence can understand CLUE-specific syntax.

4.2. Representing CLUE Encodings in SDP

Many CLUE constructs have no good analog in SDP. Entities such as 'captures', which describe spatial and other properties of a capture source such as a camera, are not tied directly to RTP streams, do not have negotiated properties and would prove a significant challenge to represent in SDP syntax (while also greatly increasing the size of the SDP).

However, two entities defined in the CLUE Framework [I-D.ietf-clue-framework] are a much closer fit for SDP: Encodings and Encoding Groups. Both describe RTP media properties and limitations, though unlike most SDP usage they describe the sender's capabilities, not the receiver's. Representing encodings in CLUE

splits media limitations across two protocols, and risks duplicated and potentially contradictory information being sent in CLUE and SDP. As such we are exploring representing this information in SDP, with the decision to convey them in the CLUE messages only to be made if the SDP approach proves impractical.

This draft presents an attempt to describe CLUE encodings in SDP. As a decision has not yet been reached on how multiplexed RTP streams are to be expressed in SDP, at this stage the draft does so without multiplexing, using existing SDP attributes, with a separate "m" line and hence port per unidirectional RTP stream. This is done with the understanding that when a decision is reached on new syntax for multiplexing RTP streams in SDP the CLUE SDP signaling will be modified to use it. Further, the framework document states that the multiplexing of streams by an implementation is optional, and in the case of a disaggregated system, with media streams going to different addresses, may not be possible.

With the current scheme of using existing syntax, an encoding is specified in SDP as a unicast "m" line, which MUST be marked as sendonly with the "a=sendonly" attribute or as inactive with the "a=inactive" attribute. The encoder capabilities of the stream are defined here using existing 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.

A receiver who wishes to receive a CLUE stream via this encoding requires a matching "a=recvonly" "m" line. 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. Representing CLUE Encoding Groups in SDP

As per the previous section, there would be advantages to conveying encoding group information in SDP. However, with current SDP syntax there is no way to express the encoding group limits defined in the Data Model [I-D.presta-clue-data-model-schema]. As such the current draft keeps encoding groups as part of the Advertisement message for the time being.

4.4. 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 channel. If a CLUE 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 negotiation has taken place as defined in this document. If no CLUE 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.

4.5. Ensuring interoperability with non-CLUE devices

A CLUE-capable device sending an initial SDP offer SHOULD include an "m" line for the CLUE channel, but SHOULD NOT include any other CLUE-controlled "m" lines. Once each side of the call is aware that the other side is CLUE-capable a new O/A exchange MAY be used to add CLUE-controlled "m" lines.

5. Interaction of CLUE 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.

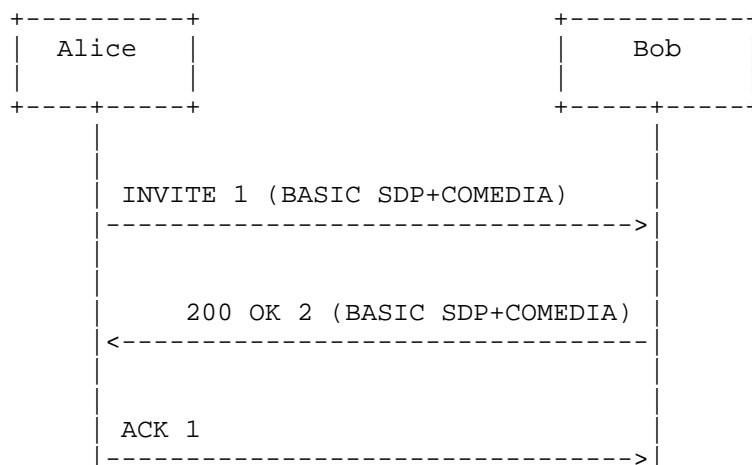
5.3. Constraints on sending media

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.



```

<##### MEDIA 1 #####>
  1 video A->B, 1 video B->A
<#####>

```

```

<=====>
  CLUE CTRL CHANNEL ESTABLISHED
<=====>

```

```

  ADVERTISEMENT 1
  *****>

```

```

                                ADVERTISEMENT 2
  <*****

```

```

  INVITE 2 (+3 sendonly)
  ----->

```

```

                                CONFIGURE 1
  <*****

```

```

                                200 OK 2 (+2 recvonly)
  <-----

```

```

  ACK 2
  ----->

```

```

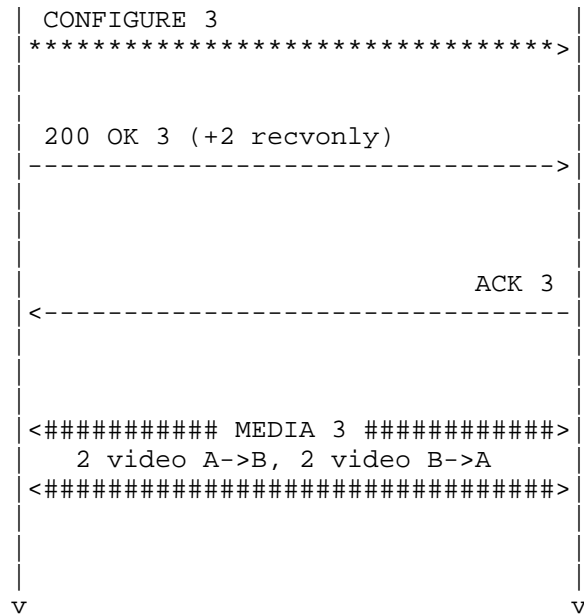
<##### MEDIA 2 #####>
  2 video A->B, 1 video B->A
<#####>

```

```

                                INVITE 3 (+2 sendonly)
  <-----

```



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-mps=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-mps=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".

Alice receives Bob's message CONFIGURE 1 but 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-mps=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-mps=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-mps=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 but 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):

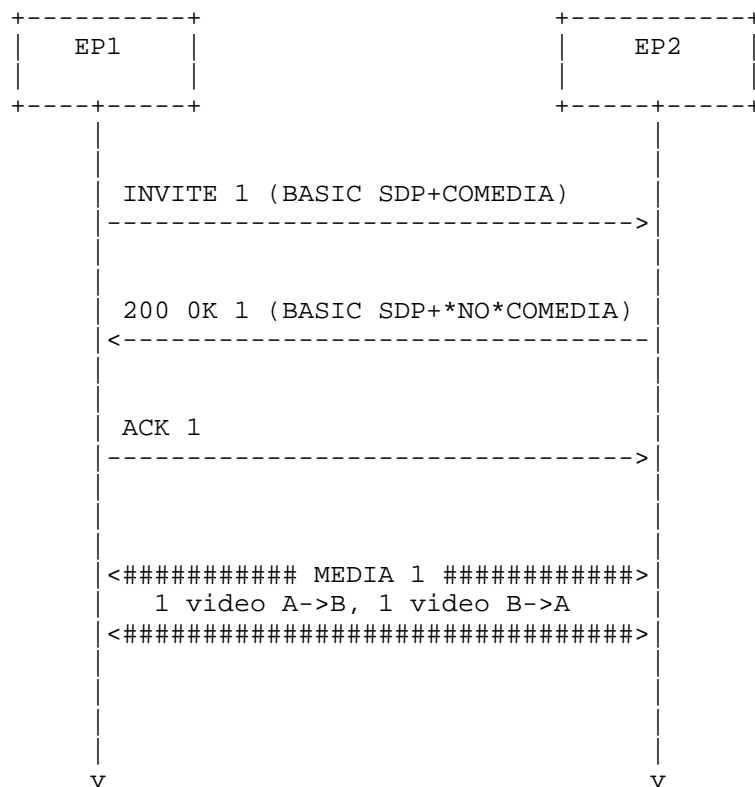
```
...
a=group:CLUE 3 4 5 7 8
...
m=video 6002 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mps=108000;max-fs=3600
a=sendrecv
a=mid:2
...
m=video 6004 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:4
a=label:enc1
m=video 6006 RTP/AVP 96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

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

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

7. Example: A call between a CLUE and non-CLUE-capable 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-mps=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. Interoperation with Legacy SIP Devices

This may just describe how the degenerate form of the general mechanisms work for legacy devices. Or it may describe special case handling that we mandate as part of CLUE. Or it may just discuss non-normative things for implementors should consider.

11. CLUE over RTCWEB

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

12. Open Issues

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

- o 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.

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

13. What else?

14. Acknowledgements

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

Christian Groves 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

-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-11 (work in progress), July 2013.
- [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.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-04 (work in progress), June 2013.
- [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

- [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.

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 group 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 tendency 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 separate port used for this purpose may impose costs that a device wishes to avoid, such as the need to open that port on firewalls and NATs, the need to collect ICE candidates [RFC5245], etc.

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

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

6.2. Usage of BUNDLE with CLUE

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

6.2.1. Generating the Initial Offer

BUNDLE mandates that the initial SDP offer MUST use a unique address for each m-line with a non-zero port. Because CLUE implementations generally 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.

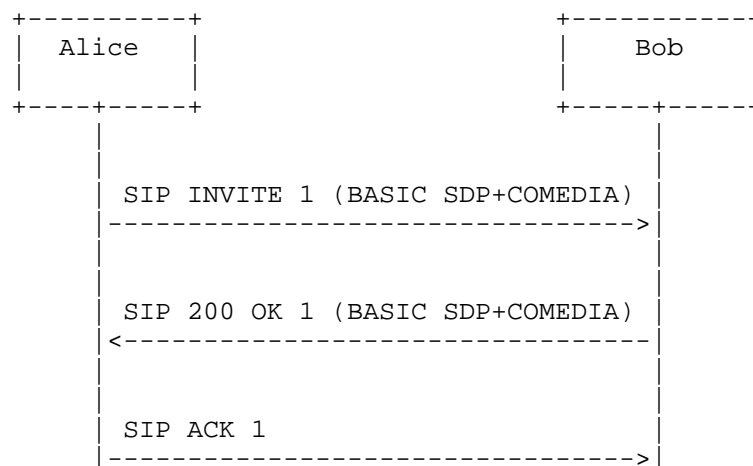
6.2.3. Multiplexing of the data channel and RTP media

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
<*****>

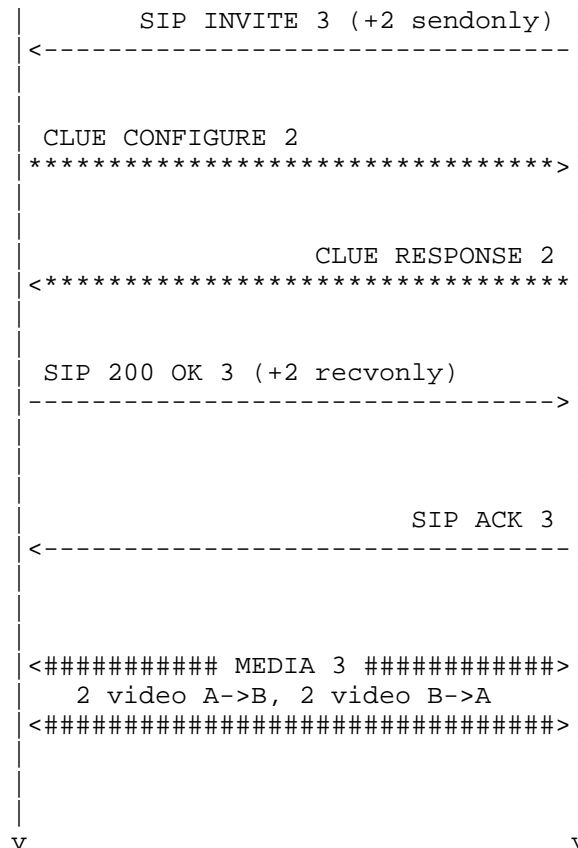
  CLUE RESPONSE 1
  *****>

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

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

Alice now sends her CLUE Advertisement (ADVERTISEMENT 1). She advertises three static captures representing her three cameras. She also includes switched captures suitable for two- and one-screen systems. All of these captures are in a single capture scene, with suitable capture scene entries to tell Bob that he should either subscribe to the three static captures, the two switched capture view or the one switched capture view. Alice has no simultaneity constraints, so includes all six captures in one simultaneous set. Finally, Alice includes an encoding group with three encoding IDs: "enc1", "enc2" and "enc3". These encoding ids aren't currently valid, but will match the next SDP offer she sends.

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

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

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

Alice now sends INVITE 2. She maintains the sendrecv audio, video

and CLUE m-lines, and she adds three new sendonly m-lines to 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 ADVERTISEMENT 1.

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

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

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

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

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

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

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

Bob receives Alice's message CONFIGURE 2 and sends RESPONSE 2 to ack its reception. 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
```

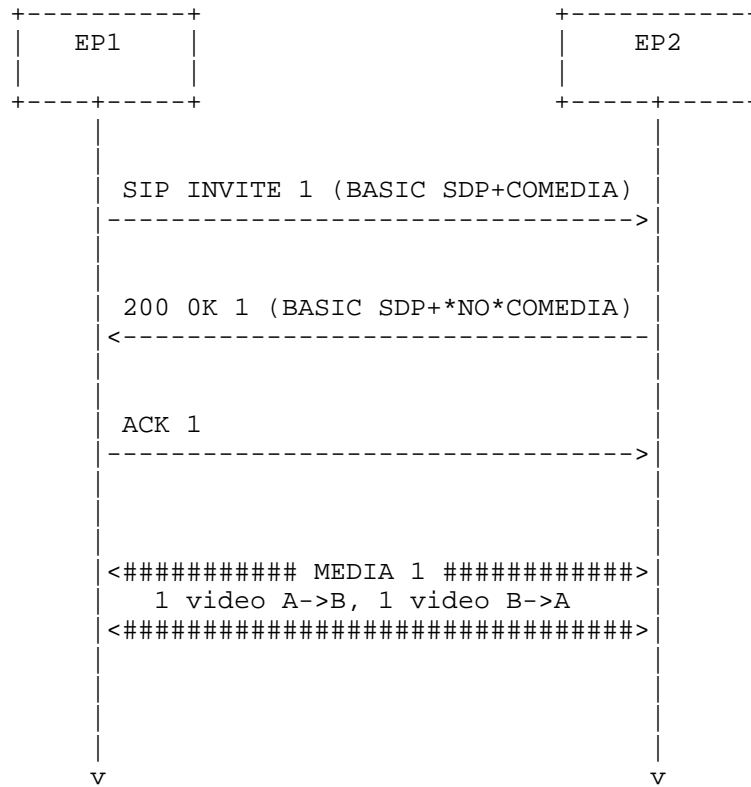
Finally, on receiving 200 OK 3 Bob is now able to send the two streams of video Alice requested - this is illustrated as MEDIA 3.

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.

- o 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.

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

13. What else?

14. Acknowledgements

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

Christian Groves 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 individual 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:

- o Coverage:
 - * Versioning of basic behavior and options,
 - * CLUE message exchange,
 - * CLUE message exchange,
 - * coordinated use of SIP and SDP,
 - * required media behavior.
- o Remain fixed for the duration of the CLUE channel
- o Be extensible for configuration of new options.
- o 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.

- o Dedicated message types are used for this negotiation.
- o The negotiation is repeated if the CLUE channel is reestablished.

The version usage is similar in philosophy to XMPP:

- o See [RFC6120] section 4.7.5.
- o A version has major and minor components. (Each a non-negative integer.)
- o Major version changes denote non-interoperable changes.
- o Minor version changes denote schema changes that are backward compatible by ignoring unknown XML elements, or other backward compatible changes.
- o 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:

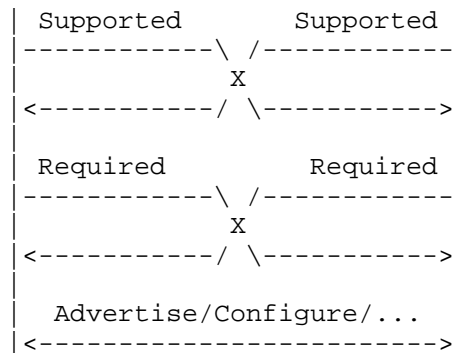


Figure 1: Basic Option Negotiation (simplified)

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.

- o If there is no major version in common between the two ends, negotiation fails.
- o The <version> elements from the two ends that have the largest matching major version are selected.
- o 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.
- o 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, an 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:

- o Version incompatibility

There is no common value between the major version numbers sent in a Supported message and those in the received Supported message.

- o 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 is 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.

- o 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:

- o the schema defined in [I-D.presta-clue-protocol];
- o the schema defined in [clue-data-model];
- o the protocol used to exchange CLUE messages;
- o the protocol defined herein that defines valid sequence of CLUE messages;
- o 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:

- o All messages valid according to the schema of Vx are also valid according to the schemas of Vy
- o 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.

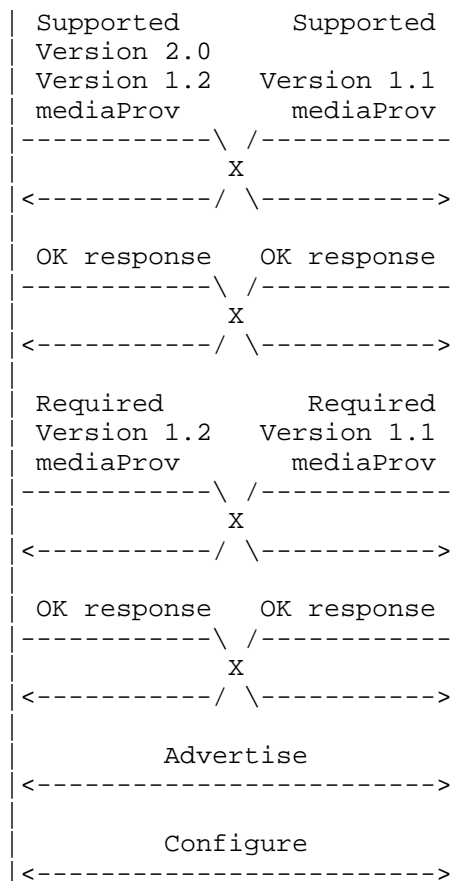
The CLUE version to be defined in a revision to this document MUST be determined as follows:

- 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.

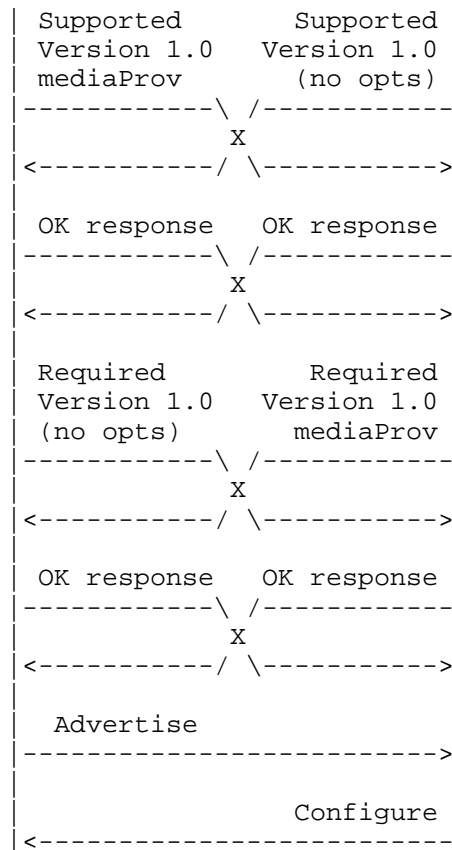
The element on the left, upon receiving the Supported message, determines that it is permitted to use version 1.2 or 1.1, and decides to use 1.2. It sends a Required message containing version 1.2 and also includes the mediaProvider option element, because it wants its peer to provide media.

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

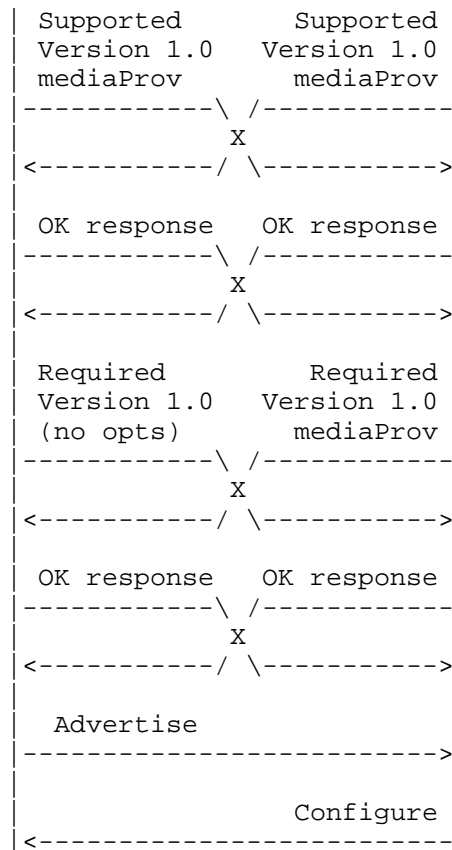


The endpoint on the right consumes media, but doesn't provide any so it doesn't include the mediaProvider option element in the Supported message it sends.

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.

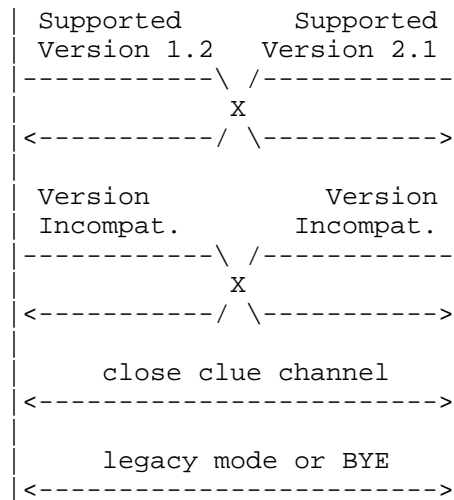
A.1.8.3. Successful Negotiation - Provider-Only Endpoint



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 doesn't include the mediaProvider option element in the Required message it sends.

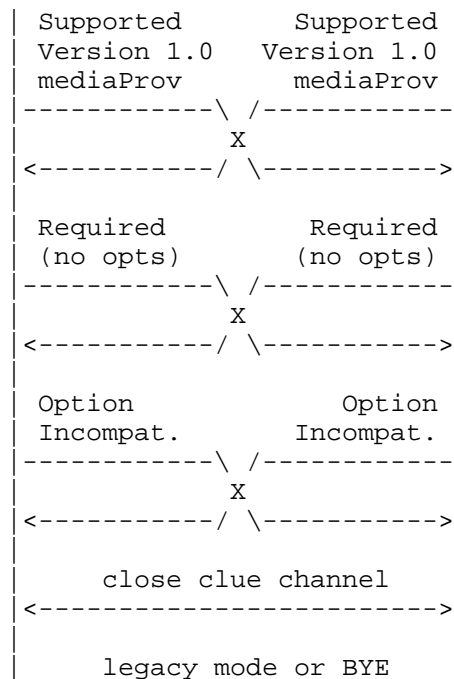
ADVERTISEMENT messages will only go from left to right, and CONFIGURE messages will only go from right to left.

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.

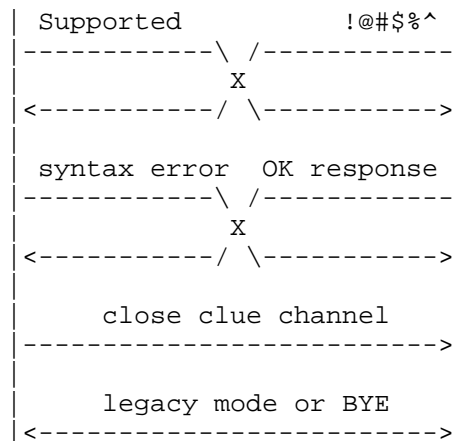
A.1.8.5. Option Incompatibility



|<----->|

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 may 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 TBD.

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:

- o Unable to establish the SCTP association after signaling it in SDP.
- o CLUE channel setup rejected by peer.
- o Error reported by transport while writing message to CLUE channel.
- o Error reported by transport while reading message from CLUE channel.
- o Timeout - overdue acknowledgement of a CLUE message.
(Requirements for how soon a message must be responded to are TBD.)
- o 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
Internet-Draft
Intended status: Informational
Expires: April 11, 2014

R. Presta
S P. Romano
University of Napoli
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 [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 conveying 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].

- o ADVERTISEMENT (ADV)
- o CONFIGURE (CONF)
- o RESPONSE
- o RE-ADV
- o OPTIONS

3.1. ADVERTISEMENT

FROM	MP
TO	MC
TYPE	Notification
DESCRIPTION	<p>This message is used by the MP to advertise the available media captures and related information to the MC.</p> <p>The ADV contains elements compliant with the CLUE data model and other information like the CLUE protocol version and a sequence number.</p>
USAGE	<p>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.</p> <p>The ADV message is also sent back to the MC when the MP receives a RE-ADV request.</p>

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.

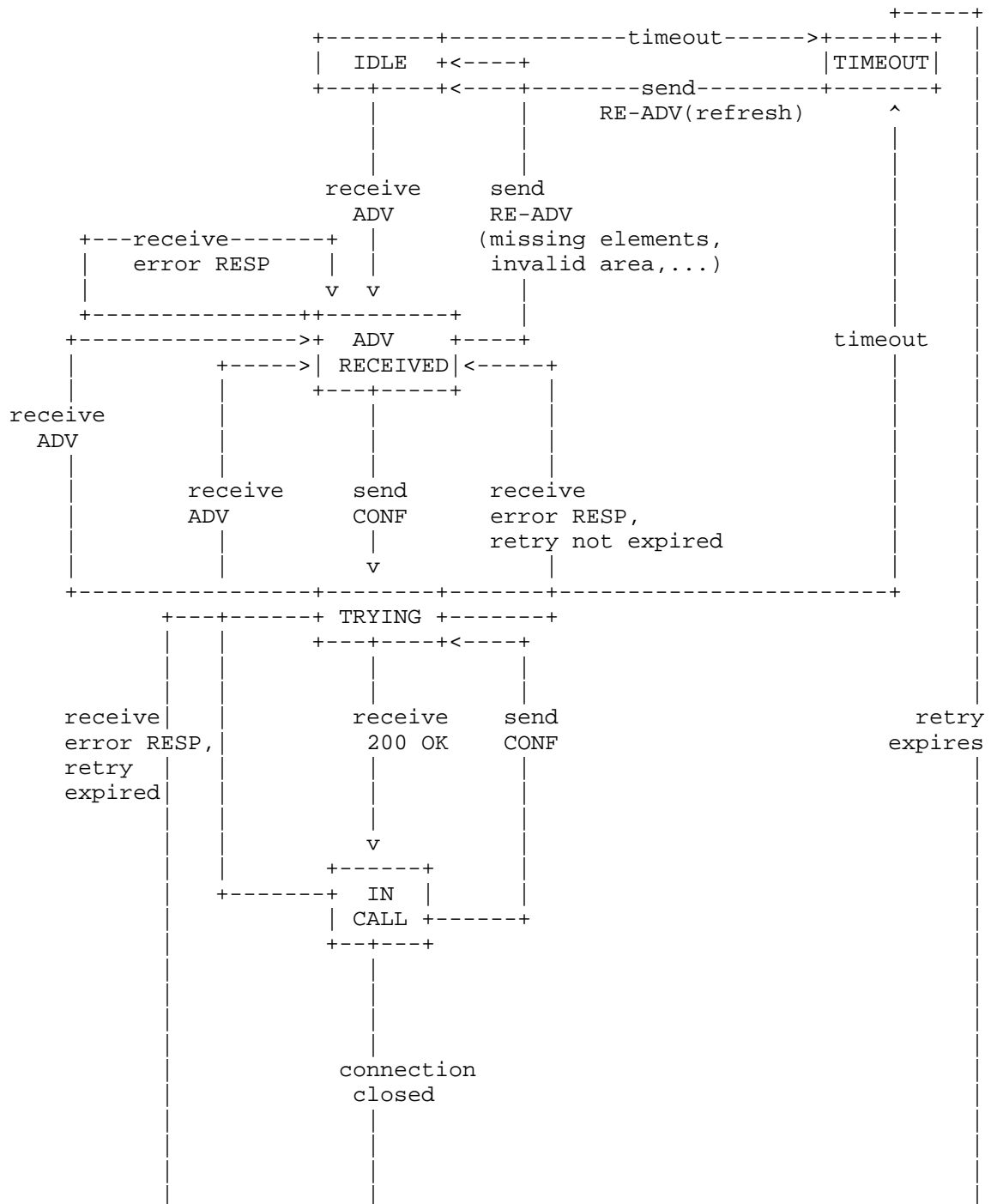
In the TIMEOUT state, if the number of trials is below the retry threshold, the MC sends a RE-ADV/refresh message to the MP ("send RE-ADV"), switching back to the IDLE state. Otherwise, the MC moves to the TERMINATED state.

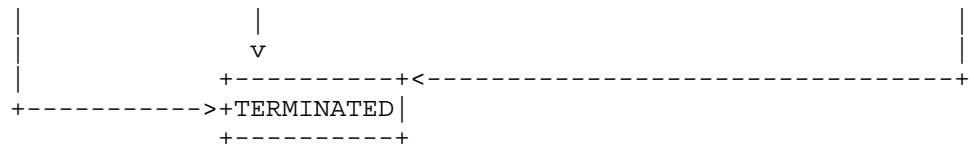
When the ADV has been received ("receive ADV"), the MC goes into the ADV RECEIVED state. The ADV is then parsed. If something goes wrong with the ADV (bad syntax, missing XML elements, etc.), the MC sends a RE-ADV message to the MP specifying the encountered problem via a proper reason phrase. In this 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.





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.

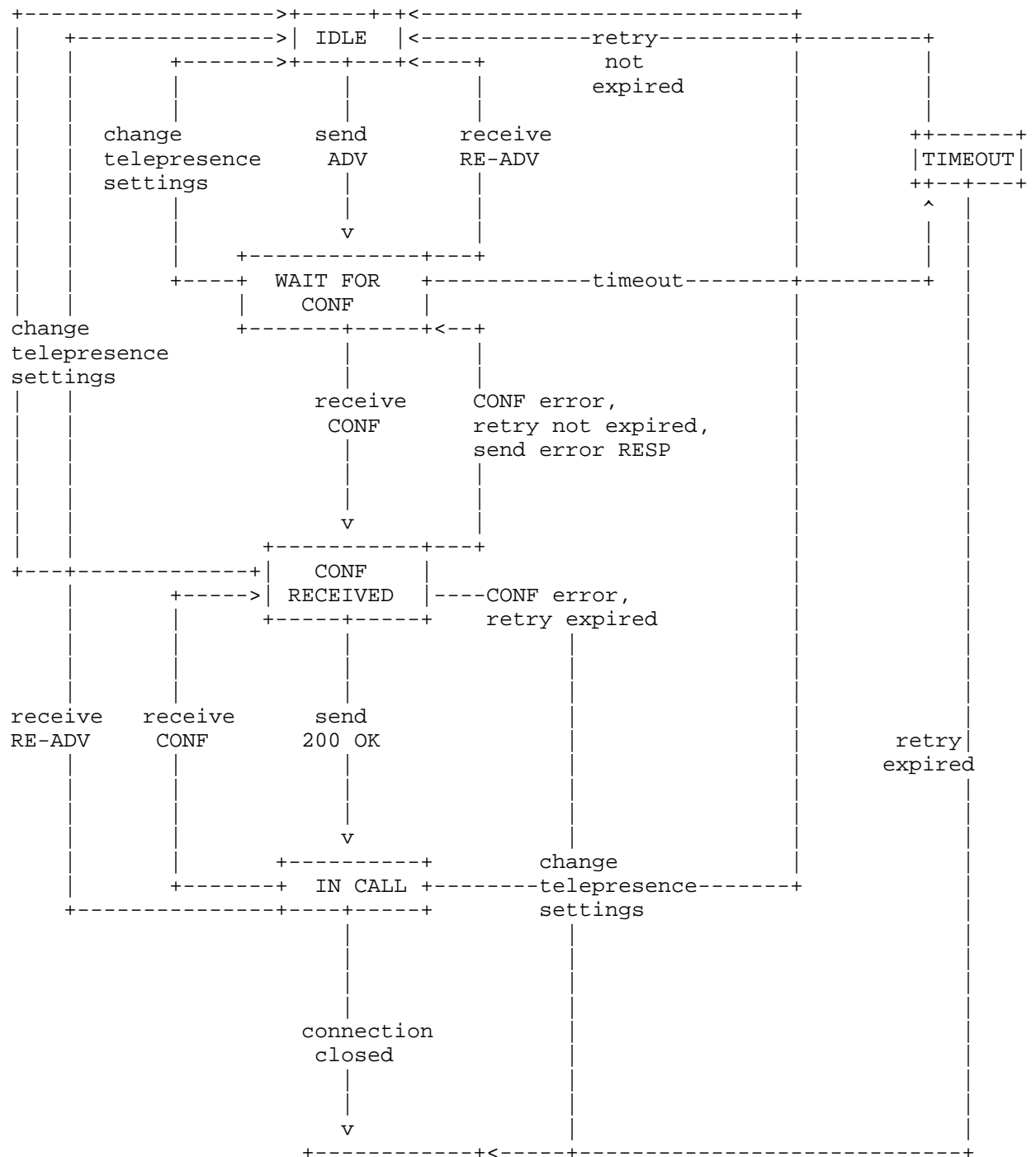
When in the WAIT FOR CONF state, the MP is listening to the channel for a CONF coming from the MC. If a RE-ADV is received, the MP goes back to the IDLE state and issues an ADV again. If telepresence settings change in the meanwhile, it moves back to the IDLE state too, and prepares a new ADV to be sent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



```
| TERMINATED |  
+-----+<-----+
```

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 subsequent 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 aspect MUST be explicitly 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 the OPTIONS message and all subsequent CLUE messages must carry that version number. In the (ii) case, it is the MP who has 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>

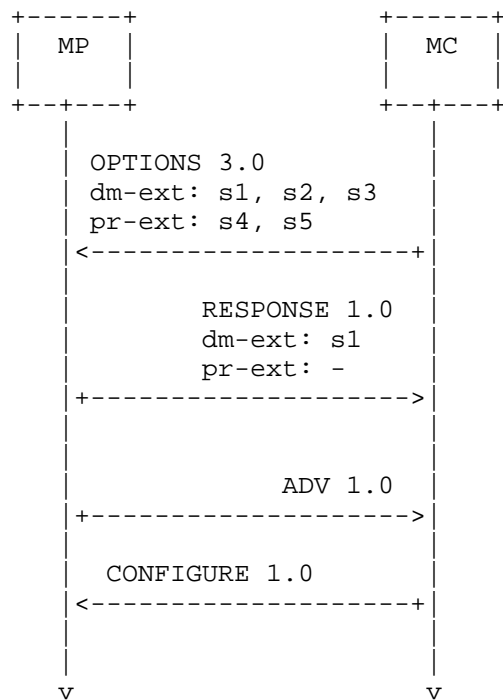
<!-- CLUE OPTIONS -->
<xs:element name="options" type="optionsType"/>

<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 reliable and ordered SCTP over DTLS transport channel connecting two CLUE Participants. Messages carries information taken from the XML-based CLUE data model ([I-D.ietf-clue-data-model-schema]). Three main communication layers can be identified:

1. Establishment of the CLUE data channel: in this phase, the CLUE data channel setup takes place. If it ends up successfully, the

CPs are able to communicate and start the initiation phase.

2. Negotiation of the CLUE protocol version and options (initiation phase): the CPs connected via the CLUE data channel agree on the version and on the options to be used during the telepresence session. Special CLUE messages are used for such a task. At the end of that basic negotiation, each CP starts its activity as a CLUE MP and/or CLUE MC.
3. CLUE telepresence capabilities description and negotiation: in this phase, the MP-MC offer-answer dialogues take place on the data channel by means of the CLUE protocol messages.

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

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

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

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

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

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

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

4. Protocol messages

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

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

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

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

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

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

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

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

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

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

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

4.1. OPTIONS

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

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

The XML Schema of such a message is reported below:


```
<!-- CLUE OPTIONS -->
<xs:complexType name="optionsMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="mediaProvider" type="xs:boolean"/>
        <xs:element name="mediaConsumer" type="xs:boolean"/>
        <xs:element name="supportedVersions" type="versionsListType" minOccurs="0"/>
        <xs:element name="supportedOptions" type="optionsListType" minOccurs="0"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
    </xs: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 within the OPTIONS message, the CI supports only the version declared in the "v" attribute. For example, if the "v" attribute has a value of "3.4" and there is not a <supportedVersions> tag in the OPTIONS message, it means the CI supports only major version 3 with all the minor versions comprised between 3.0 the 3.4 included. If a <supportedVersion> is provided, at least one <version> tag MUST be included.

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

4.2. OPTIONS RESPONSE

The OPTIONS RESPONSE is sent by a CR to a CI as a reply to the OPTIONS message. As depicted in the figure below, the OPTIONS RESPONSE contains mandatorily a response code and a response string indicating the processing result of the OPTIONS message. Following, the CR attaches two boolean tags, <mediaProvider> and <mediaConsumer>, expressing the supported roles in terms of respectively MP and MC, similarly to what the CI does in the OPTIONS message. 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:extension>
  </xs:complexContent>
</xs:complexType>
```

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

4.3. ADVERTISEMENT

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

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

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

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

4.4. ADVERTISEMENT ACKNOWLEDGEMENT

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

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

4.5. CONFIGURE

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

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

In the >advSequenceNr< element is contained the sequence number of the ADVERTISEMENT or of the READV RESPONSE message the CONFIGURE refers to.

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

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

4.6. CONFIGURE RESPONSE

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

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

4.7. READV

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

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

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

4.8. READV RESPONSE

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

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

4.9. Response codes and reason strings

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

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

... TBC.

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

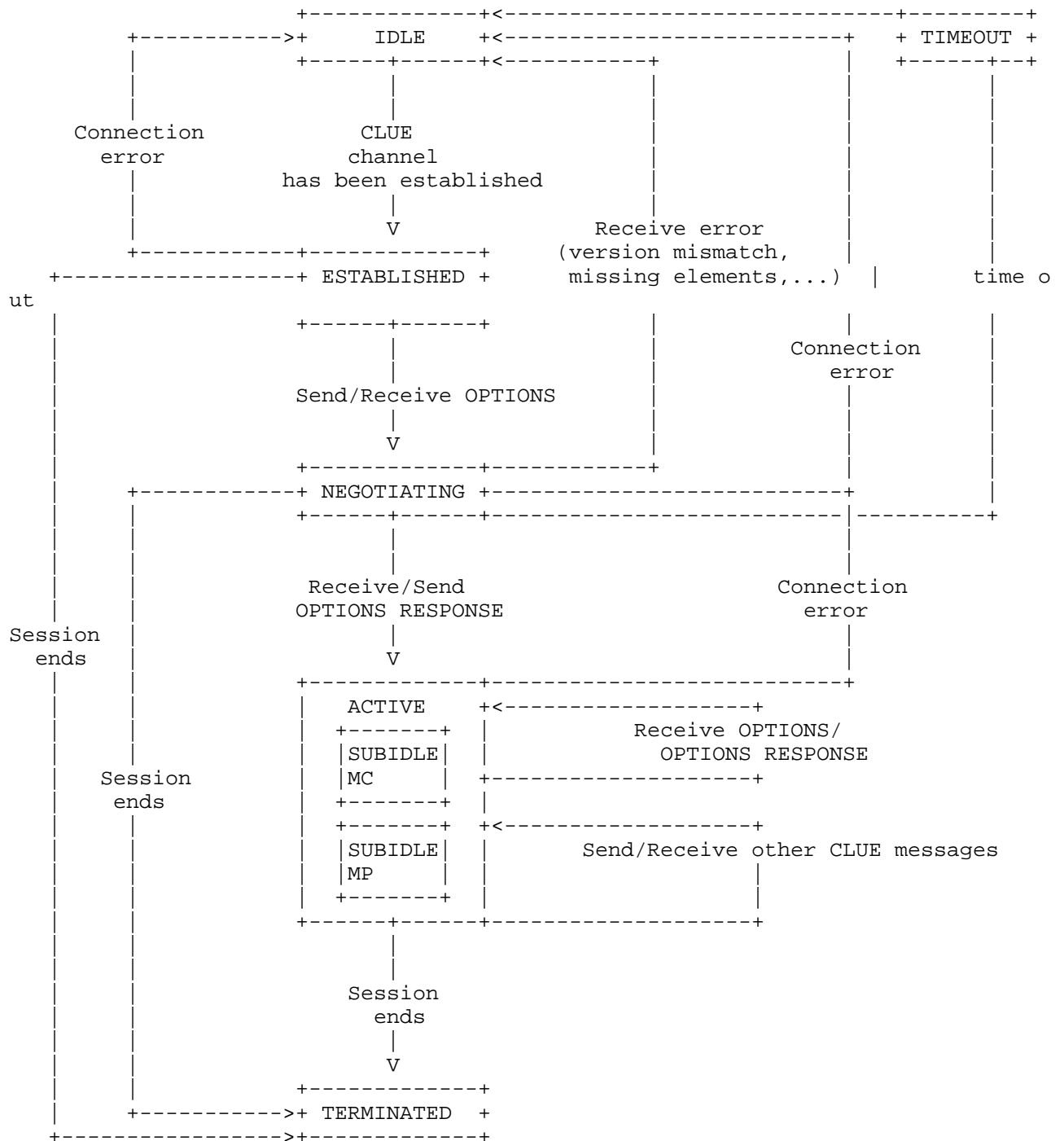
5. Protocol state machines

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

6. CLUE Participant's state machine

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

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



6.1. Media Consumer's state machine

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

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

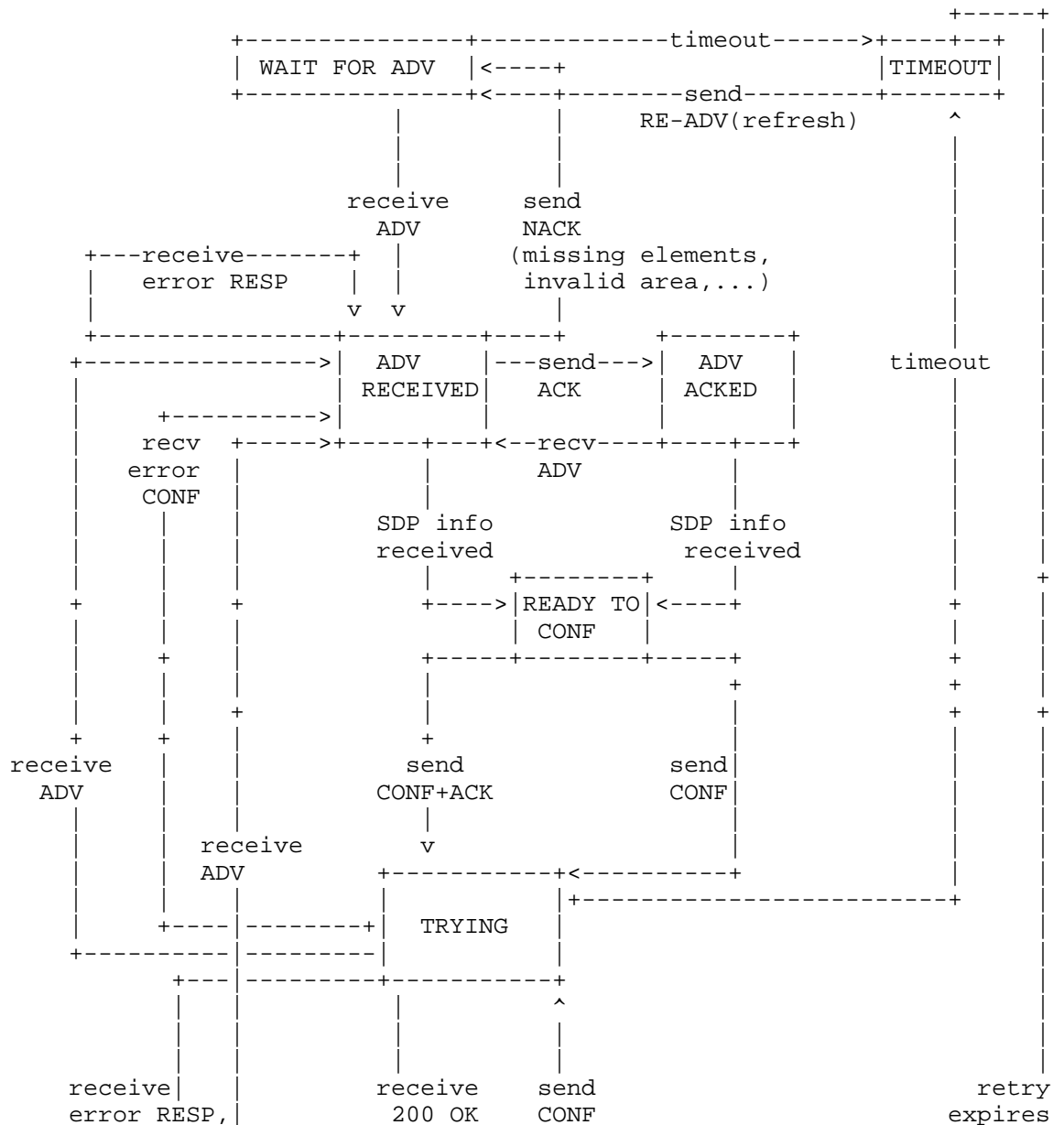
When the ADV has been received ("receive ADV"), the MC goes into the ADV RECEIVED state. The ADV is then parsed. If something goes wrong with the ADV (bad syntax, missing XML elements, etc.), the MC sends a NACK message (an ACK with an error response code) to the MP specifying the encountered problem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

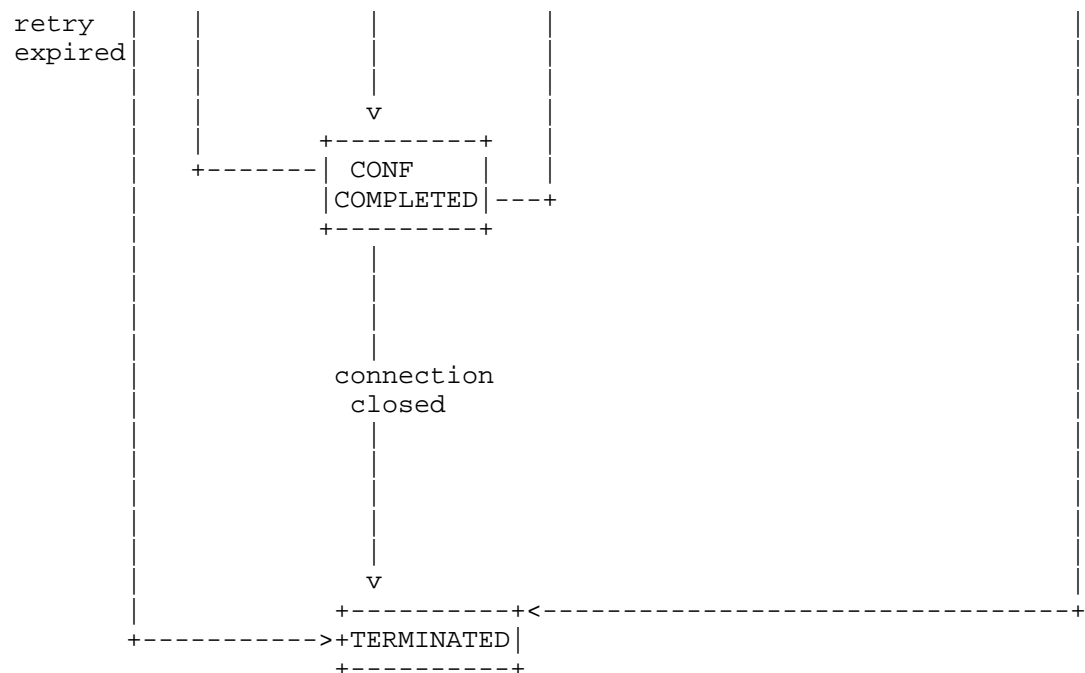
While in the TRYING state, the MC is waiting for a CONF RESPONSE message (to the issued CONF) from the MP. If the timeout expires ("timeout"), the MC moves to the TIMEOUT state and sends a READV in order to solicit a new ADV from the MP. If a CONF RESPONSE with an error code is received ("receive 4xx,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

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

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

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.

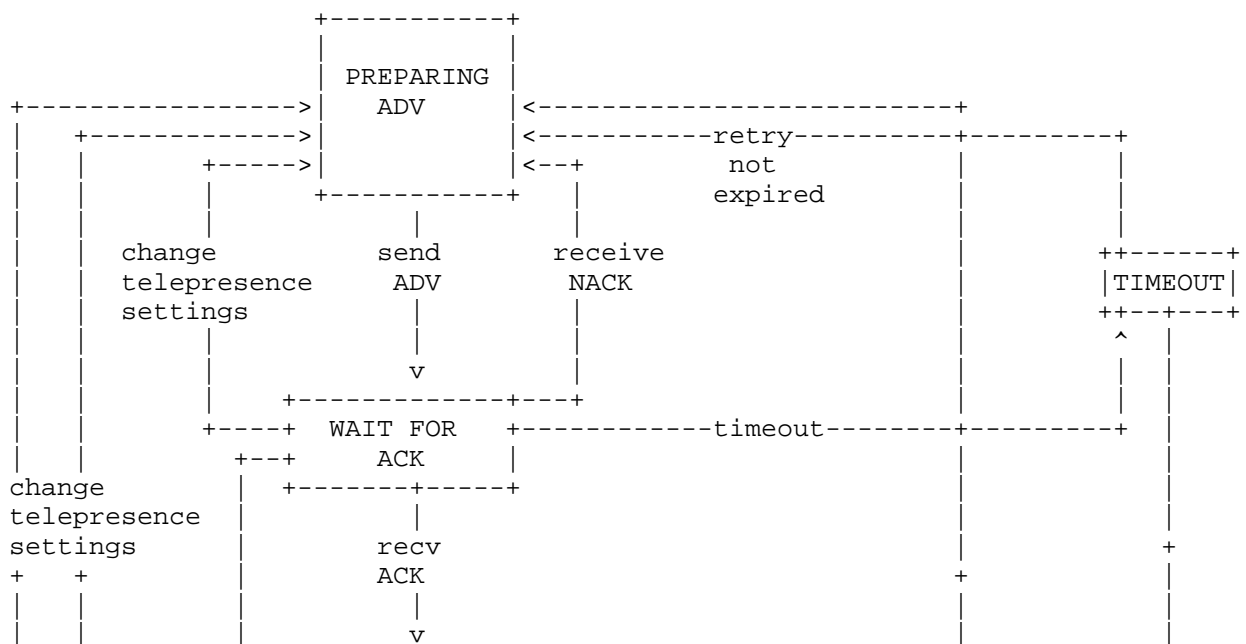
When in the WAIT FOR CONF state, the MP is listening to the channel for a CONF coming from the MC. If a RE-ADV is received, the MP goes back to the IDLE state and issues an ADV again. If telepresence settings change in the meanwhile, it moves back to the PREPARING ADV state and prepares a new ADV to 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.

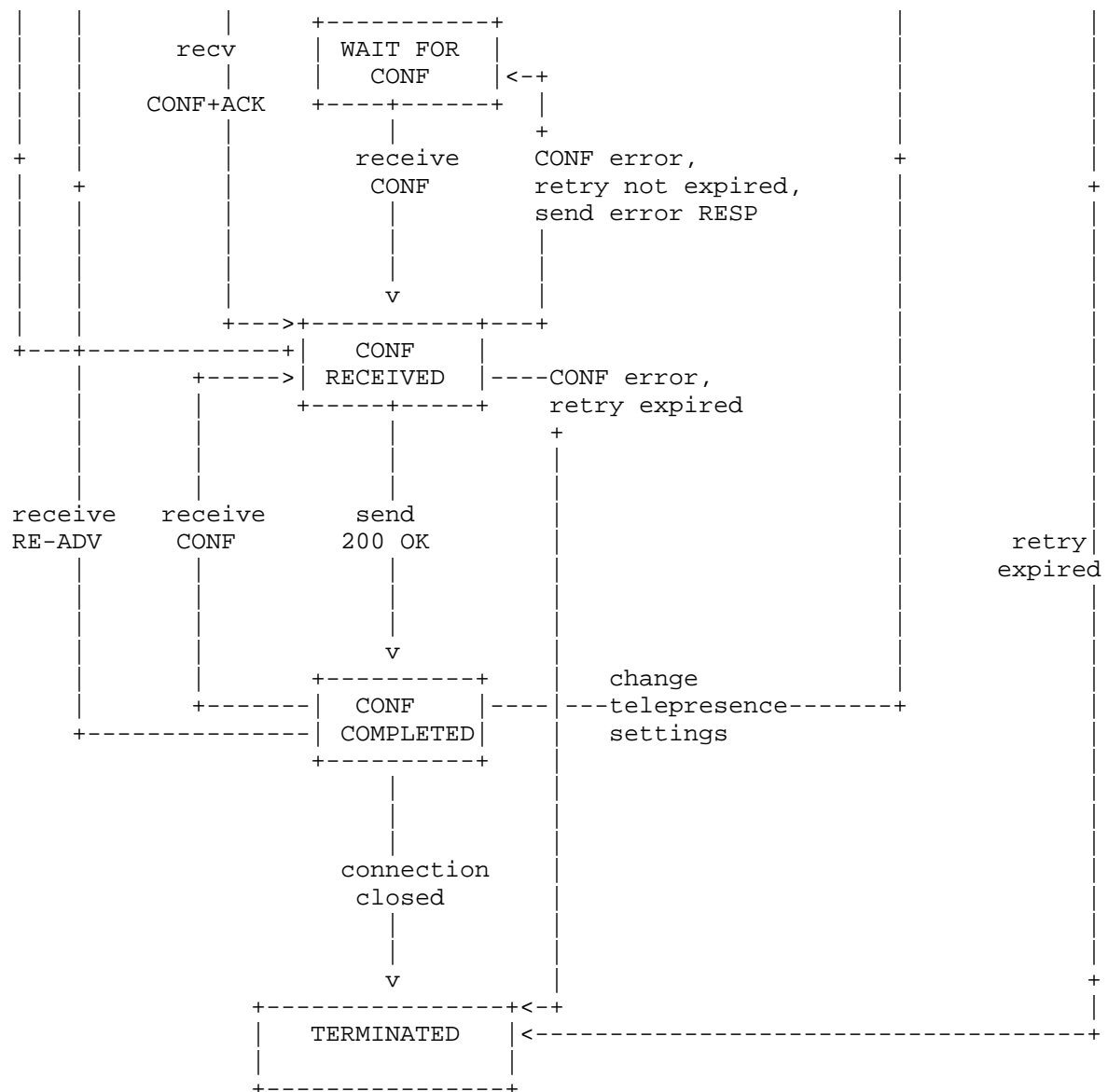
In the TIMEOUT state, if the number of trials does not exceed the retry threshold, the MC comes back to the PREPARING ADV state for sending a new ADV. Otherwise, it goes to the TERMINATED state.

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 during CONF processing, then the MC returns a CONF RESPONSE carrying an error response code. Finally, if there are changes in the telepresence settings, it goes back to the PREPARING ADV state to issue an updated ADV.

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

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





7. Versioning

CLUE protocol messages are XML messages compliant to the CLUE protocol XML schema. The version of the protocol corresponds to the version of the schema. Both client and server have to test the

compliance of the received messages with the XML schema of the CLUE protocol. If the compliance is not verified, the message cannot be processed further.

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

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

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

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

8. Extensions and options

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

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

the meaning of the messages. This is the case if there is no proper message for a certain task, so a brand new CLUE message needs to be defined.

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

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

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

```
elementFormDefault="qualified"
```

```
attributeFormDefault="unqualified"
```

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

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

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

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

extensions, can be characterized by:

a name;

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

the standard version of the protocol the extension refers to.

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

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

9. XML Schema

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

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  version="0.02"
  targetNamespace="urn:ietf:params:xml:ns:clue-message"
  xmlns:tns="urn:ietf:params:xml:ns:clue-message"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:dm="urn:ietf:params:xml:ns:clue-info"
  xmlns="urn:ietf:params:xml:ns:clue-message"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

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

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

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

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

```
<!-- CLUE MESSAGE TYPE -->
```

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

```
<!-- CLUE OPTIONS -->
```

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

```
<!-- VERSIONS LIST TYPE -->
```

```
<xs:complexType name="versionsListType">
  <xs:sequence>
    <xs:element name="version" type="xs:string" minOccurs="1"
      maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

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

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

<!-- CLUE OPTIONS RESPONSE (2 WAY) -->
<xs:complexType name="optionsResponseMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="responseCode" type="xs: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>
```



```
</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" minOccurs="0"/>
        <xs:element name="participants" type="dm:participantsType" minOccurs="0"/>
        <xs:any namespace="##other"
          processContents="lax" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

</xs:schema>
```

10. 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

