Stream: Internet Engineering Task Force (IETF)
RFC: 8847
Category: Experimental
Published: August 2020
ISSN: 2070-1721
Authors: R. Presta, *University of Napoli*
S P. Romano, *University of Napoli*

RFC 8847
Protocol for Controlling Multiple Streams for Telepresence (CLUE)

Abstract

The Controlling Multiple Streams for Telepresence (CLUE) protocol is an application protocol conceived for the description and negotiation of a telepresence session. The design of the CLUE protocol takes into account the requirements and the framework defined within the IETF CLUE Working Group. A companion document, RFC 8848, delves into CLUE signaling details as well as the SIP / Session Description Protocol (SDP) session establishment phase. CLUE messages flow over the CLUE data channel, based on reliable and ordered SCTP-over-DTLS transport. ("SCTP" stands for "Stream Control Transmission Protocol".) 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 document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8847.

Copyright Notice

Table of Contents

1. Introduction

The Controlling Multiple Streams for Telepresence (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 as follows:

1. enabling a Media Provider (MP) to properly announce its current telepresence capabilities to a Media Consumer (MC) in terms of available media captures, groups of encodings, simultaneity constraints, and other information defined in [RFC8845].
2. enabling an MC to request the desired multimedia streams from the offering MP.

CLUE-capable endpoints are connected by means of the CLUE data channel -- an SCTP-over-DTLS channel that is opened and established as described in [RFC8848] and [RFC8850]. ("SCTP" stands for "Stream Control Transmission Protocol".) CLUE protocol messages flowing over such a channel are detailed in this document, both syntactically and semantically.

In Section 4, we provide a general overview of the CLUE protocol. CLUE protocol messages are detailed in Section 5. The CLUE protocol state machines are introduced in Section 6. Versioning and extensions are discussed in Sections 7 and 8, respectively. The XML schema [W3C.REC-xml-20081126] defining the CLUE messages is provided in Section 9.

2. Terminology

This document refers to terminology that is also used in [RFC8845] and [RFC7262]. For convenience, we list those terms below. The definition of "CLUE Participant", as also listed below, originates from this document.

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

CLUE Participant (CP): An entity able to use the CLUE protocol within a telepresence session. It can be an endpoint or an MCU (Multipoint Control Unit) able to use the CLUE protocol.

CLUE-capable device: A device that (1) supports the CLUE data channel [RFC8850], the CLUE protocol, and the principles of CLUE negotiation and (2) seeks CLUE-enabled calls.

Endpoint: A CLUE-capable device that is the logical point of final termination through receiving, decoding, and rendering, and/or initiation through the capturing, encoding, and sending of media streams. An endpoint consists of one or more physical devices that source and sink media streams, and exactly one participant (as described in [RFC4353]) that, in turn, includes exactly one user agent [RFC3261]. Endpoints can be anything from multiscreen/multicamera rooms to handheld devices.

Multipoint Control Unit (MCU): A CLUE-capable device that connects two or more endpoints together into one single multimedia conference [RFC7667]. An MCU includes a mixer (as defined in [RFC4353]), without the requirement per [RFC4353] 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 -- for example, from one or more Capture Devices or constructed from other Media streams.

Media Consumer (MC): A CP (i.e., an Endpoint or an MCU) able to receive Capture Encodings.

Media Provider (MP): A CP (i.e., an Endpoint or an MCU) able to send Capture Encodings.

Stream: A Capture Encoding sent from an MP to an MC via RTP [RFC3550].

3. Conventions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

4. Overview of the CLUE Protocol

The CLUE protocol is conceived to enable CLUE telepresence sessions. It is designed to address Session Description Protocol (SDP) limitations in terms of the description of some information about the multimedia streams that are involved in a real-time multimedia conference. Indeed, by simply using SDP, it is not possible to convey information about the features of the flowing multimedia streams that are needed to enable a "being there" rendering experience. Such information is contained in the CLUE framework document [RFC8845] and formally defined and described in the CLUE data model document [RFC8846]. The CLUE protocol represents the mechanism for the exchange of telepresence information between CPs. It mainly provides the messages to enable an MP to advertise its telepresence capabilities and to enable an MC to select the desired telepresence options.

The CLUE protocol, as defined in this document and further described below, is a stateful client-server XML-based application protocol. CLUE protocol messages flow on a reliable and ordered SCTP-over-DTLS transport channel connecting two CPs. Messages carry information taken from the XML-based CLUE data model [RFC8846]. Three main communication phases can be identified:

Establishment of the CLUE data channel:
: In this phase, the CLUE data channel setup takes place. If it completes successfully, the CPs are able to communicate and start the initiation phase.

Negotiation of the CLUE protocol version and extensions (initiation phase):
: The CPs connected via the CLUE data channel agree on the protocol version and extensions to be used during the telepresence session. Special CLUE messages are used for such a task ('options' and 'optionsResponse'). The negotiation of the version and extensions can be performed once during the CLUE session and only at this stage. At the end of that basic negotiation, each CP starts its activity as a CLUE MP and/or CLUE MC.

Description and negotiation of CLUE telepresence capabilities:
: In this phase, the MP-MC dialogues take place on the data channel by means of the CLUE protocol messages.

As soon as the channel is ready, the CPs must agree on the protocol version and extensions to be used within the telepresence session. CLUE protocol version numbers are characterized by a major version number and a minor version number, both unsigned integers, separated by a dot. While minor version numbers denote backward-compatible changes in the context of a given

major version, different major version numbers generally indicate a lack of interoperability between the protocol implementations. In order to correctly establish a CLUE dialogue, the involved CPs must have in common a major version number (see Section 7 for further details). The subset of the extensions that are allowed within the CLUE session is also determined in the initiation phase. It includes only the extensions that are supported by both parties. A mechanism for the negotiation of the CLUE protocol version and extensions is part of the initiation phase. According to such a solution, the CP that is the CLUE Channel Initiator (CI) issues a proper CLUE message ('options') to the CP that 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 the negotiation phase is completed, CPs describe and agree on the media flows to be exchanged. In many cases, CPs will seek to both transmit and receive media. Hence, in a call between two CPs (e.g., CPs A and B), there would be two separate message exchange sequences, as follows:

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

CLUE messages for the media session description and negotiation are designed by considering the MP side to be 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 [RFC8845]. The CLUE message conveying the MP's multimedia offer is the 'advertisement' message. Such a message leverages the XML data model definitions provided in [RFC8846].

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 needed mechanisms and operations. Such messages are detailed in the following sections.

5. 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 in Section 9. This section includes non-normative excerpts of the schema to aid in describing it.

The XML definitions of the CLUE information provided in [RFC8846] are included within some CLUE protocol messages (namely the 'advertisement' and 'configure' messages), in order to use the concepts defined in [RFC8845].

The CLUE protocol messages are as follows:

- options
- optionsResponse
- advertisement
- ack
- configure
- configureResponse

While the 'options' and 'optionsResponse' messages are exchanged in the initiation phase between the CPs, the other messages are involved in MP-MC dialogues. Please note that the word "dialogue" as used in this document is not related to SIP's usage of the same term. It refers to message exchange sequences between a CLUE MP and a Clue MC.

Each CLUE message inherits a basic structure, as depicted in the following excerpt (Figure 1):

```
<xs:complexType name="clueMessageType" abstract="true">
  <xs:sequence>
    <xs:element name="clueId" type="xs:string" minOccurs="0"/>
    <xs:element name="sequenceNr" type="xs:positiveInteger"/>
  </xs:sequence>
  <xs:attribute name="protocol" type="xs:string" fixed="CLUE"
        use="required"/>
  <xs:attribute name="v" type="versionType" use="required"/>
</xs:complexType>

<!-- VERSION TYPE -->
<xs:simpleType name="versionType">
  <xs:restriction base="xs:string">
    <xs:pattern value="[1-9][0-9]*\.[0-9]+" />
  </xs:restriction>
</xs:simpleType>
```

Figure 1: Structure of a CLUE Message

The information contained in each CLUE message is as follows:

clueId: An optional XML element containing the identifier (in the form of a generic string) of the CP within the telepresence system.

sequenceNr: An XML element containing the local message sequence number. The sender **MUST** increment the sequence number by one for each new message sent, and the receiver **MUST** remember the most recent sequence number received and send back a 402 error if it receives a message with an unexpected sequence number (e.g., sequence number gap, repeated sequence number, sequence number too small). The initial sequence number can be chosen randomly by each party.

protocol: A mandatory attribute set to "CLUE", identifying the protocol the messages refer to.

v: A mandatory attribute carrying the version of the protocol. The content of the "v" attribute is composed of the major version number followed by a dot and then by the minor version number of the CLUE protocol in use. The major number cannot be "0", and if it is more than one digit, it cannot start with a "0". Allowed values of this kind are "1.3", "2.0", "20.44", etc. This document describes version 1.0.

Each CP is responsible for creating and updating up to three independent streams of sequence numbers in messages it sends: (i) one for the messages sent in the initiation phase, (ii) one for the messages sent as an MP (if it is acting as an MP), and (iii) one for the messages sent as an MC (if it is acting as an MC).

In particular, CLUE response messages ('optionsResponse', 'ack', 'configureResponse') derive from a base type, inheriting from the clueMessageType, which is defined as follows (Figure 2):

```
<xs:complexType name="clueResponseType">
 <xs:complexContent>
  <xs:extension base="clueMessageType">
   <xs:sequence>
    <xs:element name="responseCode" type="responseCodeType"/>
    <xs:element name="reasonString" type="xs:string" minOccurs="0"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Figure 2: Structure of CLUE Response Messages

The elements <responseCode> and <reasonString> are populated as detailed in Section 5.7.

5.1. 'options'

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

<mediaProvider>: A mandatory boolean field set to "true" if the CP is able to act as an MP.

<mediaConsumer>: A mandatory boolean field set to "true" if the CP is able to act as an MC.

<supportedVersions>: The list of supported versions.

<supportedExtensions>: The list of supported extensions.

The XML schema of such a message is shown below (Figure 3):

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

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

<!-- EXTENSIONS LIST TYPE -->
<xs:complexType name="extensionsListType">
  <xs:sequence>
    <xs:element name="extension" type="extensionType" minOccurs="1"
        maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

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

Figure 3: Structure of a CLUE 'options' Message

<supportedVersions> contains the list of versions that are supported by the CI, each one represented in a child <version> element. The content of each <version> element is a string made of the major version number followed by a dot and then by the minor version number (e.g., 1.3 or 2.4). Exactly one <version> element **MUST** be provided for each major version supported, containing the maximum minor version number of such a version, since all minor versions are backward compatible. If no <supportedVersions> is carried within the 'options' message, the CI supports only the version declared in the "v" attribute and all the versions having the same major version number and lower minor version number. For example, if the "v" attribute has a value of "3.4" and there is no <supportedVersions> element in the 'options' message, it means the CI supports only major version 3 with all minor versions from 3.0 through 3.4. If <supportedVersions> is provided, at least one <version> element **MUST** be included. In this case, the "v" attribute **SHOULD** be set to the largest minor version of the smallest major version advertised in the <supportedVersions> list. Indeed, the intention behind the "v" attribute is that some implementation that receives a version number in the "v" field with a major number higher than it understands is supposed to close the connection, since it runs a risk of misinterpreting the contents of messages. The minor version is less useful in this context, since minor versions are defined to be both backward and forward compatible and the value can in any case be parsed out of the <version> list. It is more useful to know the highest minor version supported than some random minor version, as it indicates the full feature set that is supported.

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

5.2. 'optionsResponse'

The 'optionsResponse' (Figure 4) is sent by a CR to a CI as a reply to the 'options' message. The 'optionsResponse' contains a mandatory response code and a reason string indicating the processing result of the 'options' message. If the responseCode is between 200 and 299 inclusive, the response **MUST** also include <mediaProvider>, <mediaConsumer>, <version>, and <commonExtensions> elements; it **MAY** include them for any other response code. <mediaProvider> and <mediaConsumer> elements (which are of a boolean nature) are associated with the supported roles (in terms of the MP and the MC, respectively), similarly to what the CI does in the 'options' message. The <version> element indicates the highest commonly supported version number. The content of the <version> element **MUST** be a string made of the major version number followed by a dot and then by the minor version number (e.g., 1.3 or 2.4). Finally, the commonly supported extensions are copied in the <commonExtensions> element.

```
<!-- CLUE 'optionsResponse' -->
<xs:complexType name="optionsResponseMessageType">
  <xs:complexContent>
    <xs:extension base="clueResponseType">
      <xs:sequence>
        <xs:element name="mediaProvider" type="xs:boolean"
                minOccurs="0"/>
        <xs:element name="mediaConsumer" type="xs:boolean"
                minOccurs="0"/>
        <xs:element name="version" type="versionType" minOccurs="0"/>
        <xs:element name="commonExtensions" type="extensionsListType"
                minOccurs="0"/>
        <xs:any namespace="##other" processContents="lax"
                minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figure 4: Structure of a CLUE 'optionsResponse' Message

Upon reception of the 'optionsResponse', the version to be used is the one provided in the <version> element of the message. The subsequent CLUE messages **MUST** use such a version number in the "v" attribute. The allowed extensions in the CLUE dialogue are those indicated in the <commonExtensions> element of the 'optionsResponse' message.

5.3. 'advertisement'

The 'advertisement' message is used by each MP to advertise the available media captures and related information to the corresponding MC. The MP sends an 'advertisement' to the MC as soon as it is ready after the successful completion of the initiation phase, i.e., as soon as the CPs have agreed on the version and extensions of the CLUE protocol. During a single CLUE session, an MP may send new 'advertisement' messages to replace the previous advertisement if, for instance, its CLUE telepresence media capabilities change mid-call. A new 'advertisement' completely replaces the previous 'advertisement'.

The 'advertisement' structure is defined in the schema excerpt below (Figure 5). The 'advertisement' contains elements compliant with the CLUE data model that characterize the MP's telepresence offer. Namely, such elements are the list of

- media captures (<mediaCaptures>),
- encoding groups (<encodingGroups>),
- capture scenes (<captureScenes>),
- simultaneous sets (<simultaneousSets>),
- global views (<globalViews>), and
- represented participants (<people>).

Each of them is fully described in the CLUE framework document [RFC8845] and formally defined in the CLUE data model document [RFC8846].

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

Figure 5: Structure of a CLUE 'advertisement' Message

5.4. 'ack'

The 'ack' message is sent by an MC to an MP to acknowledge an 'advertisement' message. As can be seen from the message schema provided in the following excerpt (Figure 6), the 'ack' contains a response code and may contain a reason string for describing the processing result of the 'advertisement'. The <advSequenceNr> element carries the sequence number of the 'advertisement' message the 'ack' refers to.

```
<!-- 'ack' MESSAGE TYPE -->
<xs:complexType name="advAcknowledgementMessageType">
  <xs:complexContent>
    <xs:extension base="clueResponseType">
      <xs:sequence>
        <xs:element name="advSequenceNr" type="xs:positiveInteger"/>
        <xs:any namespace="##other" processContents="lax"
                minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figure 6: Structure of a CLUE 'ack' Message

5.5. 'configure'

The 'configure' message is sent from an MC to an MP to list the advertised captures the MC wants to receive. The MC **MUST** send a 'configure' after the reception of an 'advertisement', as well as each time it wants to request other captures that have been previously advertised by the MP. The content of the 'configure' message is shown below (Figure 7).

```
<!-- CLUE 'configure' MESSAGE TYPE -->
<xs:complexType name="configureMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <!-- mandatory fields -->
        <xs:element name="advSequenceNr" type="xs:positiveInteger"/>
        <xs:element name="ack" type="successResponseCodeType"
                minOccurs="0"/>
        <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>
```

Figure 7: Structure of a CLUE 'configure' Message

The <advSequenceNr> element contains the sequence number of the 'advertisement' message the 'configure' refers to.

The optional <ack> element, when present, contains a success response code, as defined in Section 5.7. It indicates that the 'configure' message also acknowledges with success the referred advertisement ('configure+ack' message). The <ack> element **MUST NOT** be present if an 'ack' message (associated with the advertisement carrying that specific sequence number) has already been sent back to the MP.

The most important content of the 'configure' message is the list of capture encodings provided in the <captureEncodings> element (see [RFC8846] for the definition of <captureEncodings>). Such an element contains a sequence of capture encodings, representing the streams to be instantiated.

5.6. 'configureResponse'

The 'configureResponse' message is sent from the MP to the MC to communicate the processing result of requests carried in the previously received 'configure' message. As shown in Figure 8, it contains a response code (and, optionally, a reason string) indicating either the success or failure (along with failure details) of the 'configure' request processing. The <confSequenceNr> element that follows contains the sequence number of the 'configure' message the response refers to. There is no partial execution of commands. As an example, if an MP is able to understand all the selected capture encodings except one, then the whole command fails and nothing is instantiated.

```
<!-- 'configureResponse' MESSAGE TYPE -->
<xs:complexType name="configureResponseMessageType">
  <xs:complexContent>
    <xs:extension base="clueResponseType">
      <xs:sequence>
        <xs:element name="confSequenceNr"
              type="xs:positiveInteger" />
        <xs:any namespace="##other" processContents="lax"
              minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figure 8: Structure of a CLUE 'configureResponse' Message

5.7. Response Codes and Reason Strings

Response codes are defined as a sequence of three digits. A well-defined meaning is associated with the first digit. Response codes beginning with "2" are associated with successful responses. Response codes that do not begin with either "2" or "1" indicate an error response, i.e., that an error occurred while processing a CLUE request. In particular, response codes beginning with "3" indicate problems with the XML content of the message ("Bad syntax", "Invalid value", etc.), while response codes beginning with "4" refer to problems related to CLUE protocol semantics ("Invalid sequencing", "Version not supported", etc.). 200, 300, and 400 codes are the most generic

codes in their respective categories. Further response codes can be defined either in future versions of the protocol or by leveraging the extension mechanism. In both cases, the new response codes **MUST** be registered with IANA. Such new response codes **MUST NOT** override the codes defined in this document, and they **MUST** respect the semantics of the first code digit.

This document does not define response codes starting with "1", and such response codes are not allowed to appear in major version 1 of the CLUE protocol. The range from 100 to 199 inclusive is reserved for future major versions of the protocol to define response codes for delayed or incomplete operations, if necessary. Response codes starting with "5" through "9" are reserved for future major versions of the protocol to define new classes of responses and are not allowed in major version 1 of the CLUE protocol. Response codes starting with "0" are not allowed.

The response codes and reason strings defined for use with version 1 of the CLUE protocol are listed in Table 1. The "Description" text contained in the table can be sent in the <reasonString> element of a response message. Implementations can (and are encouraged to) include descriptions of the error condition that are more specific, if possible.

Response Code	Reason String	Description
200	Success	The request has been successfully processed.
300	Low-level request error	A generic low-level request error has occurred.
301	Bad syntax	The XML syntax of the message is not correct.
302	Invalid value	The message contains an invalid parameter value.
303	Conflicting values	The message contains values that cannot be used together.
400	Semantic errors	The received CLUE protocol message contains semantic errors.
401	Version not supported	The protocol version used in the message is not supported.
402	Invalid sequencing	The received message contains an unexpected sequence number (e.g., sequence number gap, repeated sequence number, or sequence number outdated).
403	Invalid identifier	The clueId used in the message is invalid or unknown.
404	Advertisement expired	The sequence number of the advertisement the 'configure' message refers to is out of date.

Response Code	Reason String	Description
405	Subset choice not allowed	The subset choice is not allowed for the specified Multiple Content Capture.

Table 1: CLUE Response Codes

6. 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 over the CLUE data channel, an SCTP-over-DTLS channel established as depicted in [RFC8850]. We herein discuss the state machines associated with the CP (Figure 9), the MP role (Figure 10 in Section 6.1), and the MC role (Figure 11 in Section 6.2), respectively. Endpoints often wish to both send and receive media, i.e., act as both an MP and an MC. As such, there will often be two sets of messages flowing in opposite directions; the state machines of these two flows do not interact with each other. Only the CLUE application logic is considered. The interaction of CLUE protocol and SDP negotiations for the media streams exchanged is discussed in [RFC8848].

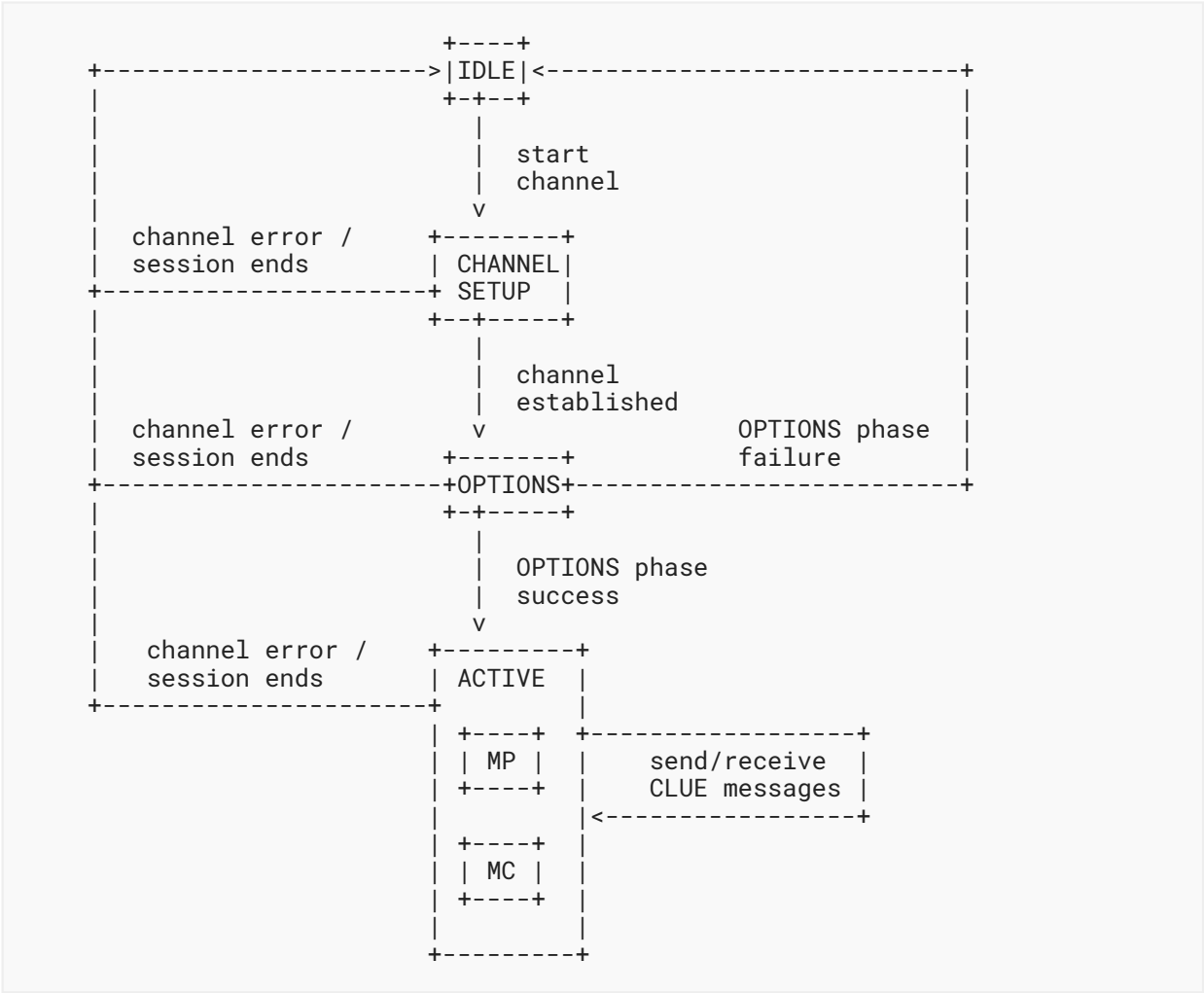


Figure 9: CLUE Participant State Machine

The main state machines focus on the behavior of the CP acting as a CLUE CI/CR.

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

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

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

When in the OPTIONS state, the CP addresses the initiation phase where both parts agree on the version and extensions to be used in the subsequent CLUE message exchange phase. If the CP is the CI, it sends an 'options' message and waits for the 'optionsResponse' message. If the CP is the CR, it waits for the 'options' message and, as soon as it arrives, replies with the 'optionsResponse' message. If the negotiation is successfully completed ("OPTIONS phase success"), the CP moves from the OPTIONS state to the ACTIVE state. If the initiation phase fails ("OPTIONS phase failure"), the CP moves from the OPTIONS state to the IDLE state. The initiation phase might fail for one of the following reasons:

1. The CI receives an 'optionsResponse' with an error response code.
2. The CI does not receive any 'optionsResponse', and a timeout error is raised.
3. The CR does not receive any 'options', and a timeout error is raised.

When in the ACTIVE state, the CP starts the envisioned sub-state machines (i.e., the MP state machine and the MC state machine) according to the roles it plays in the telepresence sessions. Such roles have been previously declared in the 'options' and 'optionsResponse' messages involved in the initiation phase (see Sections 5.1 and 5.2 for details). When in the ACTIVE state, the CP delegates the sending and processing of the CLUE messages to the appropriate MP/MC sub-state machines. If the CP receives a further 'options'/'optionsResponse' message, it **MUST** ignore the message and stay in the ACTIVE state.

6.1. Media Provider's State Machine

As soon as the sub-state machine of the MP (Figure 10) is activated, it is in the ADV state. In the ADV state, the MP prepares the 'advertisement' message reflecting its actual telepresence capabilities.

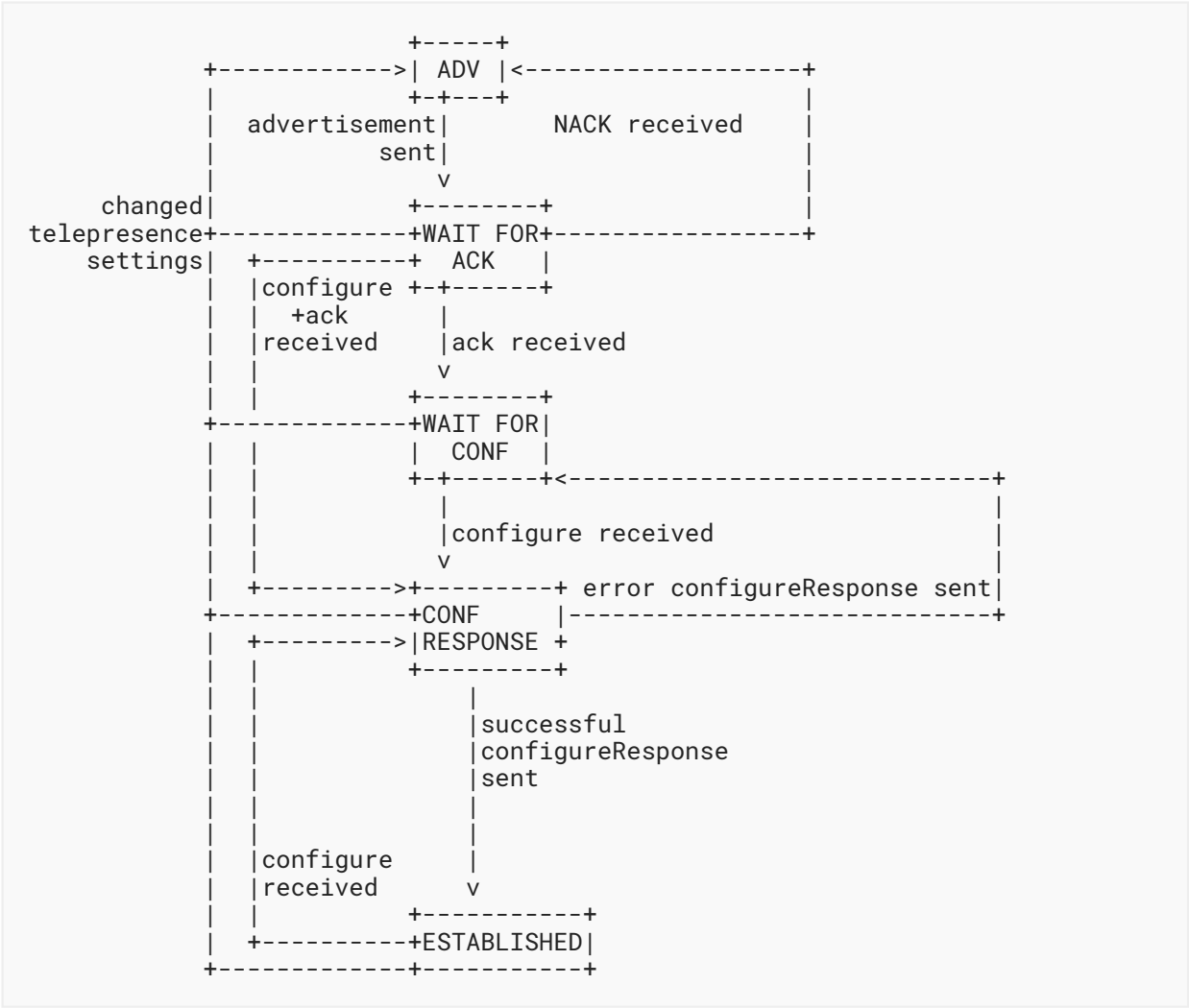


Figure 10: Media Provider's State Machine

After the 'advertisement' has been sent ("advertisement sent"), the MP moves from the ADV state to the WAIT FOR ACK state. If an 'ack' message with a successful response code arrives ("ack received"), the MP moves to the WAIT FOR CONF state. If a NACK arrives (i.e., an 'ack' message with an error response code), the MP moves back to the ADV state for preparation of a new 'advertisement'. When in the WAIT FOR ACK state, if a 'configure' message with the <ack> element set to "200" arrives ("configure+ack received"), the MP goes directly to the CONF RESPONSE state. 'configure+ack' messages referring to out-of-date (i.e., having a sequence number less than the highest generated so far) advertisements **MUST** be ignored, i.e., they do not trigger any state transition. If the telepresence settings of the MP change while in the WAIT FOR ACK state ("changed telepresence settings"), the MP switches from the WAIT FOR ACK state to the ADV state to create a new 'advertisement'.

When in the WAIT FOR CONF state, the MP listens to the channel for a 'configure' request coming from the MC. When a 'configure' arrives ("configure received"), the MP switches to the CONF RESPONSE state. If the telepresence settings change in the meantime ("changed telepresence settings"), the MP moves from the WAIT FOR CONF state back to the ADV state to create the new 'advertisement' to be sent to the MC.

The MP in the CONF RESPONSE state processes the received 'configure' in order to produce a 'configureResponse' message. If the MP successfully processes the MC's configuration, then it sends a 200 'configureResponse' ("successful configureResponse sent") and moves to the ESTABLISHED state. If there are errors in the 'configure' processing, then the MP issues a 'configureResponse' carrying an error response code and goes back to the WAIT FOR CONF state to wait for a new configuration request. Finally, if there are changes in the MP's telepresence settings ("changed telepresence settings"), the MP switches to the ADV state.

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

Messages not shown for a state do not cause the state to change.

6.2. Media Consumer's State Machine

As soon as the sub-state machine of the MC (Figure 11) is activated, it is in the WAIT FOR ADV state. An MC in the WAIT FOR ADV state is waiting for an 'advertisement' coming from the MP. If the 'advertisement' arrives ("ADV received"), the MC moves to the ADV PROCESSING state. Otherwise, the MC stays in the WAIT FOR ADV state.

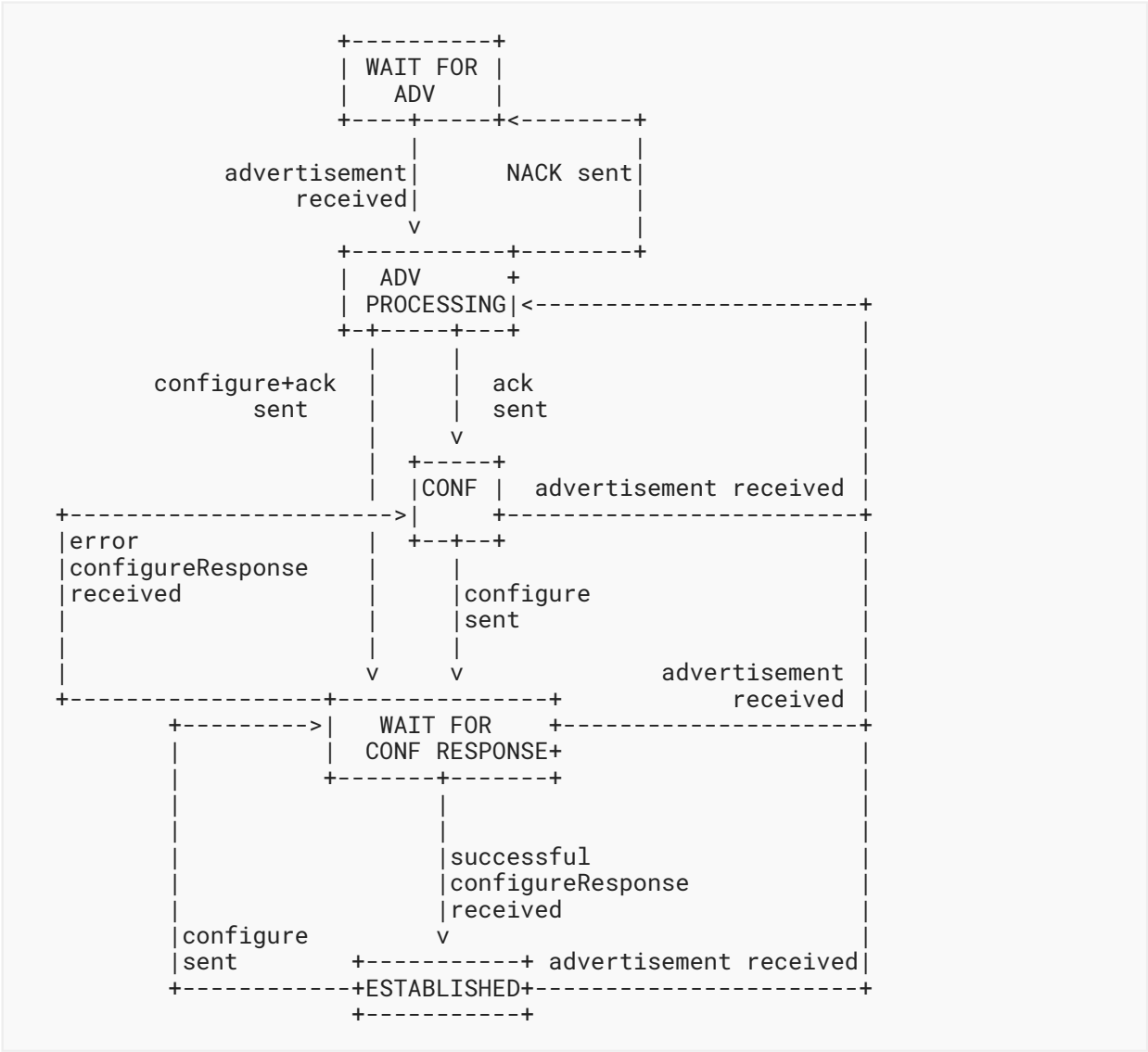


Figure 11: Media Consumer's State Machine

In the ADV PROCESSING state, the 'advertisement' is parsed by the MC. If the 'advertisement' is successfully processed, two scenarios are possible. In the first case, the MC issues a successful 'ack' message to the MP ("ack sent") and moves to the CONF state. This typically happens when the MC needs some more time to produce the 'configure' message associated with the received 'advertisement'. In the latter case, the MC is able to immediately prepare and send back to the MP a 'configure' message. Such a message will have the <ack> element set to "200" ("configure+ack sent") and will allow the MC to move directly to the WAIT FOR CONF RESPONSE state.

If the processing of the 'advertisement' is unsuccessful (bad syntax, missing XML elements, etc.), the MC sends a NACK message (i.e., an 'ack' with an error response code) to the MP and, optionally, further describes the problem via a proper reason phrase. In this scenario ("NACK sent"), the MC switches back to the WAIT FOR ADV state and waits for a new 'advertisement'.

When in the CONF state, the MC prepares the 'configure' request to be issued to the MP on the basis of the previously acked 'advertisement'. When the 'configure' has been sent ("configure sent"), the MC moves to the WAIT FOR CONF RESPONSE state. If a new 'advertisement' arrives in the meantime ("advertisement received"), the MC goes back to the ADV PROCESSING state.

In the WAIT FOR CONF RESPONSE state, the MC waits for the MP's response to the issued 'configure' or 'configure+ack'. If a 200 'configureResponse' message is received ("successful configureResponse received"), it means that the MP and the MC have successfully agreed on the media streams to be shared. Then, the MC can move to the ESTABLISHED state. On the other hand, if an error response is received ("error configureResponse received"), the MC moves back to the CONF state to prepare a new 'configure' request. If a new 'advertisement' is received in the WAIT FOR CONF RESPONSE state, the MC switches to the ADV PROCESSING state.

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

Messages not shown for a state do not cause the state to change.

7. Versioning

CLUE protocol messages are XML messages compliant to the CLUE protocol XML schema [RFC8846]. The version of the protocol corresponds to the version of the schema. Both the client and the 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.

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

This document defines version 1.0 of the CLUE protocol schema, using XML schema version 1.0 [W3C.REC-xml-20081126]. The version usage is similar in philosophy to the Extensible Messaging and Presence Protocol (XMPP) [RFC6120]. A version number has major and minor components, each a non-negative integer. Changes to the major version denote non-interoperable changes. Changes to the minor version 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 the schema but 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 an MP is able to "speak", for example, version 1.5 of the protocol while the MC only understands version 1.4, the MP should have no problem in reverting the dialogue back to version 1.4 without exploiting 1.5 features and functionality. Version 1.4 is the one to be spoken and has to appear in the "v" attribute of the subsequent CLUE messages. In other words, in this example, the MP **MUST** use version 1.4. That said, and in keeping with the general IETF protocol "robustness principle" stating that an implementation must be conservative in its sending behavior and liberal in its receiving behavior [RFC1122], CPs **MUST** ignore unknown elements or attributes that are not envisioned in the negotiated protocol version and related extensions.

8. Extensions

Although the standard version of the CLUE protocol XML schema is designed to thoroughly cope with the requirements emerging from the application domain, new needs might arise, and new extensions can then be designed. Extensions specify information and behaviors that are not described in a certain version of the protocol and specified in the related RFC document. Such information and behaviors can be optionally used in a CLUE dialogue and **MUST** be negotiated in the CLUE initiation phase. They can relate to:

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

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

1. XML elements defined within the CLUE protocol XML schema itself (protocol-specific information).
2. 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 as follows:

- elementFormDefault="qualified"
- attributeFormDefault="unqualified"

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

The other matter concerns data model information. Data model information is defined by the XML schema associated with the URN "urn:ietf:params:xml:ns:clue-info". Note that there are also extensibility issues for the XML elements defined in such a schema. Those issues are overcome

by using <any> and <anyAttribute> placeholders. 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 (the second category of extensions), "extra" CLUE protocol messages, i.e., messages not envisioned in the latest standard version of the schema, might be needed. In that case, the messages and the associated behavior should be defined in external documents that both communication parties must be aware of.

As shown in Figure 12, the fields of the <extension> element (either new information or new messages) take the following values:

- a name.
- an external XML schema defining the XML information and/or the XML messages representing the extension.
- the major standard version of the protocol that the extension refers to.

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

Figure 12: The <extension> Element

The above-described <extension> element is carried within the 'options' and 'optionsResponse' messages to represent the extensions supported by both the CI and the CR.

Extensions **MUST** be defined in a separate XML schema file and **MUST** be provided with a companion document describing their semantics and use.

8.1. Extension Example

An example of an extension might be a "new" capture attribute (i.e., a capture attribute that is not envisioned in the current standard defining the CLUE data model in [RFC8846]) needed to further describe a video capture.

The CLUE data model document [RFC8846] envisions the possibility of adding this kind of "extra" information in the description of a video capture. For convenience, the XML definition of a video capture taken from [RFC8846] is shown in Figure 13 below.

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

Figure 13: XML Definition of a CLUE Video Capture

According to such a definition, a video capture might have, after the set of generic media capture attributes, a set of new attributes defined elsewhere, i.e., in an XML schema defining an extension. The XML schema defining the extension might look like the following (Figure 14):

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema version="1.0"
  targetNamespace="https://example.extensions.com/myVideoExtensions"
  xmlns:xs="https://www.w3.org/2001/XMLSchema"
  xmlns="https://example.extensions.com/myVideoExtensions"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

        <!--
              This is the new element to be put in place of the <any>
              element in the video capture definition
              of the CLUE data model schema
        -->

        <xs:element name="myVideoExtension">
                <xs:complexType>
                        <xs:sequence>
                             <xs:element ref="newVideoAttribute1"/>
                             <xs:element ref="newVideoAttribute2"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>

        <xs:element name="newVideoAttribute1" type="xs:string"/>

        <xs:element name = "newVideoAttribute2" type = "xs:boolean"/>
</xs:schema>
```

Figure 14: XML Schema Defining an Extension

By using the extension above, a video capture can be further described in the advertisement using the <myVideoExtension> element containing two extra pieces of information (<newVideoAttribute1> and <newVideoAttribute2>), besides using the attributes envisioned for a generic media capture. As stated in this document, both participants must be aware of the extension schema and related semantics to use such an extension and must negotiate it via the 'options' and 'optionsResponse' messages.

9. XML Schema

The XML schema defining the CLUE messages is provided below (Figure 15).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="https://www.w3.org/2001/XMLSchema"
        xmlns="urn:ietf:params:xml:ns:clue-protocol"
        xmlns:dm="urn:ietf:params:xml:ns:clue-info"
        version="1.0"
        targetNamespace="urn:ietf:params:xml:ns:clue-protocol"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
  <!-- Import data model schema -->
  <xs:import namespace="urn:ietf:params:xml:ns:clue-info"/>
  <!-- ELEMENT DEFINITIONS -->
  <xs:element name="options" type="optionsMessageType" />
  <xs:element name="optionsResponse"
        type="optionsResponseMessageType"/>
  <xs:element name="advertisement" type="advertisementMessageType"/>
  <xs:element name="ack" type="advAcknowledgementMessageType"/>
  <xs:element name="configure" type="configureMessageType"/>
  <xs:element name="configureResponse"
        type="configureResponseMessageType"/>
  <!-- CLUE MESSAGE TYPE -->
  <xs:complexType name="clueMessageType" abstract="true">
    <xs:sequence>
      <xs:element name="clueId" type="xs:string" minOccurs="0" />
      <xs:element name="sequenceNr" type="xs:positiveInteger" />
    </xs:sequence>
    <xs:attribute name="protocol" type="xs:string" fixed="CLUE"
        use="required" />
    <xs:attribute name="v" type="versionType" use="required" />
  </xs:complexType>
  <!-- CLUE RESPONSE TYPE -->
  <xs:complexType name="clueResponseType">
    <xs:complexContent>
      <xs:extension base="clueMessageType">
        <xs:sequence>
          <xs:element name="responseCode" type="responseCodeType" />
          <xs:element name="reasonString" type="xs:string"
                minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- VERSION TYPE -->
  <xs:simpleType name="versionType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[1-9][0-9]*\.[0-9]+" />
    </xs:restriction>
  </xs:simpleType>
  <!-- RESPONSE CODE TYPE -->
  <xs:simpleType name="responseCodeType">
    <xs:restriction base="xs:integer">
      <xs:pattern value="[1-9][0-9][0-9]" />
    </xs:restriction>
  </xs:simpleType>
  <!-- SUCCESS RESPONSE CODE TYPE -->
  <xs:simpleType name="successResponseCodeType">
    <xs:restriction base="xs:integer">
      <xs:pattern value="2[0-9][0-9]" />
```

```
    </xs:restriction>
  </xs:simpleType>
  <!-- CLUE OPTIONS -->
  <xs:complexType name="optionsMessageType">
    <xs:complexContent>
      <xs:extension base="clueMessageType">
        <xs:sequence>
          <xs:element name="mediaProvider" type="xs:boolean"/>
          <xs:element name="mediaConsumer" type="xs:boolean"/>
          <xs:element name="supportedVersions"
                type="versionsListType"
                minOccurs="0" />
          <xs:element name="supportedExtensions"
                type="extensionsListType" minOccurs="0"/>
          <xs:any namespace="##other" processContents="lax"
                minOccurs="0"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- VERSIONS LIST TYPE -->
  <xs:complexType name="versionsListType">
    <xs:sequence>
      <xs:element name="version" type="versionType" minOccurs="1"
        maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
  <!-- EXTENSIONS LIST TYPE -->
  <xs:complexType name="extensionsListType">
    <xs:sequence>
      <xs:element name="extension" type="extensionType"
        minOccurs="1"
        maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
  <!-- EXTENSION TYPE -->
  <xs:complexType name="extensionType">
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="schemaRef" type="xs:anyURI" />
      <xs:element name="version" type="versionType" />
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <!-- CLUE 'optionsResponse' -->
  <xs:complexType name="optionsResponseMessageType">
    <xs:complexContent>
      <xs:extension base="clueResponseType">
        <xs:sequence>
```

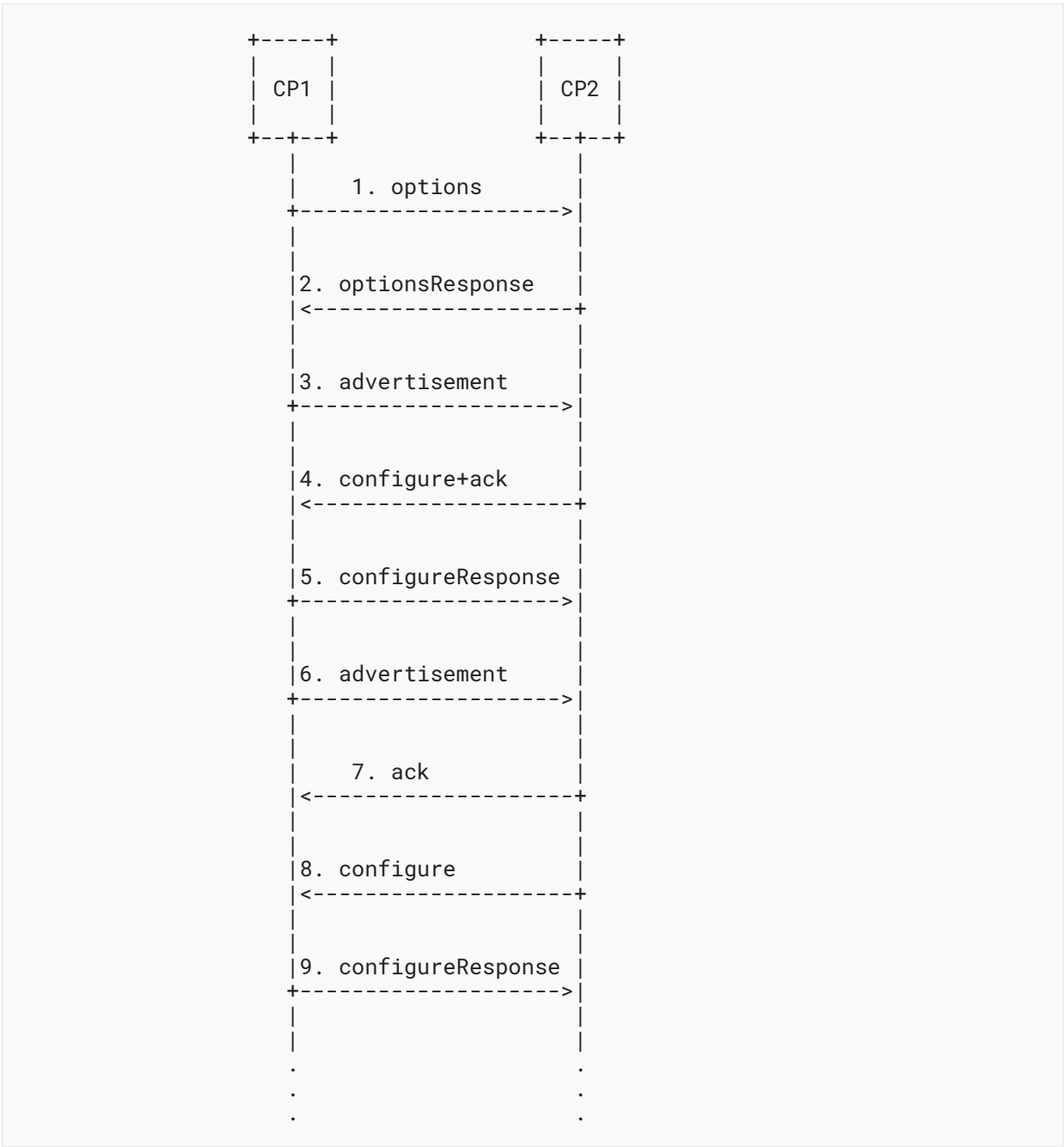
```
          <xs:element name="mediaProvider" type="xs:boolean"
                minOccurs="0"/>
          <xs:element name="mediaConsumer" type="xs:boolean"
                minOccurs="0"/>
          <xs:element name="version" type="versionType"
                minOccurs="0"/>
          <xs:element name="commonExtensions"
                type="extensionsListType" minOccurs="0"/>
          <xs:any namespace="##other" processContents="lax"
                minOccurs="0"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- CLUE ADVERTISEMENT MESSAGE TYPE -->
  <xs:complexType name="advertisementMessageType">
    <xs:complexContent>
      <xs:extension base="clueMessageType">
        <xs:sequence>
          <!-- mandatory -->
          <xs:element name="mediaCaptures"
                type="dm:mediaCapturesType"/>
          <xs:element name="encodingGroups"
                type="dm:encodingGroupsType"/>
          <xs:element name="captureScenes"
                type="dm:captureScenesType"/>
          <!-- optional -->
          <xs:element name="simultaneousSets"
                type="dm:simultaneousSetsType" minOccurs="0"/>
          <xs:element name="globalViews" type="dm:globalViewsType"
                minOccurs="0"/>
          <xs:element name="people"
                type="dm:peopleType" minOccurs="0"/>
          <xs:any namespace="##other" processContents="lax"
                minOccurs="0"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- 'ack' MESSAGE TYPE -->
  <xs:complexType name="advAcknowledgementMessageType">
    <xs:complexContent>
      <xs:extension base="clueResponseType">
        <xs:sequence>
          <xs:element name="advSequenceNr"
                type="xs:positiveInteger"/>
          <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:positiveInteger"/>
          <xs:element name="ack" type="successResponseCodeType"
                minOccurs="0"/>
          <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>
  <!-- 'configureResponse' MESSAGE TYPE -->
  <xs:complexType name="configureResponseMessageType">
    <xs:complexContent>
      <xs:extension base="clueResponseType">
        <xs:sequence>
          <xs:element name="confSequenceNr"
                type="xs:positiveInteger"/>
          <xs:any namespace="##other" processContents="lax"
                minOccurs="0"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

Figure 15: Schema Defining CLUE Messages

10. Call Flow Example

This section describes the CLUE protocol messages exchanged in the following call flow. For simplicity, only one direction of media is shown, as the other direction is precisely symmetric.



Two CPs, CP1 and CP2, have successfully set up the CLUE channel according to [RFC8850]. CP1 is the CI, and CP2 is the CR.

- The initiation phase starts (negotiation of the CLUE protocol version and extensions). CP1, as the CI, sends to CP2 an 'options' message specifying the supported versions and extensions (Section 10.1). CP1 supports (i) version 1.4 with extensions E1, E2, and E3 and (ii) version 2.7 with extensions E4 and E5. Because of such capabilities, CP1 sends an 'options' message with the "v" attribute set to "1.4" and explicitly specifies all the supported versions and extensions in the corresponding fields of the 'options' message. In the 'options' message, CP1 also specifies that it intends to act as both an MP and an MC.
- CP2 supports versions 3.0, 2.9, and 1.9 of the CLUE protocol, each version without any extensions. Version 2.7 is the best common choice. Given the received 'options' message, CP2 answers with an 'optionsResponse' message in which it specifies only version 2.7 as the agreed-upon version of the CLUE protocol to be used, leaving blank the extensions part of the message to say that no extensions will be used in the CLUE session (Section 10.2). In the 'optionsResponse' message, CP2 also specifies that it intends to act as both an MP and an MC.

After the initiation phase is completed, CP1 and CP2 start their activity as the MP and the MC, respectively. For the sake of simplicity, the rest of the call flow focuses only on the dialogue between MP CP1 and MC CP2.

- CP1 advertises a telepresence configuration like the one described in [RFC8846], Section 27, where there are (i) three main video streams captured by three cameras, with the central camera capable of capturing a zoomed-out view of the overall telepresence room, (ii) a multicontent capture of the loudest segment of the room, and (iii) one audio capture for the audio of the whole room (Section 10.3).
- CP2 receives CP1's 'advertisement' message and, after processing it, sends back to CP1 a 'configure+ack' message in which it declares its interest in the multicontent capture and the audio capture only (Section 10.4).
- CP1 answers CP2's 'configure+ack' message with a 'configureResponse' message that includes a 200 (Success) response code to accept all of CP2's requests (Section 10.5).
- To reflect the changes in its telepresence offer, CP1 issues a new 'advertisement' message to CP2 (Section 10.6), this time also adding a composed capture made of a big picture representing the current speaker and two picture-in-picture boxes representing the previous speakers (see [RFC8846], Section 28 for more details regarding the telepresence description).
- CP2 acknowledges the second 'advertisement' message with an 'ack' message (Section 10.7).
- Later in the session, CP2 changes the requested media streams from CP1 by sending to CP1 a 'configure' message replacing the previously selected video streams with the new composed media streams advertised by CP1 (Section 10.8). Media streams from the previous configuration continue to flow during the reconfiguration process.
- Finally, CP1 accepts CP2's latest request with a 'configureResponse' message (Section 10.9).

We would also like to point out that in the depicted flow three distinct sequence number spaces per sender are involved (two of which appear in the snippets, since we only show one direction of media). The discontinuity between the sequence number values in Sections 10.2 and 10.3 is hence correct.

10.1. CLUE Message No. 1: 'options'

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<options xmlns="urn:ietf:params:xml:ns:clue-protocol"
 xmlns:ns2="urn:ietf:params:xml:ns:clue-info"
 xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
 xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 protocol="CLUE" v="1.4">
    <clueId>CP1</clueId>
    <sequenceNr>51</sequenceNr>
    <mediaProvider>true</mediaProvider>
    <mediaConsumer>true</mediaConsumer>
    <supportedVersions>
        <version>1.4</version>
        <version>2.7</version>
    </supportedVersions>
    <supportedExtensions>
        <extension>
                <name>E1</name>
                <schemaRef>URL_E1</schemaRef>
                <version>1.4</version>
        </extension>
        <extension>
                <name>E2</name>
                <schemaRef>URL_E2</schemaRef>
                <version>1.4</version>
        </extension>
        <extension>
                <name>E3</name>
                <schemaRef>URL_E3</schemaRef>
                <version>1.4</version>
        </extension>
        <extension>
                <name>E4</name>
                <schemaRef>URL_E4</schemaRef>
                <version>2.7</version>
        </extension>
        <extension>
                <name>E5</name>
                <schemaRef>URL_E5</schemaRef>
                <version>2.7</version>
        </extension>
    </supportedExtensions>
</options>
```

10.2. CLUE Message No. 2: 'optionsResponse'

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<optionsResponse xmlns="urn:ietf:params:xml:ns:clue-protocol"
 xmlns:ns2="urn:ietf:params:xml:ns:clue-info"
 xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
 xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 protocol="CLUE" v="1.4">
    <clueId>CP2</clueId>
    <sequenceNr>62</sequenceNr>
    <responseCode>200</responseCode>
    <reasonString>Success</reasonString>
    <mediaProvider>true</mediaProvider>
    <mediaConsumer>true</mediaConsumer>
    <version>2.7</version>
</optionsResponse>
```

10.3. CLUE Message No. 3: 'advertisement'

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:advertisement xmlns="urn:ietf:params:xml:ns:clue-info"
 xmlns:ns2="urn:ietf:params:xml:ns:clue-protocol"
 xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
 xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 protocol="CLUE" v="2.7">
    <ns2:clueId>CP1</ns2:clueId>
    <ns2:sequenceNr>11</ns2:sequenceNr>
    <ns2:mediaCaptures>
      <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="audioCaptureType" captureID="AC0"
         mediaType="audio">
         <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                   <capturePoint>
                      <x>0.0</x>
                      <y>0.0</y>
                      <z>10.0</z>
                 </capturePoint>
                 <lineOfCapturePoint>
                   <x>0.0</x>
                   <y>1.0</y>
                   <z>10.0</z>
                  </lineOfCapturePoint>
               </captureOrigin>
             </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG1</encGroupIDREF>
            <description lang="en">main audio from the room
            </description>
            <priority>1</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>room</view>
            <capturedPeople>
                <personIDREF>alice</personIDREF>
                <personIDREF>bob</personIDREF>
                <personIDREF>ciccio</personIDREF>
            </capturedPeople>
         </mediaCapture>
         <mediaCapture
          xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
          xsi:type="videoCaptureType" captureID="VC0"
          mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>-2.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                <captureArea>
                        <bottomLeft>
```

```
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                        <x>-1.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                        <x>-1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                    </captureArea>
                </spatialInformation>
                <individual>true</individual>
                <encGroupIDREF>EG0</encGroupIDREF>
                <description lang="en">left camera video capture
                </description>
                <priority>1</priority>
                <lang>it</lang>
                <mobility>static</mobility>
                <view>individual</view>
                <capturedPeople>
                    <personIDREF>ciccio</personIDREF>
                </capturedPeople>
            </mediaCapture>
            <mediaCapture
             xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
             xsi:type="videoCaptureType" captureID="VC1"
             mediaType="video">
                <captureSceneIDREF>CS1</captureSceneIDREF>
                <spatialInformation>
                    <captureOrigin>
                            <capturePoint>
                            <x>0.0</x>
                            <y>0.0</y>
                            <z>10.0</z>
                        </capturePoint>
                    </captureOrigin>
                    <captureArea>
                            <bottomLeft>
                            <x>-1.0</x>
                            <y>20.0</y>
                            <z>9.0</z>
                            </bottomLeft>
                            <bottomRight>
                            <x>1.0</x>
                            <y>20.0</y>
                            <z>9.0</z>
                            </bottomRight>
                            <topLeft>
```

```
                        <x>-1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                        <x>1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                    </captureArea>
                </spatialInformation>
                <individual>true</individual>
                <encGroupIDREF>EG0</encGroupIDREF>
                <description lang="en">central camera video capture
                </description>
                <priority>1</priority>
                <lang>it</lang>
                <mobility>static</mobility>
                <view>individual</view>
                <capturedPeople>
                    <personIDREF>alice</personIDREF>
                </capturedPeople>
            </mediaCapture>
            <mediaCapture
             xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
             xsi:type="videoCaptureType" captureID="VC2"
             mediaType="video">
                <captureSceneIDREF>CS1</captureSceneIDREF>
                <spatialInformation>
                    <captureOrigin>
                            <capturePoint>
                            <x>2.0</x>
                            <y>0.0</y>
                            <z>10.0</z>
                        </capturePoint>
                    </captureOrigin>
                    <captureArea>
                            <bottomLeft>
                            <x>1.0</x>
                            <y>20.0</y>
                            <z>9.0</z>
                            </bottomLeft>
                            <bottomRight>
                            <x>3.0</x>
                            <y>20.0</y>
                            <z>9.0</z>
                            </bottomRight>
                            <topLeft>
                            <x>1.0</x>
                            <y>20.0</y>
                            <z>11.0</z>
                            </topLeft>
                            <topRight>
                            <x>3.0</x>
                            <y>20.0</y>
                            <z>11.0</z>
                            </topRight>
                    </captureArea>
```

```
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">right camera video capture
            </description>
            <priority>1</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>individual</view>
            <capturedPeople>
                <personIDREF>bob</personIDREF>
            </capturedPeople>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC3"
         mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureArea>
                        <bottomLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <content>
                <sceneViewIDREF>SE1</sceneViewIDREF>
            </content>
            <policy>SoundLevel:0</policy>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">loudest room segment
            </description>
            <priority>2</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>individual</view>
            </mediaCapture>
        <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC4"
         mediaType="video">
```

```
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                </captureOrigin>
                <captureArea>
                        <bottomLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>7.0</z>
                        </bottomLeft>
                        <bottomRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>7.0</z>
                        </bottomRight>
                        <topLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>13.0</z>
                        </topLeft>
                        <topRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>13.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">zoomed-out view of all people in
            the room</description>
            <priority>2</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>room</view>
            <capturedPeople>
                <personIDREF>alice</personIDREF>
                <personIDREF>bob</personIDREF>
                <personIDREF>ciccio</personIDREF>
                </capturedPeople>
        </mediaCapture>
    </ns2:mediaCaptures>
    <ns2:encodingGroups>
        <encodingGroup encodingGroupID="EG0">
            <maxGroupBandwidth>600000</maxGroupBandwidth>
            <encodingIDList>
                <encodingID>ENC1</encodingID>
                <encodingID>ENC2</encodingID>
                <encodingID>ENC3</encodingID>
            </encodingIDList>
        </encodingGroup>
        <encodingGroup encodingGroupID="EG1">
            <maxGroupBandwidth>300000</maxGroupBandwidth>
```

```
            <encodingIDList>
                <encodingID>ENC4</encodingID>
                <encodingID>ENC5</encodingID>
            </encodingIDList>
        </encodingGroup>
    </ns2:encodingGroups>
    <ns2:captureScenes>
        <captureScene scale="unknown" sceneID="CS1">
            <sceneViews>
                <sceneView sceneViewID="SE1">
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC0</mediaCaptureIDREF>
                        <mediaCaptureIDREF>VC1</mediaCaptureIDREF>
                        <mediaCaptureIDREF>VC2</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE2">
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC3</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE3">
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC4</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE4">
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>AC0</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
            </sceneViews>
        </captureScene>
    </ns2:captureScenes>
    <ns2:simultaneousSets>
        <simultaneousSet setID="SS1">
            <mediaCaptureIDREF>VC3</mediaCaptureIDREF>
            <sceneViewIDREF>SE1</sceneViewIDREF>
        </simultaneousSet>
        <simultaneousSet setID="SS2">
            <mediaCaptureIDREF>VC0</mediaCaptureIDREF>
            <mediaCaptureIDREF>VC2</mediaCaptureIDREF>
            <mediaCaptureIDREF>VC4</mediaCaptureIDREF>
        </simultaneousSet>
    </ns2:simultaneousSets>
    <ns2:people>
        <person personID="bob">
            <personInfo>
                <ns3:fn>
                  <ns3:text>Bob</ns3:text>
                </ns3:fn>
            </personInfo>
            <personType>minute taker</personType>
        </person>
        <person personID="alice">
            <personInfo>
                <ns3:fn>
                    <ns3:text>Alice</ns3:text>
```

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

10.4. CLUE Message No. 4: 'configure+ack'

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:configure xmlns="urn:ietf:params:xml:ns:clue-info"
 xmlns:ns2="urn:ietf:params:xml:ns:clue-protocol"
 xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
 xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 protocol="CLUE" v="2.7">
    <ns2:clueId>CP2</ns2:clueId>
    <ns2:sequenceNr>22</ns2:sequenceNr>
    <ns2:advSequenceNr>11</ns2:advSequenceNr>
    <ns2:ack>200</ns2:ack>
    <ns2:captureEncodings>
        <captureEncoding ID="ce123">
           <captureID>AC0</captureID>
           <encodingID>ENC4</encodingID>
        </captureEncoding>
        <captureEncoding ID="ce223">
           <captureID>VC3</captureID>
           <encodingID>ENC1</encodingID>
           <configuredContent>
              <sceneViewIDREF>SE1</sceneViewIDREF>
           </configuredContent>
       </captureEncoding>
    </ns2:captureEncodings>
</ns2:configure>
```

10.5. CLUE Message No. 5: 'configureResponse'

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:configureResponse xmlns="urn:ietf:params:xml:ns:clue-info"
 xmlns:ns2="urn:ietf:params:xml:ns:clue-protocol"
 xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
 xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 protocol="CLUE" v="2.7">
    <ns2:clueId>CP1</ns2:clueId>
    <ns2:sequenceNr>12</ns2:sequenceNr>
    <ns2:responseCode>200</ns2:responseCode>
    <ns2:reasonString>Success</ns2:reasonString>
    <ns2:confSequenceNr>22</ns2:confSequenceNr>
</ns2:configureResponse>
```

10.6. CLUE Message No. 6: 'advertisement'

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:advertisement xmlns="urn:ietf:params:xml:ns:clue-info"
 xmlns:ns2="urn:ietf:params:xml:ns:clue-protocol"
 xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
 xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 protocol="CLUE" v="2.7">
    <ns2:clueId>CP1</ns2:clueId>
    <ns2:sequenceNr>13</ns2:sequenceNr>
    <ns2:mediaCaptures>
         <mediaCapture
          xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
          xsi:type="audioCaptureType" captureID="AC0"
          mediaType="audio">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                    <lineOfCapturePoint>
                        <x>0.0</x>
                        <y>1.0</y>
                        <z>10.0</z>
                    </lineOfCapturePoint>
                </captureOrigin>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG1</encGroupIDREF>
            <description lang="en">main audio from the room
            </description>
            <priority>1</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>room</view>
            <capturedPeople>
                <personIDREF>alice</personIDREF>
                <personIDREF>bob</personIDREF>
                <personIDREF>ciccio</personIDREF>
            </capturedPeople>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC0"
         mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.5</x>
                        <y>1.0</y>
                        <z>0.5</z>
                    </capturePoint>
                    <lineOfCapturePoint>
                        <x>0.5</x>
                        <y>0.0</y>
```

```
                        <z>0.5</z>
                    </lineOfCapturePoint>
                </captureOrigin>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">left camera video capture
            </description>
            <priority>1</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>individual</view>
            <capturedPeople>
                <personIDREF>ciccio</personIDREF>
            </capturedPeople>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC1"
         mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                </captureOrigin>
                <captureArea>
                        <bottomLeft>
                        <x>-1.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                        <x>1.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                        <x>-1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                        <x>1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">central camera video capture
            </description>
            <priority>1</priority>
            <lang>it</lang>
```

```
            <mobility>static</mobility>
            <view>individual</view>
            <capturedPeople>
                <personIDREF>alice</personIDREF>
            </capturedPeople>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC2"
         mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>2.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                </captureOrigin>
                <captureArea>
                        <bottomLeft>
                        <x>1.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                        <x>1.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">right camera video capture
            </description>
            <priority>1</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>individual</view>
            <capturedPeople>
                <personIDREF>bob</personIDREF>
            </capturedPeople>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC3"
         mediaType="video">
```

```
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureArea>
                        <bottomLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <content>
                <sceneViewIDREF>SE1</sceneViewIDREF>
            </content>
            <policy>SoundLevel:0</policy>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">loudest room segment
            </description>
            <priority>2</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC4"
         mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureOrigin>
                        <capturePoint>
                        <x>0.0</x>
                        <y>0.0</y>
                        <z>10.0</z>
                    </capturePoint>
                </captureOrigin>
                <captureArea>
                        <bottomLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>7.0</z>
                        </bottomLeft>
                        <bottomRight>
                        <x>3.0</x>
```

```
                        <y>20.0</y>
                        <z>7.0</z>
                        </bottomRight>
                        <topLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>13.0</z>
                        </topLeft>
                        <topRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>13.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <individual>true</individual>
            <encGroupIDREF>EG0</encGroupIDREF>
            <description lang="en">
              zoomed-out view of all people in the room
            </description>
            <priority>2</priority>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>room</view>
            <capturedPeople>
                <personIDREF>alice</personIDREF>
                <personIDREF>bob</personIDREF>
                <personIDREF>ciccio</personIDREF>
            </capturedPeople>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC5"
         mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureArea>
                        <bottomLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
```

```
            </spatialInformation>
            <content>
                <sceneViewIDREF>SE1</sceneViewIDREF>
            </content>
            <policy>SoundLevel:1</policy>
            <description lang="en">penultimate loudest room segment
            </description>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC6"
         mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureArea>
                        <bottomLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                </captureArea>
            </spatialInformation>
            <content>
                <sceneViewIDREF>SE1</sceneViewIDREF>
            </content>
            <policy>SoundLevel:2</policy>
            <description lang="en">last but two loudest room segment
            </description>
            <lang>it</lang>
            <mobility>static</mobility>
            <view>individual</view>
        </mediaCapture>
        <mediaCapture
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:type="videoCaptureType" captureID="VC7"
         mediaType="video">
            <captureSceneIDREF>CS1</captureSceneIDREF>
            <spatialInformation>
                <captureArea>
                        <bottomLeft>
```

```
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomLeft>
                        <bottomRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>9.0</z>
                        </bottomRight>
                        <topLeft>
                        <x>-3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topLeft>
                        <topRight>
                        <x>3.0</x>
                        <y>20.0</y>
                        <z>11.0</z>
                        </topRight>
                    </captureArea>
                </spatialInformation>
                <content>
                    <mediaCaptureIDREF>VC3</mediaCaptureIDREF>
                    <mediaCaptureIDREF>VC5</mediaCaptureIDREF>
                    <mediaCaptureIDREF>VC6</mediaCaptureIDREF>
                </content>
                <maxCaptures exactNumber="true">3</maxCaptures>
                <encGroupIDREF>EG0</encGroupIDREF>
                <description lang="en">big picture of the current
                speaker + pips about previous speakers</description>
                <priority>3</priority>
                <lang>it</lang>
                <mobility>static</mobility>
                <view>individual</view>
            </mediaCapture>
        </ns2:mediaCaptures>
        <ns2:encodingGroups>
            <encodingGroup encodingGroupID="EG0">
                <maxGroupBandwidth>600000</maxGroupBandwidth>
                <encodingIDList>
                    <encodingID>ENC1</encodingID>
                    <encodingID>ENC2</encodingID>
                    <encodingID>ENC3</encodingID>
                </encodingIDList>
            </encodingGroup>
            <encodingGroup encodingGroupID="EG1">
                <maxGroupBandwidth>300000</maxGroupBandwidth>
                <encodingIDList>
                    <encodingID>ENC4</encodingID>
                    <encodingID>ENC5</encodingID>
                </encodingIDList>
            </encodingGroup>
        </ns2:encodingGroups>
        <ns2:captureScenes>
            <captureScene scale="unknown" sceneID="CS1">
                <sceneViews>
                    <sceneView sceneViewID="SE1">
                        <description lang="en">participants' individual
```

```
                    videos</description>
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC0</mediaCaptureIDREF>
                        <mediaCaptureIDREF>VC1</mediaCaptureIDREF>
                        <mediaCaptureIDREF>VC2</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE2">
                    <description lang="en">loudest segment of the
                    room</description>
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC3</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE5">
                    <description lang="en">loudest segment of the
                    room + pips</description>
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC7</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE4">
                    <description lang="en">room audio</description>
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>AC0</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
                <sceneView sceneViewID="SE3">
                    <description lang="en">room video</description>
                    <mediaCaptureIDs>
                        <mediaCaptureIDREF>VC4</mediaCaptureIDREF>
                    </mediaCaptureIDs>
                </sceneView>
            </sceneViews>
        </captureScene>
    </ns2:captureScenes>
    <ns2:simultaneousSets>
        <simultaneousSet setID="SS1">
            <mediaCaptureIDREF>VC3</mediaCaptureIDREF>
            <mediaCaptureIDREF>VC7</mediaCaptureIDREF>
            <sceneViewIDREF>SE1</sceneViewIDREF>
        </simultaneousSet>
        <simultaneousSet setID="SS2">
            <mediaCaptureIDREF>VC0</mediaCaptureIDREF>
            <mediaCaptureIDREF>VC2</mediaCaptureIDREF>
            <mediaCaptureIDREF>VC4</mediaCaptureIDREF>
        </simultaneousSet>
    </ns2:simultaneousSets>
    <ns2:people>
        <person personID="bob">
            <personInfo>
                <ns3:fn>
                    <ns3:text>Bob</ns3:text>
                </ns3:fn>
            </personInfo>
            <personType>minute taker</personType>
        </person>
        <person personID="alice">
```

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

10.7. CLUE Message No. 7: 'ack'

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ack xmlns="urn:ietf:params:xml:ns:clue-protocol"
 xmlns:ns2="urn:ietf:params:xml:ns:clue-info"
 xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
 xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 protocol="CLUE" v="2.7">
    <clueId>CP2</clueId>
    <sequenceNr>23</sequenceNr>
    <responseCode>200</responseCode>
    <reasonString>Success</reasonString>
    <advSequenceNr>13</advSequenceNr>
</ack>
```

10.8. CLUE Message No. 8: 'configure'

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:configure xmlns="urn:ietf:params:xml:ns:clue-info"
 xmlns:ns2="urn:ietf:params:xml:ns:clue-protocol"
 xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
 xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 protocol="CLUE" v="2.7">
    <ns2:clueId>CP2</ns2:clueId>
    <ns2:sequenceNr>24</ns2:sequenceNr>
    <ns2:advSequenceNr>13</ns2:advSequenceNr>
    <ns2:captureEncodings>
               <captureEncoding ID="ce123">
                        <captureID>AC0</captureID>
                        <encodingID>ENC4</encodingID>
                </captureEncoding>
                <captureEncoding ID="ce456">
                        <captureID>VC7</captureID>
                        <encodingID>ENC1</encodingID>
                        <configuredContent>
                                <sceneViewIDREF>SE5</sceneViewIDREF>
                        </configuredContent>
                </captureEncoding>
     </ns2:captureEncodings>
</ns2:configure>
```

10.9. CLUE Message No. 9: 'configureResponse'

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:configureResponse xmlns="urn:ietf:params:xml:ns:clue-info"
 xmlns:ns2="urn:ietf:params:xml:ns:clue-protocol"
 xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
 xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 protocol="CLUE" v="2.7">
    <ns2:clueId>CP1</ns2:clueId>
    <ns2:sequenceNr>14</ns2:sequenceNr>
    <ns2:responseCode>200</ns2:responseCode>
    <ns2:reasonString>Success</ns2:reasonString>
    <ns2:confSequenceNr>24</ns2:confSequenceNr>
</ns2:configureResponse>
```

11. Security Considerations

As a general consideration, we would like to point out that the CLUE framework (and related protocol) has been conceived from the outset by embracing the security-by-design paradigm. As a result, a number of requirements have been identified and properly standardized as mandatory within the entire set of documents associated with the CLUE architecture. Requirements include (i) the use of cryptography and authentication, (ii) protection of all sensitive fields, (iii) mutual authentication between CLUE endpoints, (iv) the presence of authorization mechanisms, and (v)

the presence of native defense mechanisms against malicious activities such as eavesdropping, selective modification, deletion, and replay (and related combinations thereof). Hence, security of the single components of the CLUE solution cannot be evaluated independently of the integrated view of the final architecture.

The CLUE protocol is an application-level protocol allowing a Media Producer and an MC to negotiate a variegated set of parameters associated with the establishment of a telepresence session. This unavoidably exposes a CLUE-enabled telepresence system to a number of potential threats, most of which are extensively discussed in the CLUE framework document [RFC8845]. The Security Considerations section of [RFC8845] actually discusses issues associated with the setup and management of a telepresence session in both (1) the basic case involving two CLUE endpoints acting as the MP and the MC, respectively and (2) the more advanced scenario envisaging the presence of an MCU.

The CLUE framework document [RFC8845] also mentions that the information carried within CLUE protocol messages might contain sensitive data, which **SHOULD** hence be accessed only by authenticated endpoints. Security issues associated with the CLUE data model schema are discussed in [RFC8846].

There is extra information carried by the CLUE protocol that is not associated with the CLUE data model schema and that exposes information that might be of concern. This information is primarily exchanged during the negotiation phase via the 'options' and 'optionsResponse' messages. In the CP state machine's OPTIONS state, both parties agree on the version and extensions to be used in the subsequent CLUE message exchange phase. A malicious participant might either (1) try to retrieve a detailed footprint of a specific CLUE protocol implementation during this initial setup phase or (2) force the communicating party to use a version of the protocol that is outdated and that they know how to break. Indeed, exposing all of the supported versions and extensions could conceivably leak information about the specific implementation of the protocol. In theory, an implementation could choose not to announce all of the versions it supports if it wants to avoid such leakage, although this would come at the expense of interoperability. With respect to the above considerations, it is noted that the OPTIONS state is only reached after the CLUE data channel has been successfully set up. This ensures that only authenticated parties can exchange 'options' messages and related 'optionsResponse' messages, and hence drastically reduces the attack surface that is exposed to malicious parties.

The CLUE framework clearly states the requirement to protect CLUE protocol messages against threats deriving from the presence of a malicious agent capable of gaining access to the CLUE data channel. Such a requirement is met by the CLUE data channel solution described in [RFC8850], which ensures protection from both message recovery and message tampering. With respect to this last point, any implementation of the CLUE protocol compliant with the CLUE specification **MUST** rely on the exchange of messages that flow on top of a reliable and ordered SCTP-over-DTLS transport channel connecting two CPs.

12. IANA Considerations

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

12.1. URN Sub-Namespace Registration

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

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

Registrant Contact: IESG (iesg@ietf.org).

XML:

```
<CODE BEGINS>
 <?xml version="1.0"?>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "https://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <html xmlns="https://www.w3.org/1999/xhtml" xml:lang="en">
   <head>
     <title>CLUE Messages</title>
   </head>
   <body>
     <h1>Namespace for CLUE Messages</h1>
     <h2>urn:ietf:params:xml:ns:clue-protocol</h2>
     <p>See <a href="https://www.rfc-editor.org/rfc/rfc8847.txt">
        RFC 8847</a>.</p>
   </body>
 </html>

<CODE ENDS>
```

12.2. XML Schema Registration

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

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

Registrant Contact: IESG (iesg@ietf.org).

Schema: The XML for this schema can be found in Section 9 of this document.

12.3. Media Type Registration for "application/clue+xml"

This section registers the `application/clue+xml` media type.

To: ietf-types@iana.org

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

Media type name: application

Subtype name: clue+xml

Required parameters: (none)

Optional parameters: charset. Same as the charset parameter of "application/xml" as specified in [RFC7303], Section 4.2.

Encoding considerations: Same as the encoding considerations of "application/xml" as specified in [RFC7303], Section 4.2.

Security considerations: This content type is designed to carry protocol data related to telepresence session control. Some of the data could be considered private. This media type does not provide any protection; thus, other mechanisms, such as those described in Section 11 of this document, are required to protect the data. This media type does not contain executable content.

Interoperability considerations: None.

Published specification: RFC 8847

Applications that use this media type: CLUE Participants.

Additional Information:

Magic Number(s): (none)
File extension(s): .xml
Macintosh File Type Code(s): TEXT

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

Intended usage: LIMITED USE

Author/Change controller: The IETF

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

12.4. CLUE Protocol Registry

Per this document, IANA has created new registries for CLUE messages and response codes.

12.4.1. CLUE Message Types

The following summarizes the registry for CLUE messages:

Related Registry: CLUE Message Types

Defining RFC: RFC 8847

Registration/Assignment Procedures: Following the policies outlined in [RFC8126], the IANA policy for assigning new values for the CLUE message types for the CLUE protocol is Specification Required.

Registrant Contact: IESG (iesg@ietf.org).

The initial table of CLUE messages is populated using the CLUE messages described in Section 5 and defined in the XML schema in Section 9.

Message	Description	Reference
options	Sent by the CI to the CR in the initiation phase to specify the roles played by the CI, the supported versions, and the supported extensions.	RFC 8847
optionsResponse	Sent by the CI to the CR in reply to an 'options' message, to establish the version and extensions to be used in the subsequent exchange of CLUE messages.	RFC 8847
advertisement	Sent by the MP to the MC to specify the telepresence capabilities of the MP expressed according to the CLUE framework.	RFC 8847
ack	Sent by the MC to the MP to acknowledge the reception of an 'advertisement' message.	RFC 8847
configure	Sent by the MC to the MP to specify the desired media captures among those specified in the 'advertisement'.	RFC 8847
configureResponse	Sent by the MP to the MC in reply to a 'configure' message to communicate whether or not the configuration request has been successfully processed.	RFC 8847

Table 2: Initial IANA Table of CLUE Messages

12.4.2. CLUE Response Codes

The following summarizes the registry for CLUE response codes:

Related Registry: CLUE Response Codes

Defining RFC: RFC 8847

Registration/Assignment Procedures: Following the policies outlined in [RFC8126], the IANA policy for assigning new values for the response codes for CLUE is Specification Required.

Registrant Contact: IESG (iesg@ietf.org).

The initial table of CLUE response codes is populated using the response codes defined in Section 5.7 as follows:

Number	Default Reason String	Description	Reference
200	Success	The request has been successfully processed.	RFC 8847
300	Low-level request error	A generic low-level request error has occurred.	RFC 8847
301	Bad syntax	The XML syntax of the message is not correct.	RFC 8847
302	Invalid value	The message contains an invalid parameter value.	RFC 8847
303	Conflicting values	The message contains values that cannot be used together.	RFC 8847
400	Semantic errors	The received CLUE protocol message contains semantic errors.	RFC 8847
401	Version not supported	The protocol version used in the message is not supported.	RFC 8847
402	Invalid sequencing	The received message contains an unexpected sequence number (e.g., sequence number gap, repeated sequence number, or sequence number outdated).	RFC 8847
403	Invalid identifier	The clueId used in the message is invalid or unknown.	RFC 8847
404	Advertisement expired	The sequence number of the advertisement the 'configure' message refers to is out of date.	RFC 8847
405	Subset choice not allowed	The subset choice is not allowed for the specified Multiple Content Capture.	RFC 8847

Table 3: Initial IANA Table of CLUE Response Codes

13. References

13.1. Normative References

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

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

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

[RFC7303] Thompson, H. and C. Lilley, "XML Media Types", RFC 7303, DOI 10.17487/RFC7303, July 2014, <https://www.rfc-editor.org/info/rfc7303>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8845] Duckworth, M., Ed., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", RFC 8845, DOI 10.17487/RFC8845, August 2020, <https://www.rfc-editor.org/info/rfc8845>.

[RFC8846] Presta, R. and S P. Romano, "An XML Schema for the Controlling Multiple Streams for Telepresence (CLUE) Data Model", RFC 8846, DOI 10.17487/RFC8846, August 2020, <https://www.rfc-editor.org/info/rfc8846>.

[RFC8848] Hanton, R., Kyzivat, P., Xiao, L., and C. Groves, "Session Signaling for Controlling Multiple Streams for Telepresence (CLUE)", RFC 8848, DOI 10.17487/RFC8848, August 2020, <https://www.rfc-editor.org/info/rfc8848>.

[RFC8850] Holmberg, C., "Controlling Multiple Streams for Telepresence (CLUE) Protocol Data Channel", RFC 8850, DOI 10.17487/RFC8850, August 2020, <https://www.rfc-editor.org/info/rfc8850>.

[W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <https://www.w3.org/TR/2008/REC-xml-20081126>.

13.2. Informative References

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

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

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/rfc6120>.

[RFC7262] Romanow, A., Botzko, S., and M. Barnes, "Requirements for Telepresence Multistreams", RFC 7262, DOI 10.17487/RFC7262, June 2014, <https://www.rfc-editor.org/info/rfc7262>.

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

Acknowledgements

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

Authors' Addresses

Roberta Presta
University of Napoli
Via Claudio 21
80125 Napoli
Italy
Email: roberta.presta@unina.it

Simon Pietro Romano
University of Napoli
Via Claudio 21
80125 Napoli
Italy
Email: spromano@unina.it