CLUE Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

R. Presta
S. Romano
University of Napoli
October 19, 2015

CLUE protocol
draft-ietf-clue-protocol-06

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 21, 2016.

Copyright Notice

Table of Contents

1. Introduction

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

1. enabling a 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 envisioned in [I-D.ietf-clue-framework];

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 which is opened and established as described in [I-D.ietf-clue-signaling] and [I-D.ietf-clue-datachannel]. CLUE protocol messages flowing upon such a channel are detailed in this document, both syntactically and semantically.

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

2. Terminology

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

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

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

Endpoint: The logical point of final termination through receiving, decoding and rendering, and/or initiation through capturing, encoding, and sending of media streams. An endpoint consists of one or more physical devices which source and sink media streams,

and exactly one [RFC4353] Participant (which, in turn, includes exactly one SIP User Agent). Endpoints can be anything from multiscreen/multicamera room controllers to handheld devices.

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

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

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

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

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

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

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

3. Overview of the CLUE protocol

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

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

can be identified:

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

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

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

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

After that negotiation phase is completed, CLUE Participants describe and agree on the media flows to be exchanged. In many cases CPs will seek to both transmit and receive media. Hence in a call between two CPs, A and B, there would be two seperate dialogs, 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 Media Provider side and B's Media Consumer side

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

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

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

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

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

4. Protocol messages

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

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

The CLUE protocol messages are the following:

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

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

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

```
<!-- CLUE MESSAGE TYPE -->
<xs:complexType name="clueMessageType" abstract="true">
<xs:sequence>
<xs:element name="clueId" type="xs:string"/>
<xs:element name="sequenceNr" type="xs: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="([0-9])+\.([0-9]+)"></xs:pattern>
</xs:restriction>
</xs:simpleType>
```

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

- clueId: an 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 numbers by one for each new message sent, the receiver must remember the most recent

sequence number received and send back a 401 error if it receives a message with an unexpected sequence number;

- protocol: a mandatory attribute set to "CLUE", identifying the procotol the messages refer to;

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

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

4.1. OPTIONS

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

- mediaProvider: a mandatory boolean field set to "true" if the CP is able to act as a MP

- mediaConsumer: a mandatory boolean field set to "true" if the CP is able to act as a MC

- supportedVersions: the list of the supported versions

- supportedOptions: the list of the supported options

The XML Schema of such a message is reported below:

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

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

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

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

<supportedVersions> contains the list of the versions that are supported by the CI, each one represented in a child <version> element. The content of each <version> element is a string made by the major version number followed by a dot and then by the minor version number (e.g., 1.3 or 2.43). Only one <version> element 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> tag in the OPTIONS message, it means the CI supports only major version 3 with all the minor versions comprised between 3.0 and 3.4, with version 3.4 included. If a <supportedVersion> is provided, at least one <version> tag MUST be included.

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

4.2. OPTIONS RESPONSE

The OPTIONS RESPONSE is sent by a CR to a CI as a reply to the OPTIONS message. As depicted in the figure below, the OPTIONS RESPONSE contains mandatorily a response code and a reason string indicating the processing result of the OPTIONS message. If the responseCode is of the type 2xx the response MUST also include <mediaProvider>, <mediaConsumer>, <version> and <commonOptions> elements; it MAY include them for any other response code. <mediaProvider> and <mediaConsumer> elements are associated with the supported roles (in terms of, respectively MP and MC), similarly to what the CI does in the OPTIONS message. The <version> field 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 2.43). Finally, the commonly supported options are copied in the the <commonOptions> field.

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

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

4.3. ADVERTISEMENT

The ADVERTISEMENT message (ADV) is used by the MP to advertise the available media captures and related information to the MC. The MP sends to the MC an ADV as soon as it is ready after the successful completion of the initiation phase, i.e., as soon as the version and the options of the CLUE protocol are agreed between the CPs. During a single CLUE session, an MP may send new ADV messages to replace the previously advertised options, if, for instance, its media CLUE telepresence capabilities change mid-call. A new ADV completely invalidates the previous ADV.

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

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

4.4. ADVERTISEMENT ACKNOWLEDGEMENT

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

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

4.5. CONFIGURE

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

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

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

The optional boolean <ack> element, set to "true", if present, indicates that the CONF message also acknowledges the referred advertisement, by applying in that way a piggybacking mechanism for simultaneously acknowledging and replying to the ADV message. In the above case, the CONF message MUST carry a 200 (Success) response code. The <ack> element MUST not be present if an ACK message has been already sent back to the MP.

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

4.6. CONFIGURE RESPONSE

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

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

4.7. Response codes and reason strings

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

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

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

Response codes are defined as a sequence of three digits. A well-defined meaning is associated with the first digit. Response codes beginning with "2" are associated with successful responses. Response codes beginning with "1" will represent a delayed or incomplete response. 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.). Further response codes can be designed in future versions of the protocol, provided they do not overwrite the ones here defined and they respect the semantics of the first code digit.

5. Protocol state machines

The CLUE protocol is an application protocol used between two CPs in order to properly configure a multimedia telepresence session. CLUE protocol messages flow upon the CLUE Data Channel, a DTLS/SCTP channel established as depicted in [I-D.ietf-clue-datachannel]. We herein discuss the state machines associated, respectively, with the CLUE Participant, with the MC process and with the MP process. Endpoints often wish to both send a receive media, i.e., act as both MP and 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 treated in [I-D.ietf-clue-signaling].

6. CLUE Participant's state machine

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

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

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

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

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

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

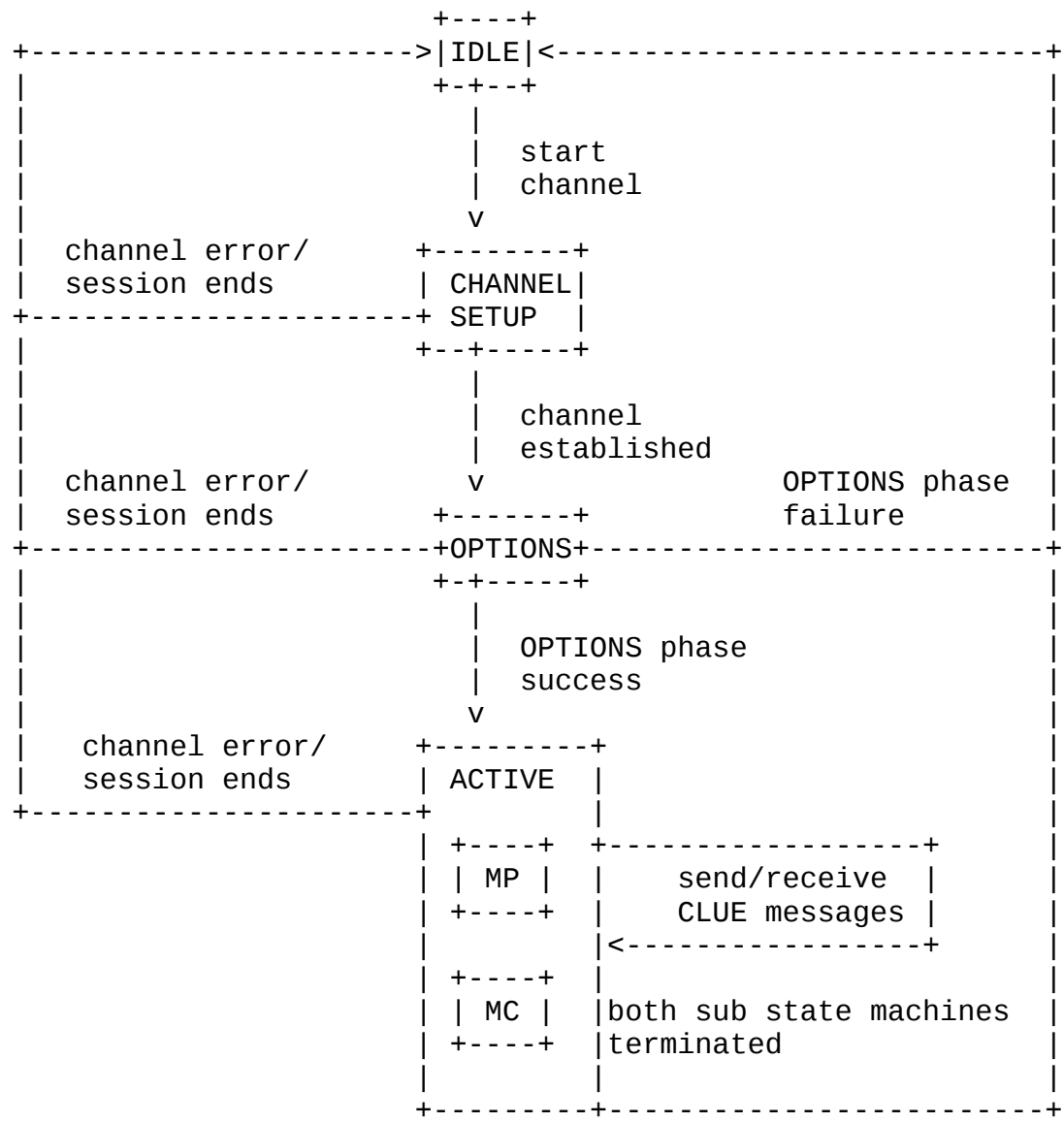
1. the CI receives an OPTIONS RESPONSE with an error response code

2. the CI does not receive any OPTIONS RESPONSE and a timeout error is raised

3. the CR does not receive any OPTIONS and a timeout error is raised

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

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



6.1. Media Provider's state machine

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

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

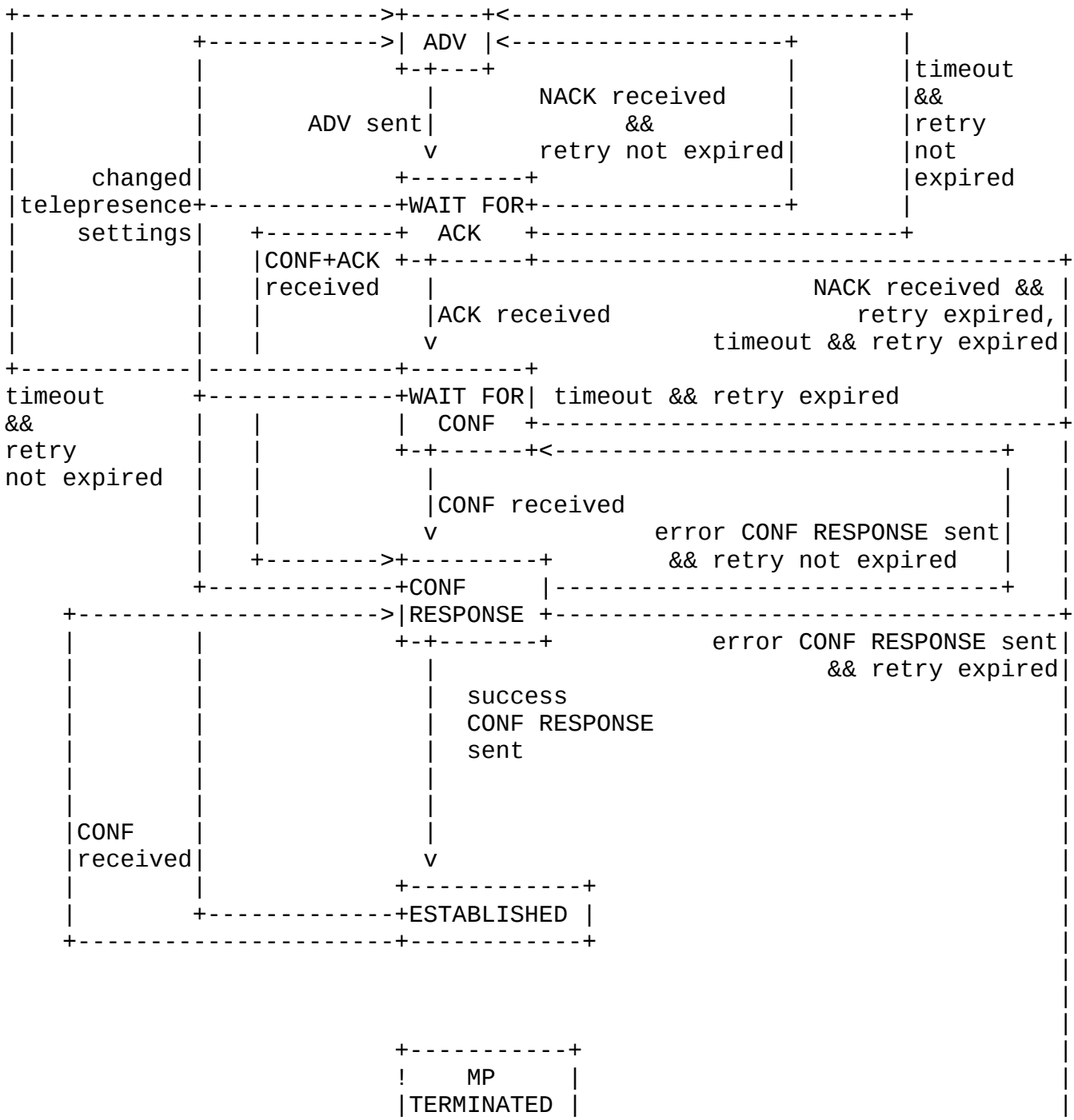
response code), and the number of NACKs for the issued ADV is under the retry threshold ("NACK received && retry not expired"), the MP moves back to the ADV state for preparing a new ADV. If a NACK arrives and the number of received NACKs for that ADV overcomes the threshold ("NACK received and && retry expired"), the MP goes to the MP-TERMINATED state. The same happens if the waiting time for the ACK is fired a number of times under the retry threshold ("timeout && retry not expired"): also in this case, the MP goes back to the ADV state to send a new copy of the ADV. If the number of retries overcomes the threshold ("timeout && retry expired"), the MP moves from the WAIT FOR ACK state to the MP-TERMINATED state. When in the WAIT FOR ACK state, if a CONFIGURE message with the <ack> element set to TRUE arrives ("CONF+ACK received"), the MP goes directly to the CONF RESPONSE state. CONF+ACK messages referring to out-of-date (i.e., having a sequence number equal to or less than the highest seen so far) ADVs MUST be ignored, i.e., they do not trigger any state transition. If the telepresence settings of the MP change while in the WAIT FOR ACK state ("changed telepresence settings"), the MP switches from the WAIT FOR ACK state to the ADV state to create a new ADV.

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

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

The MP in the ESTABLISHED state has successfully negotiated the media streams with the MC by means of the CLUE messages. If there are

changes in the MP's telepresence settings ("changed telepresence settings"), the MP moves back to the ADV state. In the ESTABLISHED state, the CLUE-controlled media streams of the session are those described in the last successfully processed CONF message.



+----------+<--------------------------------+

6.2. Media Consumer's state machine

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

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

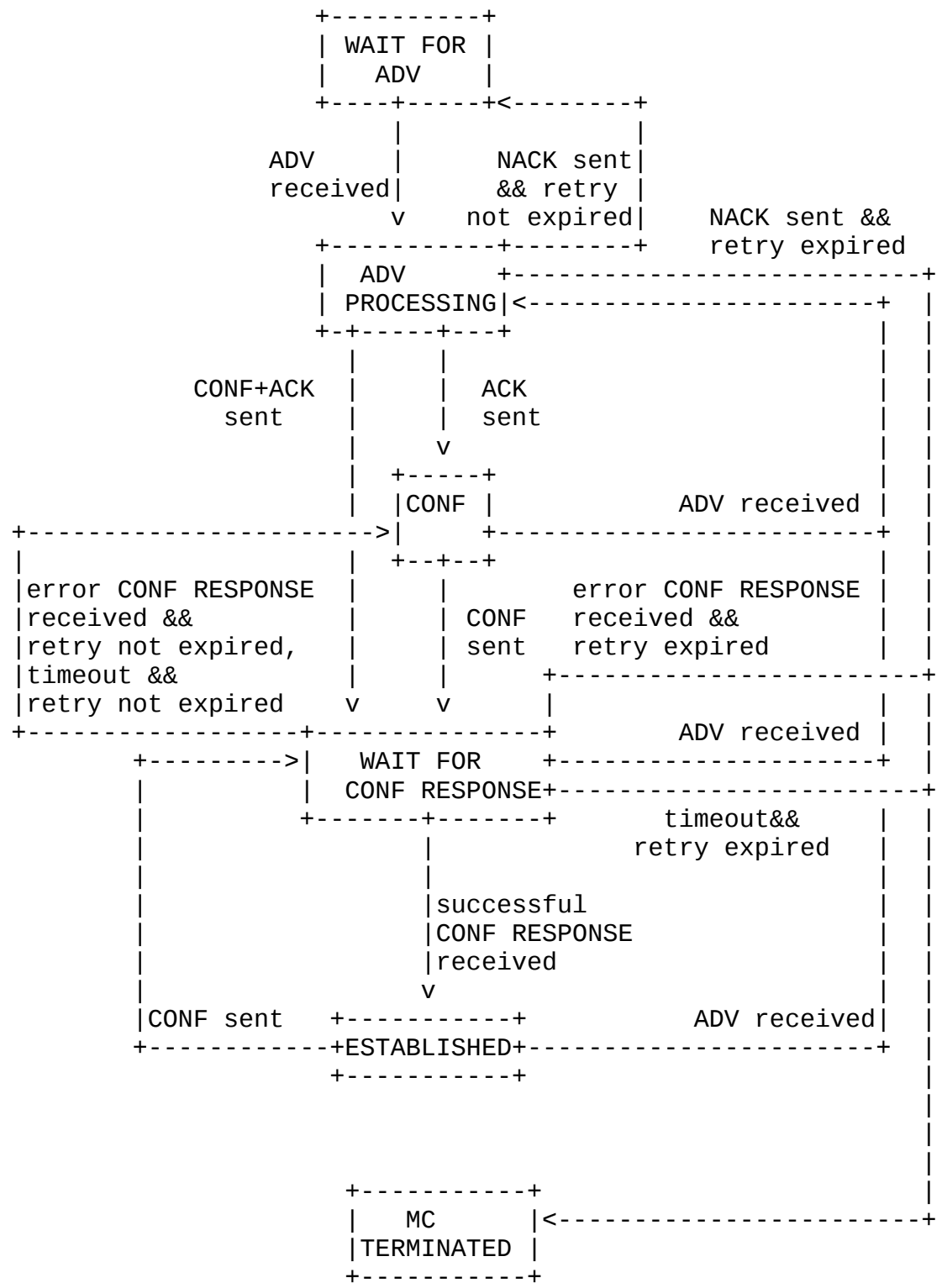
If the ADV elaboration is unsuccessful (bad syntax, missing XML elements, etc.), and the number of times this has happened is under the retry treshold, the MC sends a NACK message (i.e., an ACK with an error response code) to the MP describing the problem via a proper reason phrase. By this way ("NACK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

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

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

and the number of timeouts is over the threshold ("timeout && retry expired"), the MC moves to the MC TERMINATED state. If a new ADV is received in the WAIT FOR CONF RESPONSE state, the MC switches to the ADV PROCESSING state.

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


WAIT FOR
ADV
ADV
received
NACK sent
&& retry
not expired
NACK sent &&
retry expired
ADV
PROCESSING
CONF+ACK
sent
ACK
sent
CONF
ADV received
error CONF RESPONSE
received &&
retry not expired,
timeout &&
retry not expired
CONF
sent
error CONF RESPONSE
received &&
retry expired
ADV received
WAIT FOR
CONF RESPONSE
timeout&&
retry expired
successful
CONF RESPONSE
received
CONF sent
ADV received
ESTABLISHED
MC
TERMINATED

7. Versioning

CLUE protocol messages are XML messages compliant to the CLUE protocol XML schema [I-D.ietf-clue-data-model-schema]. The version of the protocol corresponds to the version of the schema. Both client and server have to test the compliance of the received messages with the XML schema of the CLUE protocol. If the compliance is not verified, the message cannot be processed further.

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

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

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

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

8. Extensions and options

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

1. new information, to be carried in the existing messages. For example, we may want to add more fields within an existing message;

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

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

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

- (ii) other XML elements compliant to the CLUE data model schema (data model information)

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

- elementFormDefault="qualified"

- attributeFormDefault="unqualified"

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

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

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

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

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

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

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

9. XML Schema

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

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
version="0.6"
targetNamespace="urn:ietf:params:xml:ns:clue-protocol"
xmlns:tns="urn:ietf:params:xml:ns:clue-protocol"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

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

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

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

<!-- CLUE MESSAGE TYPE -->
<xs:complexType name="clueMessageType" abstract="true">
<xs:sequence>
<xs:element name="clueId" type="xs:string"/>
<xs:element name="sequenceNr" type="xs: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="([0-9])+\.([0-9]+)"></xs:pattern>
</xs:restriction>
</xs:simpleType>


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


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

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

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

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

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

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

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

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

<!-- ACK MESSAGE TYPE -->
<xs:complexType name="advAcknowledgementMessageType">
<xs:complexContent>
<xs:extension base="clueMessageType">
<xs:sequence>
<xs:element name="responseCode" type="responseCodeType"/>
<xs:element name="reasonString" type="xs:string"/>
<xs:element name="advSequenceNr" type="xs: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>
<xs:element name="advSequenceNr" type="xs:positiveInteger"/>
<xs:element name="ack" type="xs:boolean"
 minOccurs="0" fixed="true"/>
<xs:element name="captureEncodings" type="dm:captureEncodingsType"
 minOccurs="0"/>
<xs:any namespace="##other" processContents="lax" minOccurs="0"/>
</xs:sequence>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- CONFIGURE RESPONSE MESSAGE TYPE -->
<xs:complexType name="configureResponseMessageType">
<xs:complexContent>
<xs:extension base="clueMessageType">
<xs:sequence>
<xs:element name="responseCode" type="responseCodeType"/>
<xs:element name="reasonString" type="xs:string"/>
<xs:element name="confSequenceNr" type="xs: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>
```

10. Examples

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

10.1. Simple ADV

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

file".

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

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

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

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

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

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

10.2. ADV with MCCs

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

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

```
            <ns2:priority>1</ns2:priority>
            <ns2:lang>it</ns2:lang>
            <ns2:mobility>static</ns2:mobility>
            <ns2:view>room</ns2:view>
            <ns2:capturedPeople>
                <ns2:personIDREF>alice</ns2:personIDREF>
                <ns2:personIDREF>bob</ns2:personIDREF>
                <ns2:personIDREF>ciccio</ns2:personIDREF>
            </ns2:capturedPeople>
        </ns2:mediaCapture>
        <ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ns2:videoCaptureType" captureID="VC0" mediaType="video" >
            <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
            <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
            <ns2:spatialInformation>
                <ns2:capturePoint>
                    <ns2:x>0.5</ns2:x>
                    <ns2:y>1.0</ns2:y>
                    <ns2:z>0.5</ns2:z>
                    <ns2:lineOfCapturePoint>
                        <ns2:x>0.5</ns2:x>
                        <ns2:y>0.0</ns2:y>
                        <ns2:z>0.5</ns2:z>
                    </ns2:lineOfCapturePoint>
                </ns2:capturePoint>
            </ns2:spatialInformation>
            <ns2:individual>true</ns2:individual>
            <ns2:description lang="en">left camera video capture</ns2:descripti
            <ns2:priority>1</ns2:priority>
            <ns2:lang>it</ns2:lang>
            <ns2:mobility>static</ns2:mobility>
            <ns2:view>individual</ns2:view>
            <ns2:capturedPeople>
                <ns2:personIDREF>ciccio</ns2:personIDREF>
            </ns2:capturedPeople>
        </ns2:mediaCapture>
        <ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="ns2:videoCaptureType" captureID="VC1" mediaType="video">
            <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
            <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
            <ns2:spatialInformation>
                <ns2:capturePoint>
                    <ns2:x>0.5</ns2:x>
                    <ns2:y>1.0</ns2:y>
                    <ns2:z>0.5</ns2:z>
                    <ns2:lineOfCapturePoint>
                        <ns2:x>0.5</ns2:x>
                        <ns2:y>0.0</ns2:y>
```

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

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

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

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

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

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

11. IANA Considerations

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

11.1. URN Sub-Namespace Registration

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

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

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

XML:

BEGIN

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

END

11.2. XML Schema registration

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

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

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

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

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

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

To: ietf-types@iana.org

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

MIME media type name: application

MIME subtype name: clue+xml

Required parameters: (none)

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

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

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

Interoperability considerations: None.

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

Applications that use this media type: CLUE participants.

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

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

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

Intended usage: LIMITED USE

Author/Change controller: The IETF

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

11.4. DNS Registrations

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

11.4.1. Application Service tag

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

Application Service Tag: CLUE

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

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

Contact information: The authors of this document

Author/Change controller: The IESG

11.4.2. Application Protocol tag

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

Application Service Tag: CLUE

Intended Usage: Identifies the CLUE Protocol.

Applicable Service Tag(s): CLUE

Terminal NAPTR Record Type(s): U

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

Contact Information: The authors of this document

Author/Change Controller: The IESG

11.5. CLUE Protocol Registry

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

11.5.1. CLUE Message Types

The following summarizes the registry for CLUE messages:

Related Registry: CLUE Message Types Registry

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

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

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

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

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

11.5.2. CLUE Response Codes

The following summarizes the requested registry for CLUE response codes:

Related Registry: CLUE Response Code Registry

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

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

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

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

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

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

12. Diff with draft-ietf-clue-protocol-05

- This document corrects some versioning errors of draft-ietf-clue-protocol-05.

13. Diff with draft-ietf-clue-protocol-04

- The document has been revised based on feedback recevied on the ML. No major modification is included in this version.

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

- Response codes section updated.
- maxCaptureEncodings removed from examples, allowSubsetChoice added.
- State machines descriptions aligned with pictures.

- Applied recommended updates indicated in Christian's review (2015-03-19).

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

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

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

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

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

1. The XML schema of the ADVERTISEMENT and of the READV have been aligned with the current definitions in [I-D.ietf-clue-data-model-schema] (example of updates: <participants> --> <people>, <globalCaptureEntries> --> <globalSceneEntries>)
2. Text has been added to clarify that, in the OPTIONS RESPONSE, when the response code is not an error response code, both <mediaProvider> and <mediaConsumer> are mandatory.

3. The content of the "v" attribute and of the <version> elements carried in the OPTIONS and OPTIONS RESPONSE messages has been described more precisely.

4. Advertisement examples have been added.

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

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

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

1. The XML Schema has been deeply revised and completed.

2. The descriptions of the CLUE messages have been added.

3. The distinction between major version numbers and minor version numbers has been cut and pasted from [I-D.ietf-clue-signaling].

4. Besides the two way one, a three way mechanism for the options negotiation has been proposed and provided to foster discussion.

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

1. "Terminology" section added.

2. Introduced the concept of "CLUE Participant" - an Endpoint or a MCU able to use the CLUE protocol within a telepresence session. A CLUE Participant can act as a Media Provider and/or as a Media Consumer.

3. Introduced the ACK/NACK mechanism for the ADVERTISEMENT.

4. MP and MC state machines have been updated. The CP state machine has been added.

21. Acknowledgments

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

22. References

22.1. Normative References

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

[I-D.ietf-clue-datachannel] Holmberg, C., "CLUE Protocol data channel", draft-ietf-clue-datachannel-10 (work in progress), September 2015.

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

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

[RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, DOI 10.17487/RFC3023, January 2001, <http://www.rfc-editor.org/info/rfc3023>.

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

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

[RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service

(DDDS)", RFC 3958, DOI 10.17487/RFC3958, January 2005, <http://www.rfc-editor.org/info/rfc3958>.

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

22.2. Informative References

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

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, DOI 10.17487/RFC5117, January 2008, <http://www.rfc-editor.org/info/rfc5117>.

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

[RFC6502] Camarillo, G., Srinivasan, S., Even, R., and J. Urpalainen, "Conference Event Package Data Format Extension for Centralized Conferencing (XCON)", RFC 6502, DOI 10.17487/RFC6502, March 2012, <http://www.rfc-editor.org/info/rfc6502>.

[RFC6503] Barnes, M., Boulton, C., Romano, S., and H. Schulzrinne, "Centralized Conferencing Manipulation Protocol", RFC 6503, DOI 10.17487/RFC6503, March 2012, <http://www.rfc-editor.org/info/

rfc6503>.

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

Authors' Addresses

Roberta Presta
University of Napoli
Via Claudio 21
Napoli 80125
Italy

EMail: roberta.presta@unina.it

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

EMail: spromano@unina.it