Internet Engineering Task Force (IETF)
Request for Comments: 6504
Category: Informational
ISSN: 2070-1721

M. Barnes
Polycom
L. Miniero
Meetecho
R. Presta
S P. Romano
University of Napoli

March 2012

Centralized Conferencing Manipulation Protocol (CCMP) Call Flow Examples

Abstract

This document provides detailed call flows for the scenarios documented in the Framework for Centralized Conferencing (XCON) (RFC 5239) and in the XCON scenarios (RFC 4597). The call flows document the use of the interface between a conference control client and a conference control server using the Centralized Conferencing Manipulation Protocol (CCMP) (RFC 6503). The objective is to provide detailed examples for reference by both protocol researchers and developers.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

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 a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6504.

Table of Contents

1. Introduction

This document provides detailed call flows for the scenarios documented in the Framework for Centralized Conferencing (XCON) [RFC5239] and in the XCON scenarios [RFC4597]. The XCON scenarios describe a broad range of use cases taking advantage of the advanced conferencing capabilities provided by a system realization of the XCON framework. The call flows document the use of the interface between a conference control client and a conference control server using the Centralized Conferencing Manipulation Protocol (CCMP) [RFC6503].

Due to the broad range of functionality provided by the XCON framework and the flexibility of the CCMP messaging, these call flows should not be considered inclusive of all the functionality that can provided by the XCON framework and protocol implementations. These flows represent a sample to provide an overview of the feature-rich capabilities of the XCON framework and CCMP messaging for protocol developers, software developers, and researchers.

2. Terminology

This document uses the same terminology as found in the Architectural Framework for Media Server Control [RFC5567] and in the Media Control Channel Framework Call Flow Examples [CALL-FLOWS], with the following terms and abbreviations used in the call flows. Also, note that the term "call flows" is used in a very generic sense in this document since the media is not limited to voice. The calls supported by the XCON framework and CCMP can consist of media such as text, voice, and video, including multiple media types in a single active conference.

Conference and Media Control Client (CMCC): as defined in the XCON framework. In the flows in this document, the CMCC is logically equivalent to the use of a User Agent Client (UAC) as the client notation in the media control call flows [CALL-FLOWS]. A CMCC differs from a generic media client in being an XCON-aware entity, thus, also being able to issue CCMP requests.

Conference Server (ConfS): In this document, the term "conference server" is used interchangeably with the term "Application Server (AS)" as used in the media control architectural framework [RFC5567]. A conference server is intended to be able to act as a conference control server, as defined in the XCON framework, i.e., it is able to handle CCMP requests and issue CCMP responses.

Media Server (MS): as defined in the media control architectural framework [RFC5567].

3. Overview

This document provides a sampling of detailed call flows that can be implemented based on a system realization of the XCON framework [RFC5239] and implementation of CCMP [RFC6503]. This is intended to be a simple guide for the use of the conference control protocol between the conference server and the conference control client. The objective is to provide an informational base reference for protocol developers, software developers, and researchers.

This document focuses on the interaction between the conference and media control client and the conferencing system, specifically the conference server. The scenarios are based on those described in the XCON framework, many of which are based on the advanced conferencing capabilities described in the XCON scenarios. Additional scenarios have been added to provide examples of other real-life scenarios that are anticipated to be supported by the framework. With the exception of an initial example with media control messaging, the examples do not include the details for the media control [RFC6505], call signaling, or Binary Floor Control Protocols (BFCPs) [RFC4582]. This document references the scenarios in the media control call flows [CALL-FLOWS], SIP call control conferencing, [RFC4579], and BFCP documents.

The rest of this document is organized as follows. Section 4 presents an overview on CCMP, together with some implementation-related details and related matters like HTTPS transport and notifications. Section 5 presents the reader with examples showing the different approaches CCMP provides to create a new conference. Section 6 more generally addresses the different user-related manipulations that can be achieved by means of CCMP, by presenting a number of interesting scenarios. Section 7 addresses several scenarios that may involve the use of sidebars. Section 8 shows how CCMP can be used to remove conferences and users from the system. Finally, Section 9 provides a few details on the security considerations when it comes to implementing CCMP.

4. Working with CCMP

This section provides a brief introduction as to how the Centralized Conferencing Manipulation Protocol (CCMP) [RFC6503] works and how it can be transported across a network. A typical CCMP interaction focusing on relevant aspects of the client-server communication is

described. Please note that this section assumes the reader has read and understood the CCMP document. This section is intended to help the reader understand the actual protocol interactions.

First, a description of the protocol itself is provided Section 4.1, including some implementation considerations. In Section 4.2, an effective CCMP interaction is presented by exploiting HTTPS as a transport. Finally, notifications are described in Section 4.3.

The document then presents and describes some actual flows in detail in the sections to follow.

4.1. CCMP and the Data Model

CCMP is an protocol based on XML [W3C.REC-xml-20081126]. It has been designed as a request/response protocol. It is completely stateless, which means implementations can safely handle transactions independently from each other.

The protocol allows for the manipulation of conference objects and related users. This manipulation allows a conference and media control client (briefly CMCC in all the following sections) to create, update, and remove basically everything that is related to the objects handled by a conferencing system. This is reflected in the allowed operations (retrieve, create, update, delete) and the specified request types (ranging from the manipulation of blueprints and conferences to users and sidebars). For instance, CCMP provides ways to:

- retrieve the list of registered and/or active conferences in the system;
- create new conferences by exploiting several different approaches;
- add/remove users to/from a conference;
- update a conference with respect to all of its aspects;

and so on.

While CCMP acts as the means to manipulate conference objects, CCMP does not define these conference objects. A separate document specifies how a conference object and all its components have to be constructed (Conference Information Data Model for Centralized Conferencing (XCON) [RFC6501]). CCMP, depending upon the request type and the related operation, carries pieces of conference objects (or any object as a whole) according to the aforementioned

specification. This means that any implementation aiming at being compliant with CCMP has to make sure that the transported objects are completely compliant with the data model specification and coherent with the constraints defined therein. To make this clearer, there are elements that are mandatory in a conference object: issuing a syntactically correct CCMP request that carries a wrong conference object is doomed to result in a failure. For this reason, it is suggested that the interested implementers take special care in carefully checking the data model handlers as well in order to avoid potential mistakes.

However, there are cases when a mandatory element in the data model cannot be assigned in a conference object by a CCMP user. For example, a CMCC may be requesting the direct creation of a new conference; in this case, a conference object assumes an 'entity' attribute uniquely identifying the conference to be in place. Thus, the CMCC has no way to know a priori what the entity will be, since it is generated by the ConfS after the request. For scenarios like this one, the CCMP specification describes the use of a dedicated placeholder wildcard (i.e., "AUTO_GENERATE_X", where X is an integer) to make the conference object compliant with the data model: the wildcard would then be replaced by the ConfS with the right value.

4.2. Using HTTP/TLS as a Transport

CCMP requires that implementations support HTTP/TLS as the transport mechanism. Per CCMP, a CMCC sends a request as part of an HTTPS POST message, and the ConfS would reply with a 200 OK HTTPS response. In both cases, the HTTPS messages carry the CCMP messages as payload, which is reflected in the Content-Type header ("application/ccmp+xml"). Figure 1 presents a ladder diagram of such an interaction, which is followed by a dump of the exchanged HTTPS messages for further analysis. The examples in the remainder of this document show only the CCMP interactions.

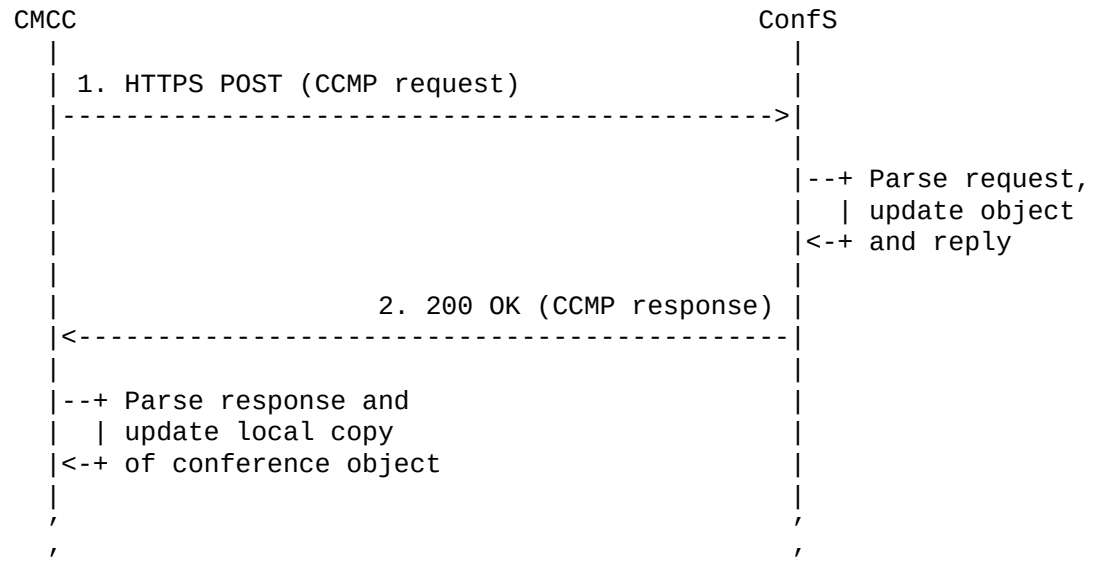


Figure 1: CCMP on HTTPS

Per the protocol dump in the following lines, the CMCC has issued a CCMP request (a blueprintRequest message asking for a blueprint retrieval, i.e., with the <operation> element set to "retrieve") towards the ConfS. The request has been carried as payload of an HTTPS POST (message 1.) towards a previously known location. The mandatory Host header has been specified, and the Content-Type header has been correctly set as well ("application/ccmp+xml").

The ConfS, in turn, has handled the request and replied accordingly. The response (a blueprintResponse message with a <response-code> set to a successful value, "200") has been carried as payload of a 200 OK HTTPS response (message 2.). As before, the Content-Type header has been correctly set ("application/ccmp+xml").

1. CMCC -> ConfS (HTTPS POST, CCMP request)

```
POST /Xcon/Ccmp HTTP/1.1
Content-Length: 657
Content-Type: application/ccmp+xml
Host: example.com:443
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.0.1 (java 1.5)
```

```
   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <ccmp:ccmpRequest
         xmlns:info="urn:ietf:params:xml:ns:conference-info"
         xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
         xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
     <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:type="ccmp:ccmp-blueprint-request-message-type">
           <confUserID>xcon-userid:Alice@example.com</confUserID>
           <confObjID>xcon:AudioRoom@example.com</confObjID>
           <operation>retrieve</operation>
           <ccmp:blueprintRequest/>
     </ccmpRequest>
   </ccmp:ccmpRequest>
```

2. CMCC <- ConfS (200 to POST, CCMP response)

```
   HTTP/1.1 200 OK
   X-Powered-By: Servlet/2.5
   Server: Sun GlassFish Communications Server 1.5
   Content-Type: application/ccmp+xml;charset=ISO-8859-1
   Content-Length: 1652
   Date: Thu, 04 Feb 2010 14:47:56 GMT

   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <ccmp:ccmpResponse
         xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
         xmlns:info="urn:ietf:params:xml:ns:conference-info"
         xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
     <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xsi:type="ccmp:ccmp-blueprint-response-message-type">
       <confUserID>xcon-userid:Alice@example.com</confUserID>
       <confObjID>xcon:AudioRoom@example.com</confObjID>
       <operation>retrieve</operation>
       <response-code>200</response-code>
       <response-string>success</response-string>
       <ccmp:blueprintResponse>
         <blueprintInfo entity="xcon:AudioRoom@example.com">
           <info:conference-description>
              <info:display-text>AudioRoom</info:display-text>
              <info:maximum-user-count>2</info:maximum-user-count>
              <info:available-media>
                <info:entry label="audioLabel">
                    <info:type>audio</info:type>
                </info:entry>
                </info:available-media>
           </info:conference-description>
           <info:users>
              <xcon:join-handling>allow</xcon:join-handling>
```

```
          </info:users>
          <xcon:floor-information>
            <xcon:floor-request-handling>confirm
            </xcon:floor-request-handling>
            <xcon:conference-floor-policy>
                  <xcon:floor id="audioLabel"></xcon:floor>
            </xcon:conference-floor-policy>
          </xcon:floor-information>
        </blueprintInfo>
      </ccmp:blueprintResponse>
    </ccmpResponse>
  </ccmp:ccmpResponse>
```

For completeness, the following provides some details of the CCMP interaction. Despite the simplicity of the request, this flow provides some relevant information on how CCMP messages are built. Specifically, both the CCMP request and the CCMP response share a subset of the message:

- <confUserID>: this element, provided by the CMCC, refers to the requester by means of his XCON-USERID; except in a few scenarios (presented in the following sections), this element must always contain a valid value;

- <confObjID>: this element refers to the target conference object, according to the request in place;

- <operation>: this element specifies the operation the CMCC wants to perform, according to the specific request type.

Besides those elements, the CMCC (let’s say Alice, whose XCON-USERID is "xcon-userid:Alice@example.com") has also provided an additional element, <blueprintRequest>. The name of that element varies according to the request type in which the CMCC is interested. In this specific scenario, the CMCC was interested in acquiring details concerning a specific blueprint (identified by its XCON-URI "xcon:AudioRoom@example.com", as reflected in the provided <confObjID> target element), and so the request consisted in an empty <blueprintRequest> element. It will be clearer in the following sections that different request types may require different elements and, as a consequence, different content.

Considering the request was a blueprintRequest message, the ConfS has replied with a blueprintResponse message containing a <blueprintResponse> element. This element includes a complete dump of the conference object (compliant with the data model) describing the requested blueprint.

Without providing additional details of this interaction, it is worth noting that this was the example of the simplest CCMP communication that could take place between a CMCC and a ConfS, a blueprint request: this scenario will be described in more detail in Section 5.2.

4.3. Conference Notifications

The XCON framework [RFC5239] identifies several different possible protocol interactions between a conference server and a conferencing client. One of those interactions is generically called "notification protocol" providing a mechanism for all clients interested in being informed by the server whenever something relevant happens in a conference. When SIP is used as the call signaling protocol in a CCMP implementation, the XCON event package [RFC6502], which extends the SIP event package for conference state [RFC4575] must be supported. A SIP client uses the SIP SUBSCRIBE message for the XCON event package to subscribe to notifications related to a specific conference. A SIP client would receive notifications describing all the changes to the document via a SIP NOTIFY message. An example ladder diagram is presented in Figure 2; in this figure, we assume a CMCC has updated a conference object, and a previously subscribed SIP client is notified of the update.

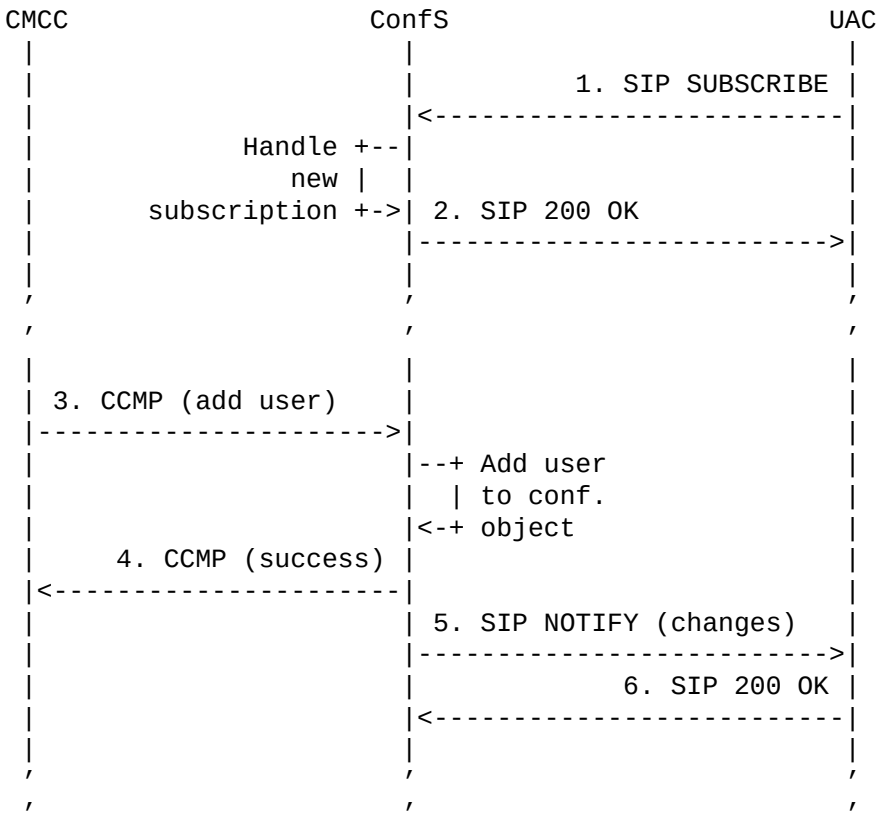


Figure 2: XCON Event Package: SIP Notifications

The detailed flows in this document generically present a notification, when appropriate, but do not include the SIP messaging details.

5. Conference Creation

This section provides details associated with the various ways in which a conference can be created using CCMP and the XCON framework constructs. As previously mentioned, the details of the media control, call signaling, and floor control protocols, where applicable, are annotated in the flows without showing all the details. This also applies to CCMP, whose flows are related to the protocol alone, hiding any detail concerning the transport that may have been used (e.g., HTTPS). However, for clarification purposes, the first example in Section 5.1 provides the details of the media control messaging with the standard annotation used throughout the remainder of this document. In subsequent flows, only this

annotation (identified by lowercase letters) is included, and the reader is encouraged to refer to the call flows in the relevant documents for details about the other protocols. The annotations for the call signaling are on the left side of the conference server vertical bar, and those for the media control messaging are on the right side.

5.1. Basic Conference Creation

The simplest manner in which a conference can be created is accomplished by the client sending a confRequest message with the <operation> element set to "create" as the only parameter to the conference server, together with the <confUserID> associated with the requesting client itself. This results in the creation of a default conference, with an XCON-URI in the form of the <confObjID> element, the XCON-USERID in the form of the <confUserID> element (the same one already present in the request), and the data for the conference object in the <confInfo> parameter all returned in the confResponse message. This example also adds the issuing user to the conference upon creation with the 'method' attribute in the <target> child element of <allowed-users-list> set to "dial-out".

The specific data for the conference object is returned in the confResponse message in the <confInfo> parameter. This allows the client (with the appropriate authorization) to manipulate these data and add additional participants to the conference, as well as change the data during the conference. In addition, the client may distribute the conferencing information to other participants allowing them to join, the details of which are provided in additional flows. Please notice that, according to the CCMP specification, the return of the new conference data in the <confInfo> element is not mandatory: if the <confInfo> parameter of is not included in the successful confResponse/create message, a subsequent confRequest/retrieve message of the returned <confObjID> can be triggered to provide the requesting client with the detailed conference description.

Clients that are not XCON-aware can join the conference using a specific signaling interface such as SIP [RFC3261] (using the signaling interface to the conference focus as described in [RFC4579]), or other supported signaling protocols, being XCON-agnostic with respect to them. However, these details are not shown in the message flows. The message flows in this document identify the point in the message flows at which this signaling occurs via the lowercase letter items (i.e., (a)...(x)) along with the appropriate text for the processing done by the conference server.

As previously described, this example also shows how the conferencing system may make use of other standard protocol components for complete functionality. An example of that is the media control framework [RFC5567], which allows the conferencing system to configure conference mixes, Interactive Voice Response (IVR) dialogs, and all sorts of media-related interactions an application like this may need. In order to provide the reader with some insight on these interactions, the conference server in this example also configures and starts a mixer via a media control channel as soon as the conference is created (transactions A1 and A2), and attaches clients to it when necessary (e.g., when CMCC1 joins the conference by means of SIP signaling, its media channels are attached to the media server (MS) in B1/B2). Note, that the media control interfaces are NOT shown in the remaining call flows in this document but rather follow the same annotation as with the SIP signaling such that (b) correlates with the A1 and A2 transactions and (d) correlates with the B1 and B2 transactions.

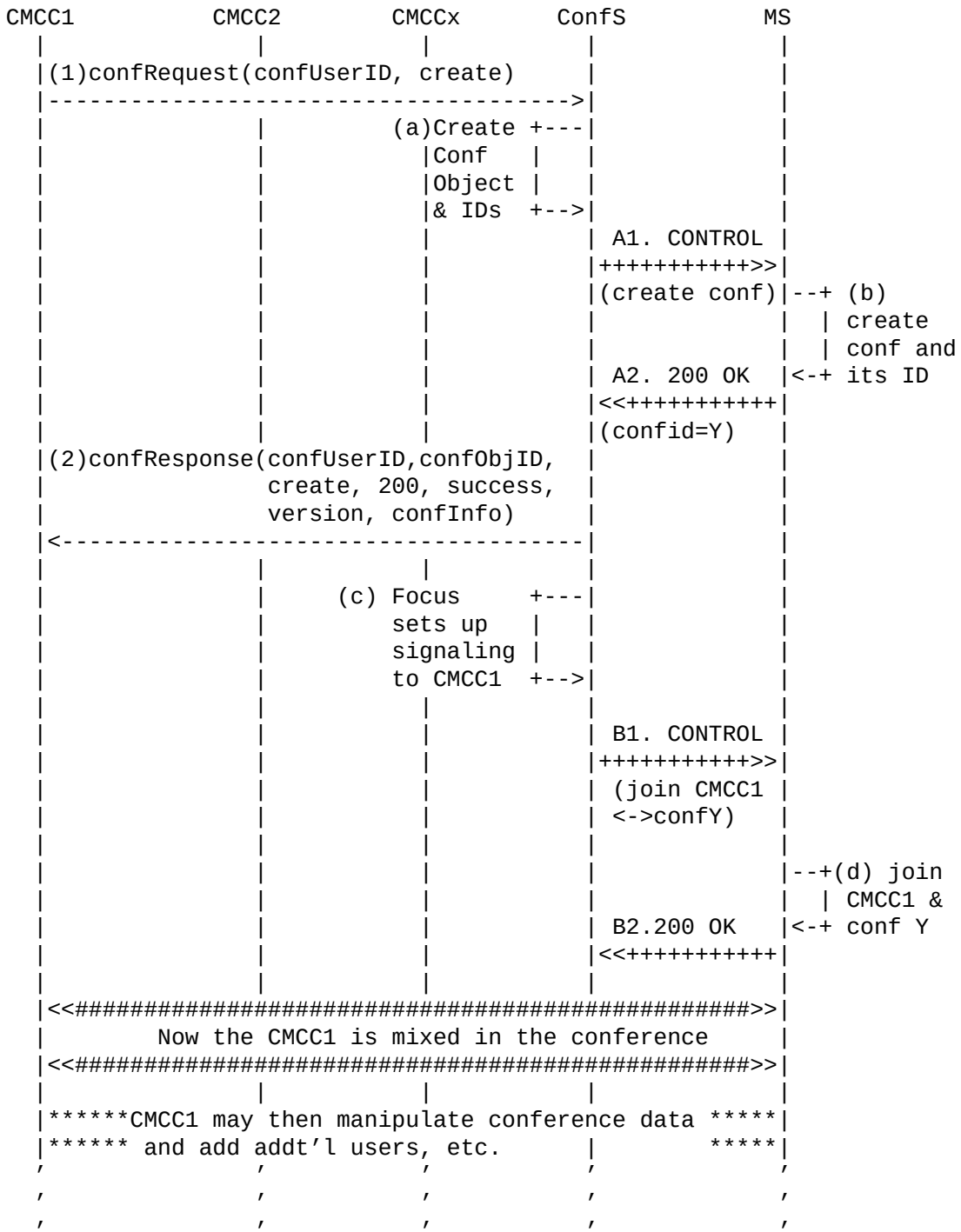


Figure 3: Create Basic Conference - Complete flow

1. confRequest/create message (Alice creates a default conference)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest
      xmlns:info="urn:ietf:params:xml:ns:conference-info"
      xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
      xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
   <ccmpRequest
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ccmp:ccmp-conf-request-message-type">
      <confUserID>xcon-userid:Alice@example.com</confUserID>
      <operation>create</operation>
      <ccmp:confRequest/>
   </ccmpRequest>
</ccmp:ccmpRequest>
```

2. confResponse/create message ("success", created conference object returned)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <ccmp:ccmpResponse
        xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
        xmlns:info="urn:ietf:params:xml:ns:conference-info"
        xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
       <ccmpResponse
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="ccmp:ccmp-conf-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
        <operation>create</operation>
        <response-code>200</response-code>
        <response-string>success</response-string>
        <version>1</version>
        <ccmp:confResponse>
            <confInfo entity="xcon:8977794@example.com">
               <info:conference-description>
                   <info:display-text>
                      Default conference initiated by Alice
                   </info:display-text>
                   <info:conf-uris>
                      <info:entry>
                         <info:uri>
                             xcon:8977794@example.com
                         </info:uri>
                         <info:display-text>
                             Conference XCON-URI
                         </info:display-text>
```

```
                        </info:entry>
                    </info:conf-uris>
                    <info:maximum-user-count>10
                    </info:maximum-user-count>
                    <info:available-media>
                           <info:entry label="11">
                               <info:type>audio</info:type>
                           </info:entry>
                    </info:available-media>
                    </info:conference-description>
                     <info:conference-state>
                       <info:active>false</info:active>
                     </info:conference-state>
                <info:users>
                   <xcon:join-handling>allow</xcon:join-handling>
                   <xcon:allowed-users-list>
                      <xcon:target uri="xcon-userid:Alice@example.com"
                                   method="dial-out"/>
                    </xcon:allowed-users-list>
                </info:users>
            </confInfo>
        </ccmp:confResponse>
     </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 4: Create Basic Conference Detailed Messaging

5.2. Conference Creation Using Blueprints

The previous example showed the creation of a new conference using default values. This means the client provided no information about how she wanted the conference to be created. The XCON framework (and CCMP as a consequence) allows for the implementation of templates. These templates are called "conference blueprints" and are basically conference objects with predefined settings. This means that a client might get a list of blueprints, choose the one that most fits his needs, and use the chosen blueprint to create a new conference.

Figure 5 provides an example of one client, Alice, discovering the conference blueprints available for a particular conferencing system and creating a conference based on the desired blueprint. In particular, Alice is interested in those blueprints suitable to represent a video conference, i.e., a conference in which both audio and video are available, so she makes use of the filter mechanism provided by CCMP to make a selective blueprints retrieve request. This results in three distinct CCMP transactions.

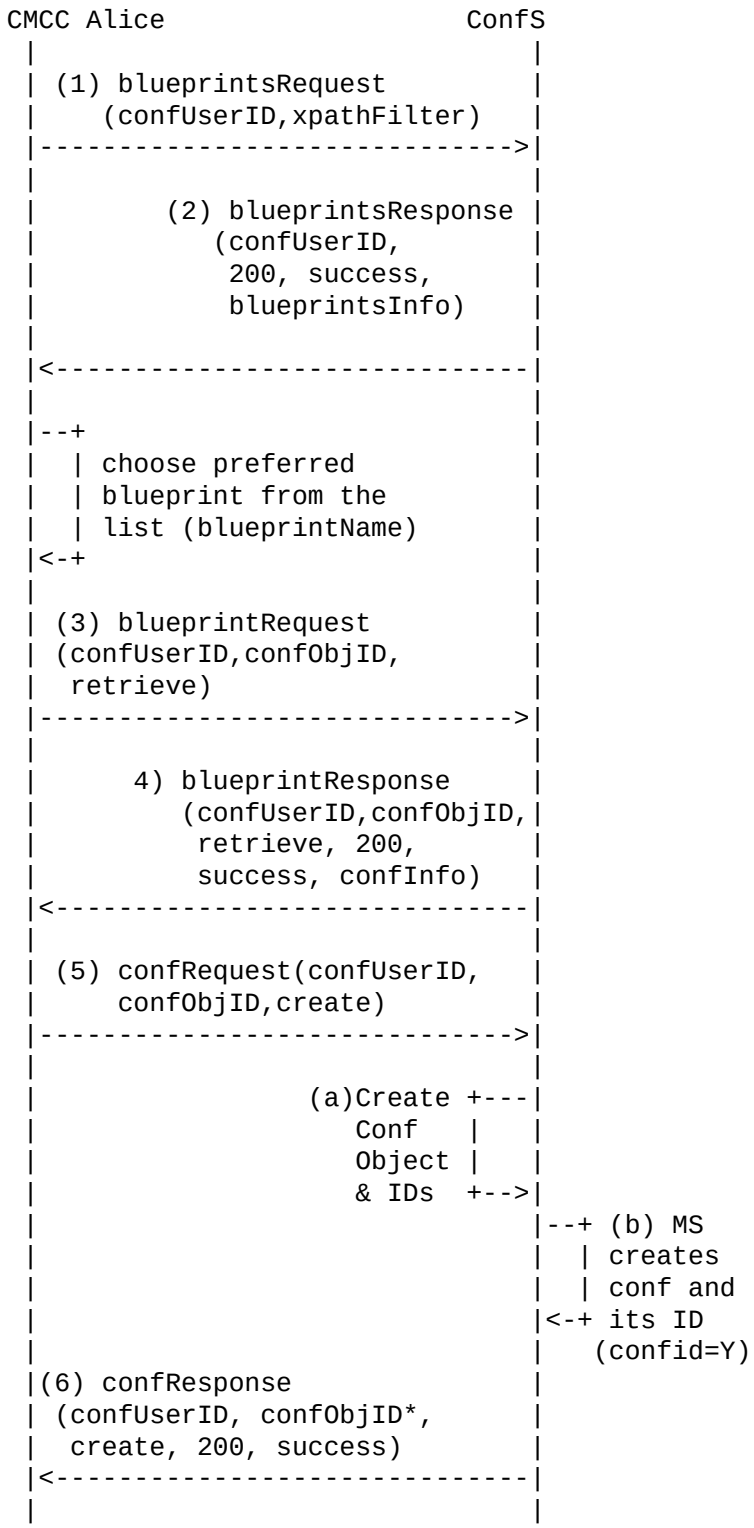

CMCC Alice
ConfS
(1) blueprintsRequest
(confUserID,xpathFilter)
(2) blueprintsResponse
(confUserID,
200, success,
blueprintsInfo)
choose preferred
blueprint from the
list (blueprintName)
(3) blueprintRequest
(confUserID,confObjID,
retrieve)
4) blueprintResponse
(confUserID,confObjID,
retrieve, 200,
success, confInfo)
(5) confRequest(confUserID,
confObjID,create)
(a)Create
Conf
Object
& IDs
(b) MS
creates
conf and
its ID
(confid=Y)
(6) confResponse
(confUserID, confObjID*,
create, 200, success)

Figure 5: Client Creation of Conference Using Blueprints

1. Alice first sends a blueprintsRequest message to the conference server identified by the conference server discovery process. This request contains the <confUserID> set to the XCON-USERID of the user issuing the request (in this case, the one belonging to Alice) and the <xpathFilter> element by which Alice specifies she desires to obtain only blueprints providing support for both audio and video: for this purpose, the xpath query contained in this field is: "/conference-info[conference-description/ available-media/entry/type='audio' and conference-description/ available-media/entry/type='video']". Upon receipt of the blueprintsRequest message, the conference server would first ensure, on the basis of the <confUserID> parameter, that Alice has the appropriate authority based on system policies to receive the requested kind of blueprints supported by that system.

2. All blueprints that Alice is authorized to use are returned in a blueprintsResponse message in the <blueprintsInfo> element.

3. Upon receipt of the blueprintsResponse message containing the blueprints, Alice determines which blueprint to use for the conference to be created. Alice sends a blueprintRequest message to get the specific blueprint as identified by the <confObjID>.

4. The conference server returns the details associated with the specific blueprint identified by the <confObjID> in the <confInfo> element within the blueprintResponse message.

5. Alice finally sends a confRequest message with a "create" <operation> to the conference server to create a conference reservation cloning the chosen blueprint. This is achieved by writing the blueprint's XCON-URI in the <confObjID> parameter.

6. Upon receipt of the confRequest/create message, the conference server uses the received blueprint to clone a conference, allocating a new XCON-URI (called "confObjID*" in the example). The conference server then sends a confResponse message including the new "confObjID*" associated with the newly created conference instance as the value of the <confObjID> parameter. Upon receipt of the confResponse message, Alice can now add other users to the conference.

1. blueprintsRequest message (Alice requires the list of the available blueprints with video support)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest xmlns:info="urn:ietf:params:xml:ns:conference-info"
  xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
  xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
 <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="ccmp:ccmp-blueprints-request-message-type">
    <confUserID>xcon-userid:Alice@example.com</confUserID>
    <ccmp:blueprintsRequest>
      <xpathFilter>/conference-info[conference-description/
         available-media/entry/type='audio'
         and
         conference-description/available-media/entry/type='video']
      </xpathFilter>
    </ccmp:blueprintsRequest>
 </ccmpRequest>
</ccmp:ccmpRequest>
```

2. blueprintsResponse message (the server provides a descriptions of the available blueprints fitting Alice's request)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
 xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
 xmlns:info="urn:ietf:params:xml:ns:conference-info"
 xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
<ccmpResponse
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ccmp:ccmp-blueprints-response-message-type">
   <confUserID>xcon-userid:Alice@example.com</confUserID>
   <response-code>200</response-code>
   <response-string>success</response-string>
     <ccmp:blueprintsResponse>
      <blueprintsInfo>
       <info:entry>
        <info:uri>xcon:VideoRoom@example.com</info:uri>
        <info:display-text>VideoRoom</info:display-text>
        <info:purpose>Video Room:
            conference room with public access,
            where both audio and video are available,
            4 users can talk and be seen at the same time,
            and the floor requests are automatically accepted.
        </info:purpose>
       </info:entry>
       <info:entry>
```

```
          <info:uri>xcon:VideoConference1@example.com</info:uri>
          <info:display-text>VideoConference1</info:display-text>
            <info:purpose>Public Video Conference: conference
                where both audio and video are available,
                only one user can talk
            </info:purpose>
          </info:entry>
        </blueprintsInfo>
       </ccmp:blueprintsResponse>
     </ccmpResponse>
   </ccmp:ccmpResponse>
```

3. blueprintRequest/retrieve message (Alice wants the "VideoRoom" blueprint)

```
   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <ccmp:ccmpRequest
         xmlns:info="urn:ietf:params:xml:ns:conference-info"
         xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
         xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
     <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xsi:type="ccmp:ccmp-blueprint-request-message-type">
           <confUserID>xcon-userid:Alice@example.com</confUserID>
           <confObjID>xcon:VideoRoom@example.com</confObjID>
           <operation>retrieve</operation>
           <ccmp:blueprintRequest/>
     </ccmpRequest>
   </ccmp:ccmpRequest>
```

4. blueprintResponse/retrieve message ("VideoRoom" conference object returned)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <ccmp:ccmpResponse
         xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
         xmlns:info="urn:ietf:params:xml:ns:conference-info"
         xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
     <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:type="ccmp:ccmp-blueprint-response-message-type">
       <confUserID>xcon-userid:Alice@example.com</confUserID>
       <confObjID>xcon:VideoRoom@example.com</confObjID>
       <operation>retrieve</operation>
       <response-code>200</response-code>
       <response-string>success</response-string>
       <ccmp:blueprintResponse>
         <blueprintInfo entity="xcon:VideoRoom@example.com">
           <info:conference-description>
              <info:display-text>VideoRoom</info:display-text>
```

```
              <info:maximum-user-count>4</info:maximum-user-count>
              <info:available-media>
                <info:entry label="audioLabel">
                    <info:type>audio</info:type>
                </info:entry>
                <info:entry label="videoLabel">
                    <info:type>video</info:type>
                </info:entry>
                </info:available-media>
            </info:conference-description>
            <info:users>
               <xcon:join-handling>allow</xcon:join-handling>
            </info:users>
            <xcon:floor-information>
              <xcon:floor-request-handling>confirm
              </xcon:floor-request-handling>
              <xcon:conference-floor-policy>
                    <xcon:floor id="audioFloor">
                     <xcon:media-label>audioLabel</xcon:media-label>
                    </xcon:floor>
                    <xcon:floor id="videoFloor">
                         <xcon:media-label>videoLabel</xcon:media-label>
                    </xcon:floor>
              </xcon:conference-floor-policy>
            </xcon:floor-information>
          </blueprintInfo>
        </ccmp:blueprintResponse>
      </ccmpResponse>
    </ccmp:ccmpResponse>
```

5. confRequest/create message (Alice clones the "VideoRoom" blueprint)

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpRequest
        xmlns:info="urn:ietf:params:xml:ns:conference-info"
        xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
        xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
     <ccmpRequest
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:type="ccmp:ccmp-conf-request-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:VideoRoom@example.com</confObjID>
        <operation>create</operation>
        <ccmp:confRequest/>
     </ccmpRequest>
  </ccmp:ccmpRequest>
```

6. confResponse/create message (cloned conference object returned)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
     xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
     xmlns:info="urn:ietf:params:xml:ns:conference-info"
     xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
  <ccmpResponse
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:type="ccmp:ccmp-conf-response-message-type">
     <confUserID>xcon-userid:Alice@example.com</confUserID>
     <confObjID>xcon:8977794@example.com</confObjID>
     <operation>create</operation>
     <response-code>200</response-code>
     <response-string>success</response-string>
     <version>1</version>
     <ccmp:confResponse>
          <confInfo entity="xcon:8977794@example.com">
            <info:conference-description>
                <info:display-text>
                   New conference by Alice cloned from VideoRoom
                </info:display-text>
                <info:conf-uris>
                   <info:entry>
                      <info:uri>
                          xcon:8977794@example.com
                      </info:uri>
                      <info:display-text>
                          conference xcon-uri
                      </info:display-text>
                      <xcon:conference-password>
                          8601
                      </xcon:conference-password>
                   </info:entry>
                </info:conf-uris>
                <info:maximum-user-count>10</info:maximum-user-count>
                <info:available-media>
                       <info:entry label="11">
                           <info:type>audio</info:type>
                       </info:entry>
                       <info:entry label="12">
                           <info:type>video</info:type>
                       </info:entry>
                </info:available-media>
            </info:conference-description>
            <info:users>
                <xcon:join-handling>allow</xcon:join-handling>
```

```
              </info:users>
                 <xcon:floor-information>
                    <xcon:floor-request-handling>
                       confirm</xcon:floor-request-handling>
                    <xcon:conference-floor-policy>
                      <xcon:floor id="1">
                      <xcon:media-label>11</xcon:media-label>
                      </xcon:floor>
                      <xcon:floor id="2">
                      <xcon:media-label>12</xcon:media-label>
                      </xcon:floor>
                    </xcon:conference-floor-policy>
                 </xcon:floor-information>
              </confInfo>
           </ccmp:confResponse>
        </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 6: Create Conference (Blueprint) Detailed Messaging

5.3. Conference Creation Using User-Provided Conference Information

A conference can also be created by the client sending a confRequest message with the "create" <operation>, along with the desired data in the form of the <confInfo> element for the conference to be created. The request also includes the <confUserID> set to the XCON-USERID of the requesting entity.

This approach allows for a client (in this example Alice) to completely describe what the conference object should look like, without relying on defaults or blueprints; for example, which media should be available, the topic, the users allowed to join, any scheduling-related information, and so on. This can be done by providing, in the creation request, a full conference object for the server to parse.

This <confInfo> parameter must comply with the data model specification. This means that the 'entity' attribute is mandatory and cannot be missing in the document. However, in this example, the client is actually requesting the creation of a new conference, which doesn't exist yet, so the 'entity' attribute is unknown. As discussed in Section 4.1, CCMP allows for the use of a wildcard placeholder. This placeholder ("xcon:AUTO_GENERATE_1@example.com" in the example) is only to ensure the <confInfo> element is compliant with the data model and would subsequently be replaced by the conference server with the actual value. Thus, when the conference server actually creates the conference, a valid value for the 'entity' attribute is created for it as well, which takes the place

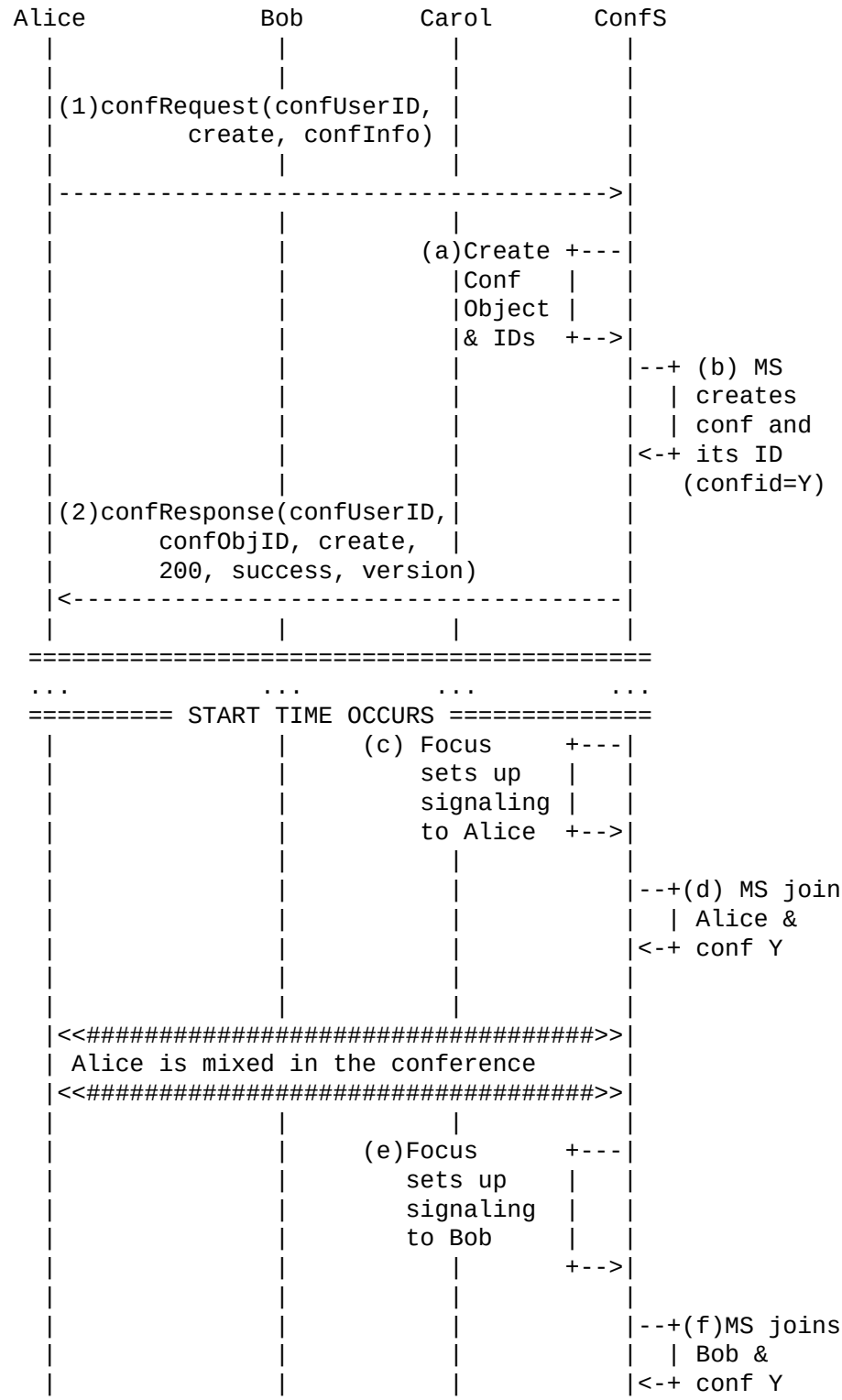
of the wildcard value when the actual conference object provided by the client is populated.

To give a flavor of what could be added to the conference object, we assume Alice is also interested in providing scheduling-related information. So, in this example, Alice also specifies by the <conference-time> element included in the <confInfo> that the conference she wants to create has to occur on a certain date spanning from a certain start time to a certain stop time and has to be replicated weekly.

Moreover, Alice indicates by means of the <allowed-users-list> element that at the start time Bob, Carol, and herself have to be called by the conferencing system to join the conference (in fact, for each <target> field corresponding to one of the aforementioned clients, the 'method' attribute is set to "dial-out").

Once Alice has prepared the <confInfo> element and sent it as part of her request to the server, if the conferencing system can support that specific type of conference (capabilities, etc.), then the request results in the creation of a conference. We assume the request has been successful, and as a consequence, the XCON-URI in the form of the <confObjID> parameter and the XCON-USERID in the form of the <confUserID> parameter (again, the same as the requesting entity) are returned in the confResponse message.

In this example, the created conference object is not returned in the successful confResponse message in the <confInfo> parameter. Nevertheless, Alice could still retrieve the actual conference object by issuing a confRequest message with a "retrieve" <operation> on the XCON-URI returned in the <confObjID> of the previous response. Such a request would show how, as described at the beginning of this section, the 'entity' attribute of the conference object in the <confInfo> field is replaced with the actual information (i.e., "xcon:6845432@example.com").


Alice
Bob
Carol
ConfS
(1)confRequest(confUserID, create, confInfo)
(a)Create Conf Object & IDs
(b) MS creates conf and its ID (confid=Y)
(2)confResponse(confUserID, confObjID, create, 200, success, version)
START TIME OCCURS
(c) Focus sets up signaling to Alice
(d) MS joins Alice & conf Y
Alice is mixed in the conference
(e)Focus sets up signaling to Bob
(f)MS joins Bob & conf Y

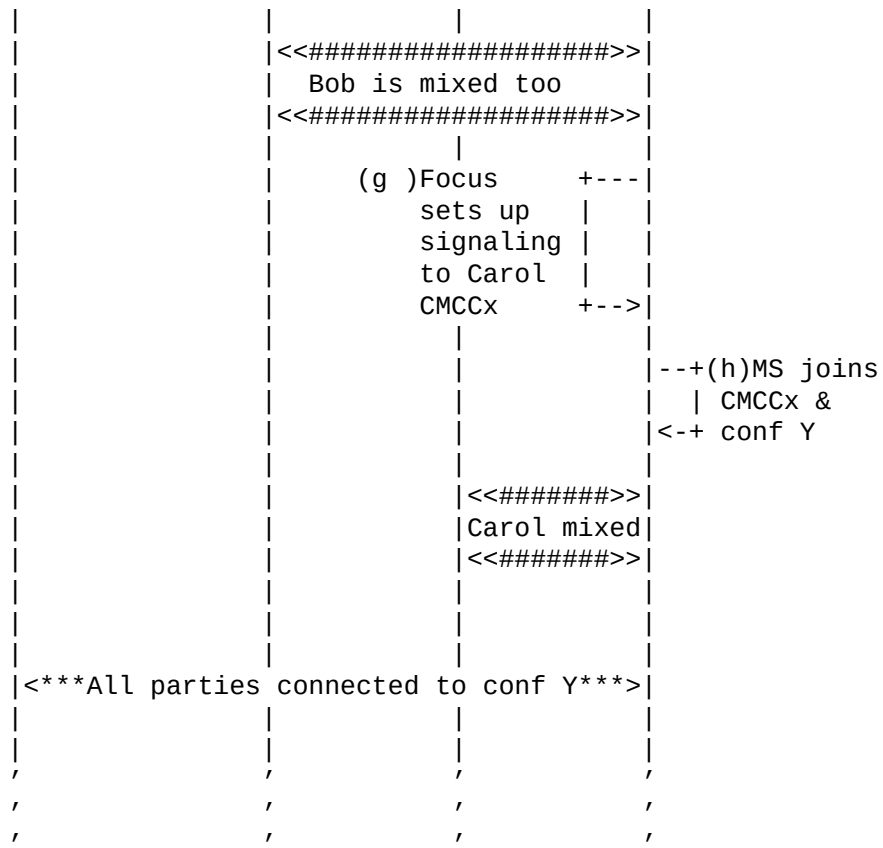


Figure 7: Create Basic Conference from User-Provided Conference Info

1. confRequest/create message (Alice proposes a conference object to be created)

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest
      xmlns:info="urn:ietf:params:xml:ns:conference-info"
      xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
      xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
   <ccmpRequest
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ccmp:ccmp-conf-request-message-type">
      <confUserID>xcon-userid:Alice@example.com</confUserID>
      <operation>create</operation>
      <ccmp:confRequest>
         <confInfo entity="xcon:AUTO_GENERATE_1@example.com">
            <info:conference-description>
                <info:display-text>
                   Dial-out conference initiated by Alice
```

```
                </info:display-text>
                 <info:maximum-user-count>10</info:maximum-user-count>
                 <info:available-media>
                        <info:entry label="AUTO_GENERATE_2">
                            <info:type>audio</info:type>
                        </info:entry>
                 </info:available-media>
                 <xcon:conference-time>
                  <xcon:entry>
                   <xcon:base>
                     BEGIN:VCALENDAR
                     VERSION:2.0
                     PRODID:-//Mozilla.org/NONSGML
                               Mozilla Calendar V1.0//EN
                     BEGIN:VEVENT
                     DTSTAMP: 20100127T140728Z
                     UID: 20100127T140728Z-345FDA-alice@example.com
                     ORGANIZER:MAILTO:alice@example.com
                     DTSTART:20100127T143000Z
                     RRULE:FREQ=WEEKLY
                     DTEND: 20100127T163000Z
                     END:VEVENT
                     END:VCALENDAR
                   </xcon:base>
                   <xcon:mixing-start-offset
                    required-participant="moderator">
                        2010-01-27T14:29:00Z
                   </xcon:mixing-start-offset>
                   <xcon:mixing-end-offset
                    required-participant="participant">
                        2010-01-27T16:31:00Z
                   </xcon:mixing-end-offset>
                   <xcon:must-join-before-offset>
                        2010-01-27T15:30:00Z
                   </xcon:must-join-before-offset>
                  </xcon:entry>
                 </xcon:conference-time>
            </info:conference-description>
            <info:users>
               <xcon:join-handling>allow</xcon:join-handling>
               <xcon:allowed-users-list>
                  <xcon:target uri="xcon-userid:alice@example.com"
                               method="dial-out"/>
                  <xcon:target uri="sip:bob83@example.com"
                               method="dial-out"/>
                  <xcon:target uri="sip:carol@example.com"
                               method="dial-out"/>
                </xcon:allowed-users-list>
```

```
              </info:users>
           </confInfo>
        </ccmp:confRequest>
     </ccmpRequest>
  </ccmp:ccmpRequest>

  2. confResponse/create message ("200", "success")

  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpResponse
       xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
       xmlns:info="urn:ietf:params:xml:ns:conference-info"
       xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
      <ccmpResponse
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ccmp:ccmp-conf-response-message-type">
       <confUserID>xcon-userid:Alice@example.com</confUserID>
       <confObjID>xcon:6845432@example.com</confObjID>
       <operation>create</operation>
       <response-code>200</response-code>
       <response-string>success</response-string>
       <version>1</version>
       <ccmp:confResponse/>
      </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 8: Create Basic Conference Detailed Messaging

5.4. Cloning an Existing Conference

A client can also create another conference by cloning an existing conference, such as an active conference or conference reservation. This approach can be seen as a logical extension of the creation of a new conference using a blueprint: the difference is that, instead of cloning the predefined settings listed in a blueprint, the settings of an existing conference would be cloned.

In this example, the client sends a confRequest message with the "create" <operation>, along with her XCON-USERID in the <confUserID> element and the XCON-URI of the conference from which a new conference is to be cloned in the <confObjID> element.

An example of how a client can create a conference based on a blueprint is provided in Section 5.2. The manner by which a client in this example might learn about a conference reservation or active conferences is similar to the first step in the blueprint example, with the exception of querying for different types of conference

objects supported by the specific conferencing system. For instance, in this example, the client clones a conference reservation (i.e., an inactive conference).

If the conferencing system can support a new instance of the specific type of conference (capabilities, etc.), then the request results in the creation of a conference, with an XCON-URI in the form of a new value in the <confObjID> parameter to reflect the newly cloned conference object returned in the confResponse message.

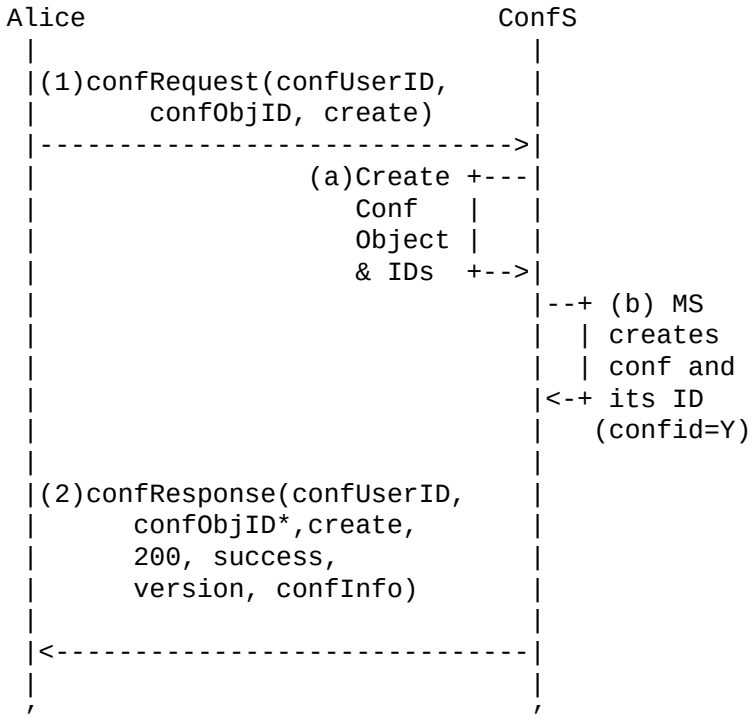


Figure 9: Create Basic Conference - Clone

1. Alice, a conferencing system client, sends a confRequest message to clone a conference based on an existing conference reservation. Alice indicates this conference should be cloned from the specified parent conference represented by the XCON-URI in the <confObjID> provided in the request.

2. Upon receipt of the confRequest message containing a "create" <operation> and the aforementioned XCON-URI in the <confObjID>, the conference server ensures that such received XCON-URI is valid. The conference server determines the appropriate read/write access of any users to be added to a conference based on this XCON-URI (using membership, roles, etc.). The conference

server uses the received <confObjID> to clone a conference reservation. The conference server also reserves or allocates a new XCON-URI (called "confObjID*" in Figure 9) to be used for the cloned conference object. This new identifier is, of course, different from the one associated with the conference to be cloned, since it represents a different conference object. Any subsequent protocol requests from any of the members of the conference must use this new identifier. The conference server maintains the mapping between this conference ID and the parent conference object ID associated with the reservation through the conference instance, and this mapping is explicitly addressed through the <cloning-parent> element of the <conference-description> in the new conference object.

1. confRequest/create message (Alice clones an existing conference)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest
      xmlns:info="urn:ietf:params:xml:ns:conference-info"
      xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
      xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
   <ccmpRequest
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ccmp:ccmp-conf-request-message-type">
      <confUserID>xcon-userid:Alice@example.com</confUserID>
      <confObjID>xcon:6845432@example.com</confObjID>
      <operation>create</operation>
      <ccmp:confRequest/>
   </ccmpRequest>
</ccmp:ccmpRequest>
```

2. confResponse/create message (created conference object returned)

```
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
     xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
     xmlns:info="urn:ietf:params:xml:ns:conference-info"
     xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
  <ccmpResponse
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:type="ccmp:ccmp-conf-response-message-type">
     <confUserID>xcon-userid:Alice@example.com</confUserID>
     <confObjID>xcon:8977794@example.com</confObjID>
     <operation>create</operation>
     <response-code>200</response-code>
     <response-string>success</response-string>
     <version>1</version>
```

```
      <ccmp:confResponse>
           <confInfo entity="xcon:8977794@example.com">
             <info:conference-description>
                 <info:display-text>
                    New conference by Alice cloned from 6845432
                 </info:display-text>
                  <info:maximum-user-count>10</info:maximum-user-count>
                  <info:available-media>
                         <info:entry label="11">
                             <info:type>audio</info:type>
                         </info:entry>
                  </info:available-media>
                  <xcon:cloning-parent>
                     xcon:6845432@example.com
                 </xcon:cloning-parent>
              </info:conference-description>
              <info:users>
                  <xcon:join-handling>allow</xcon:join-handling>
                     <xcon:allowed-users-list>
                    <xcon:target uri="sip:alice@example.com"
                                 method="dial-out"/>
                    <xcon:target uri="sip:bob83@example.com"
                                 method="dial-out"/>
                    <xcon:target uri="sip:carol@example.com"
                                 method="dial-out"/>
                  </xcon:allowed-users-list>
              </info:users>
                 <xcon:floor-information>
                    <xcon:floor-request-handling>
                       confirm</xcon:floor-request-handling>
                    <xcon:conference-floor-policy>
                      <xcon:floor id="1">
                       <xcon:media-label>11</xcon:media-label>
                      </xcon:floor>
                    </xcon:conference-floor-policy>
                 </xcon:floor-information>
            </confInfo>
         </ccmp:confResponse>
     </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 10: Create Basic Conference (Clone) Detailed Messaging

6. Conference Users Scenarios and Examples

Section 5 showed examples describing the several different ways a conference might be created using CCMP. This section focuses on user-related scenarios, i.e., typical scenarios that may occur during

the lifetime of a conference, like adding new users and handling their media. The following scenarios are based on those documented in the XCON framework. The examples assume that a conference has already been correctly established, with media, if applicable, per one of the examples in Section 5.

6.1. Adding a Party

In this example, Alice wants to add Bob to an established conference. In the following example we assume Bob is a new user of the system, which means Alice also needs to provide some details about him. In fact, the case of Bob already present as a user in the conferencing system is much easier to address, and will be discussed later.

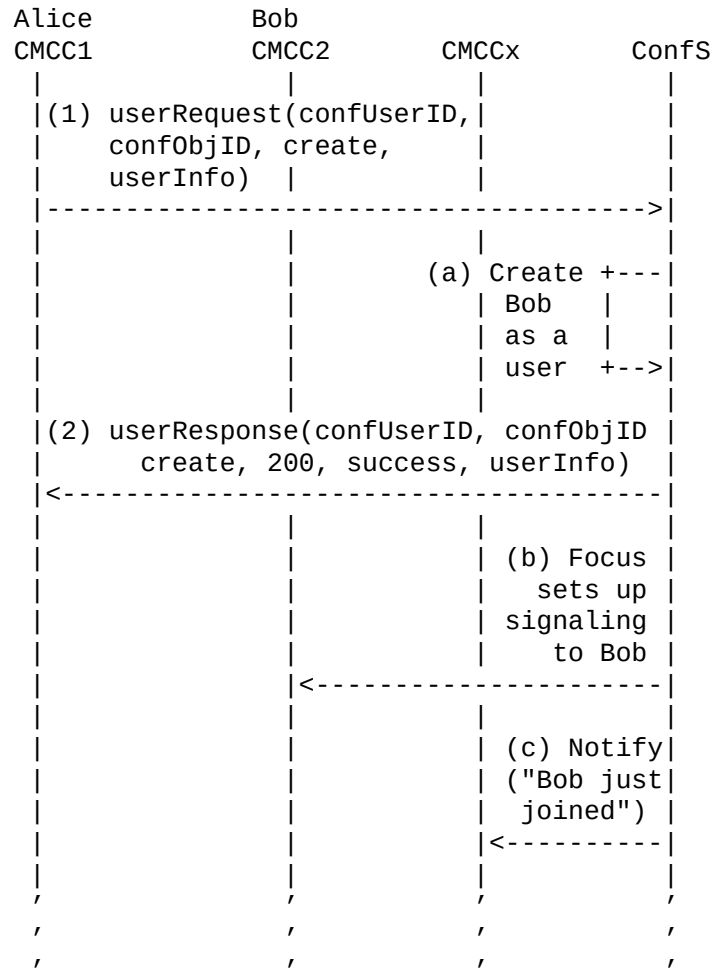


Figure 11: Client Manipulation of Conference - Add a Party

1. Alice sends a userRequest message with an operation of "create" to add Bob to the specific conference as identified by the XCON-URI in the <confObjID>. The "create" <operation> also makes sure that Bob is created as a user in the whole conferencing system. This is done by adding in the request a <userInfo> element describing Bob as a user. This is needed in order to let the conferencing system be aware of Bob's characteristics. In case Bob was already a registered user, Alice would just have referenced him through his XCON-USERID in the 'entity' attribute of the <userInfo> field, without providing additional data. In fact, that data (including, for instance, Bob's SIP-URI to be used subsequently for dial-out) would be obtained by referencing the extant registration. The conference server ensures that Alice has the appropriate authority based on the policies associated with that specific conference object to perform the operation. As mentioned before, a new XCON-USERID is created for Bob, and the <userInfo> is used to update the conference object accordingly. As already seen in Section 5.3, a placeholder wildcard ("xcon-userid:AUTO_GENERATE_1@example.com" in the CCMP messages below) is used for the 'entity' attribute of the <userInfo> element, to be replaced by the actual XCON-USERID later;

2. Bob is successfully added to the conference object, and an XCON-USERID is allocated for him ("xcon-userid:Bob@example.com"); this identifier is reported in the response as the value of the 'entity' attribute of the returned <userInfo>;

3. In the presented example, the call signaling to add Bob to the conference is instigated through the focus as well. As noted previously, this is implementation specific. In fact, a conferencing system may accomplish different actions after the user creation, just as it may do nothing at all. Among the possible actions, for instance, Bob may be added as a <target> element to the <allowed-users-list> element, whose joining 'method' may be either "dial-in" or "dial-out". Besides, out-of-band notification mechanisms may be involved as well, e.g., to notify Bob via mail of the new conference, including details as the date, password, expected participants, and so on (see Section 4.3).

Once Bob has been successfully added to the specified conference, per updates to the state, and depending upon the policies, other participants (including Bob himself) may be notified of the addition of Bob to the conference via the conference notification service in use.

1. userRequest/create message (Alice adds Bob)

```
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpRequest xmlns:info="urn:ietf:params:xml:ns:conference-info"
        xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
        xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
    <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:type="ccmp:ccmp-user-request-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977878@example.com</confObjID>
        <operation>create</operation>
        <ccmp:userRequest>
          <userInfo entity="xcon-userid:AUTO_GENERATE_1@example.com">
              <info:display-text>Bob</info:display-text>
              <info:associated-aors>
                  <info:entry>
                    <info:uri>mailto:bob.depippis@example.com</info:uri>
                    <info:display-text>Bob's email</info:display-text>
                  </info:entry>
              </info:associated-aors>
              <info:endpoint entity="sip:bob83@example.com">
                  <info:display-text>Bob's laptop</info:display-text>
              </info:endpoint>
          </userInfo>
        </ccmp:userRequest>
    </ccmpRequest>
  </ccmp:ccmpRequest>
```

2. userResponse/create message (a new XCON-USERID is created for Bob and he is added to the conference)

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpResponse xmlns:info="urn:ietf:params:xml:ns:conference-info"
         xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
         xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
    <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="ccmp:ccmp-user-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977878@example.com</confObjID>
        <operation>create</operation>
        <response-code>200</response-code>
        <response-string>success</response-string>
        <version>10</version>
        <ccmp:userResponse>
          <userInfo entity="xcon-userid:Bob@example.com">
              <info:display-text>Bob</info:display-text>
              <info:associated-aors>
                  <info:entry>
```

```
                    <info:uri>mailto:bob.depippis@example.com</info:uri>
                    <info:display-text>Bob's email</info:display-text>
                </info:entry>
            </info:associated-aors>
            <info:endpoint entity="sip:bob83@example.com">
                <info:display-text>Bob's laptop</info:display-text>
            </info:endpoint>
        </userInfo>
      </ccmp:userResponse>
   </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 12: Add Party Message Details

6.2. Muting a Party

This section provides an example of the muting of a party in an active conference. We assume that the user to mute has already been added to the conference. The document only addresses muting and not unmuting, since the latter would involve an almost identical CCMP message flow anyway. However, if any external floor control is involved, whether a particular conferencing client can actually mute/unmute itself must be considered by the conferencing system.

> Please notice that interaction between CCMP and floor control should be carefully considered. In fact, handling CCMP- and BFCP-based media control has to be considered as multiple layers: that is, a participant may have the BFCP floor granted, but be muted by means of CCMP. If so, he would still be muted in the conference, and would only be unmuted if both the protocols allowed for this.

Figure 13 provides an example of one client, Alice, impacting the media state of another client, Bob. This example assumes an established conference. In this example, Alice, who is the moderator of the conference, wants to mute Bob on a medium-sized multi-party conference, as his device is not muted (and he's obviously not listening to the call) and background noise in his office environment is disruptive to the conference. BFCP floor control is assumed not to be involved.

Muting can be accomplished using the <mute> control element associated with the target user's audio, in which case the conference server must update the settings associated with the user's media streams. Muting/unmuting can also be accomplished by directly modifying the settings related to the target user's media streams, which is the approach shown in this example. Specifically, Bob's <userInfo> is updated by modifying the <endpoint> element in the

<media> part related to audio information, identified by the 'id' attribute. The modification consists in setting the audio <status> to "recvonly", in case of muting.

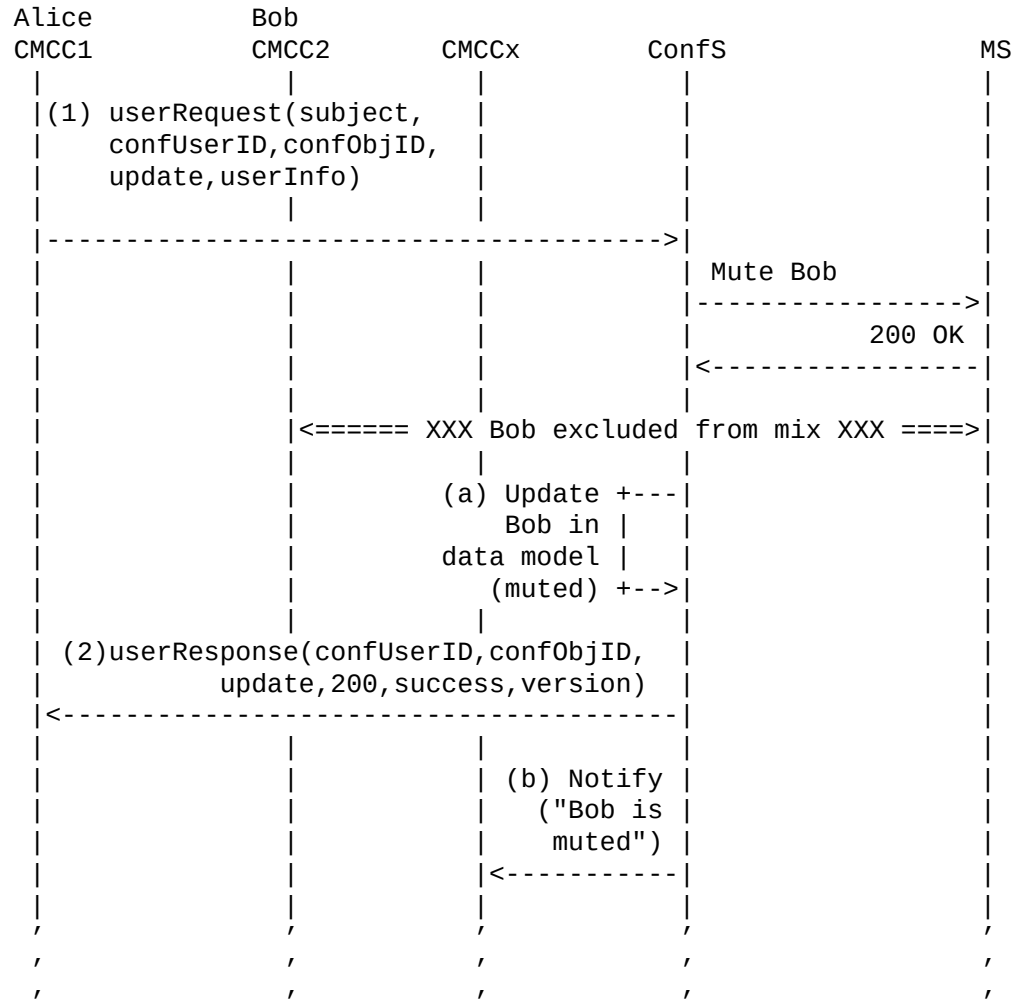


Figure 13: Client Manipulation of Conference - Mute a Party

1. Alice sends a userRequest message with an "update" <operation> and the <userInfo> with the <status> field in the <media> element for Bob's <endpoint> set to "revconly". In order to authenticate herself, Alice provides in the <subject> request parameter her registration credentials (i.e., username and password). The <subject> parameter is an optional one: its use can be systematic whenever the conference server envisages to authenticate each

requester. In such cases, if the client does not provide the required authentication information, the conferencing server answers with a CCMP "authenticationRequired" <response-code>, indicating that the request cannot be processed without including the proper <subject> parameter. The conference server ensures that Alice has the appropriate authority based on the policies associated with that specific conference object to perform the operation. It recognizes that Alice is allowed to request the specified modification, since she is moderator of the target conference, and updates the <userInfo> in the conference object reflecting that Bob's media is not to be mixed with the conference media. If the conference server relies on a remote media server for its multimedia functionality, it subsequently changes Bob's media profile accordingly by means of the related protocol interaction with the MS. An example describing a possible way of dealing with such a situation using the media server control architecture [RFC5567] is described in Figure 31, "Simple Bridging: Framework Transactions (2)", in [CALL-FLOWS].

2. A userResponse message with a "200" <response-code> ("success") is then sent to Alice. Depending upon the policies, the conference server may notify other participants (including Bob) of this update via any conference notification service that may be in use.

1. userRequest/update message (Alice mutes Bob)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest xmlns:info="urn:ietf:params:xml:ns:conference-info"
      xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
      xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
  <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:type="ccmp:ccmp-user-request-message-type">
     <subject>
        <username>Alice83</username>
        <conference-password>13011983</conference-password>
      </subject>
      <confUserID>xcon-userid:Alice@example.com</confUserID>
      <confObjID>xcon:8977878@example.com</confObjID>
      <operation>update</operation>
      <ccmp:userRequest>
          <userInfo entity="xcon-userid:Bob@example.com">
              <info:endpoint entity="sip:bob83@example.com">
                  <info:media id="1">
                      <info:label>123</info:label>
                      <info:status>recvonly</info:status>
                  </info:media>
              </info:endpoint>
```

```
            </userInfo>
        </ccmp:userRequest>
    </ccmpRequest>
  </ccmp:ccmpRequest>
```

2. userResponse/update message (Bob has been muted)

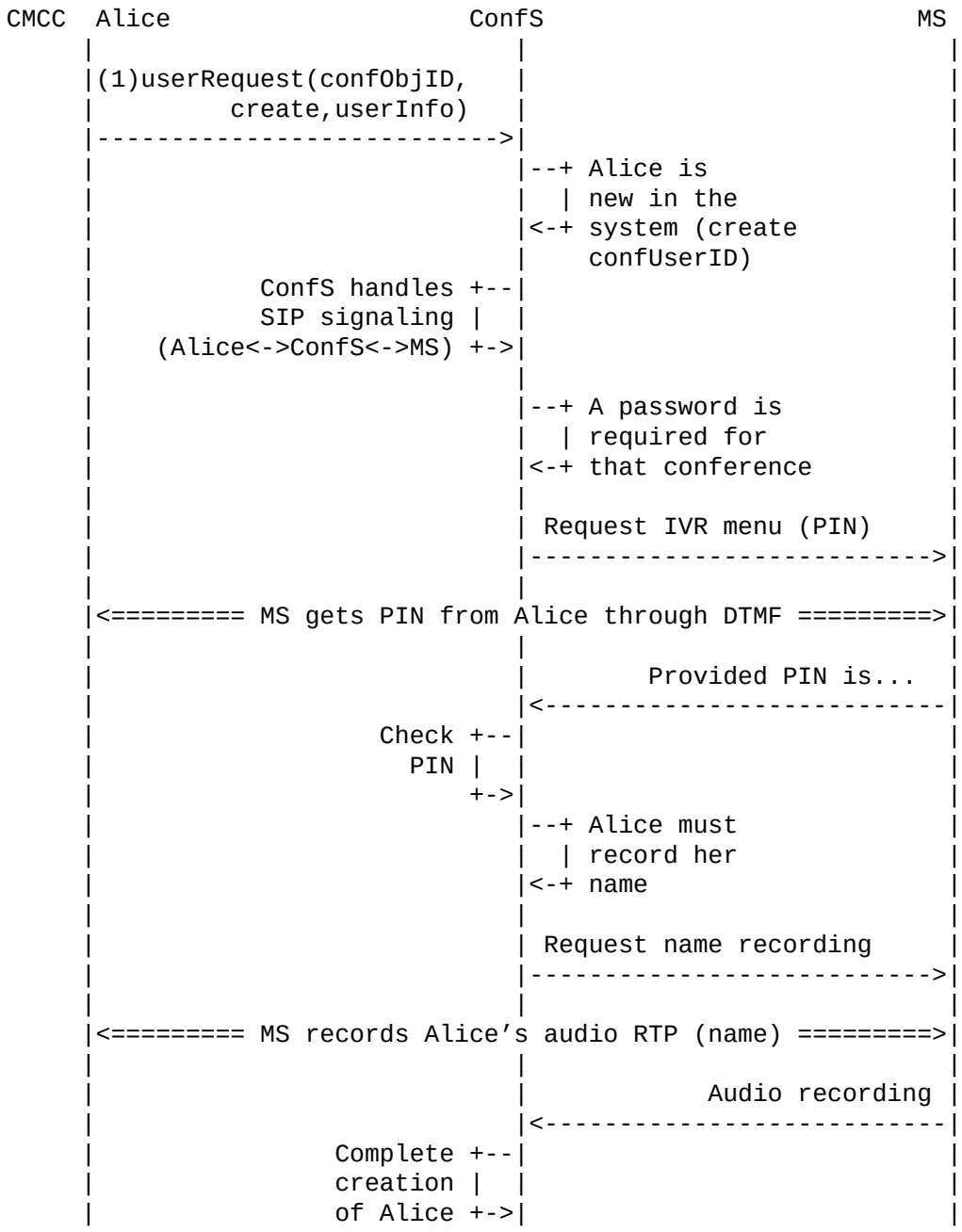
```
  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpResponse xmlns:info="urn:ietf:params:xml:ns:conference-info"
                xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
                xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
    <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:type="ccmp:ccmp-user-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977878@example.com</confObjID>
        <operation>update</operation>
        <response-code>200</response-code>
        <response-string>success</response-string>
            <version>7</version>
        <ccmp:userResponse/>
    </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 14: Mute Message Details

6.3. Conference Announcements and Recordings

This section deals with features that are typically required in a conferencing system, such as public announcements (e.g., to notify vocally that a new user joined a conference) and name recording. While this is not strictly CCMP-related (the CCMP signaling is actually the same as the one seen in Section 6.1), it is an interesting scenario to address to see how several components of an XCON-compliant architecture interact with each other to make it happen.

In this example, as shown in Figure 15, Alice is joining Bob's conference that requires that she first enter a passcode. After successfully entering the passcode, an announcement prompts Alice to speak her name so it can be recorded. When Alice is added to the active conference, the recording is played back to all the existing participants. A very similar example is presented in Figure 33 of [CALL-FLOWS].


CMCC
Alice
ConfS
MS
(1)userRequest(confObjID, create,userInfo)
--+ Alice is new in the
<-+ system (create confUserID)
ConfS handles SIP signaling (Alice<->ConfS<->MS)
--+ A password is required for
<-+ that conference
Request IVR menu (PIN)
<========= MS gets PIN from Alice through DTMF =========>
Provided PIN is...
Check PIN
--+ Alice must record her
<-+ name
Request name recording
<========= MS records Alice's audio RTP (name) =========>
Audio recording
Complete creation of Alice

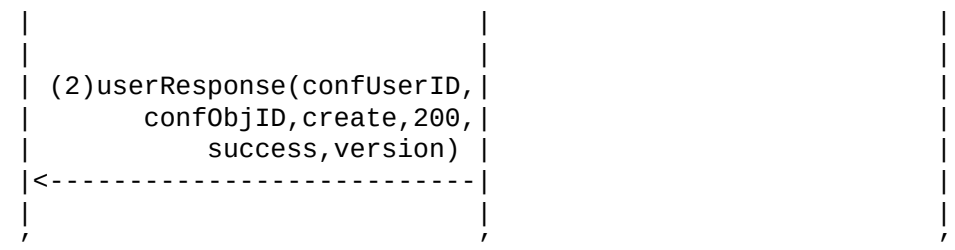


Figure 15: Recording and Announcements

1. Upon receipt of the userRequest message from Alice to be added to Bob's conference, the conference server determines that a password is required for this specific conference. Thus, an announcement asking Alice to enter the password is sent back. This may be achieved by means of typical IVR functionality. Once Alice enters the password, it is validated against the policies associated with Bob's active conference. The conference server then connects to a server that prompts and records Alice's name. The conference server must also determine whether Alice is already a user of this conferencing system or whether she is a new user. In this case, Alice is a new user for this conferencing system, so a new XCON-USERID is created for Alice. Based upon the contact information provided by Alice, the call signaling to add Alice to the conference is instigated through the focus.

2. The conference server sends Alice a userResponse message that includes in the <confUserID> the XCON-USERID assigned by the conferencing system to her. This would allow Alice to later perform operations on the conference (if she were to have the appropriate policies), including registering for event notifications associated with the conference. Once the call signaling indicates that Alice has been successfully added to the specific conference, per updates to the state, and depending upon the policies, other participants (e.g., Bob) are notified of the addition of Alice to the conference via the conference notification service and an announcement is provided to all the participants indicating that Alice has joined the conference.

1. userRequest/create message (a new conferencing system client, Alice, enters Bob's conference)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest
     xmlns:info="urn:ietf:params:xml:ns:conference-info"
           xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
           xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
```

```
      <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:type="ccmp:ccmp-user-request-message-type">
          <confObjID>xcon:bobConf@example.com</confObjID>
          <operation>create</operation>
          <ccmp:userRequest>
            <userInfo entity="xcon-userid:AUTO_GENERATE_1@example.com">
                  <info:associated-aors>
                      <info:entry>
                          <info:uri>
                             mailto:Alice83@example.com
                          </info:uri>
                          <info:display-text>email</info:display-text>
                      </info:entry>
                  </info:associated-aors>
                  <info:endpoint entity="sip:alice_789@example.com"/>
              </userInfo>
          </ccmp:userRequest>
      </ccmpRequest>
  </ccmp:ccmpRequest>
```

2. userResponse/create message (Alice provided with a new XCON-USERID and added to the conference)

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpResponse
      xmlns:info="urn:ietf:params:xml:ns:conference-info"
      xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
      xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
     <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xsi:type="ccmp:ccmp-user-response-message-type">
          <confUserID>xcon-userid:Alice@example.com</confUserID>
          <confObjID>xcon:bobConf@example.com</confObjID>
          <operation>create</operation>
          <response-code>200</response-code>
          <response-string>success</response-string>
          <version>5</version>
          <ccmp:userResponse/>
      </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 16: Announcement Messaging Details

6.4. Monitoring for DTMF

Conferencing systems also often need the capability to monitor for dual-tone multi-frequency (DTMF) from each individual participant. This would typically be used to enter the identifier and/or access code for joining a specific conference. This feature is also often

exploited to achieve interaction between participants and the conferencing system for non-XCON-aware user agents (e.g., using DTMF tones to get muted/unmuted).

An example of DTMF monitoring, within the context of the framework elements, is shown in Figure 15. The media control architecture and protocols [RFC5567] can be used by the conference server for all the DTMF interactions. Examples for DTMF interception in conference instances are presented in [CALL-FLOWS].

6.5. Entering a Password-Protected Conference

Some conferences may require a password to be provided by a user who wants to manipulate the conference objects (e.g., join, update, delete) via CCMP. In this case, a password would be included in the <conference-password> element in the appropriate <conference-uris> entry of the conference data model. Such password must be then included in the <conference-password> field in the CCMP request addressed to that conference.

In the following example, Alice, a conferencing system client, attempts to join a password-protected conference.

1. Alice sends a userRequest message with a "create" <operation> to add herself in the conference with XCON-URI "xcon:8977777@example.com" (written in the <confObjID> parameter). Alice provides her XCON-USERID via the <confUserID> field of the userRequest message and leaves out the <userInfo> one (first-party join). In this first attempt, she doesn't insert any password parameter.

2. Upon receipt the userRequest/create message, the conference server detects that the indicated conference is not joinable without providing the appropriate passcode. A userResponse message with a "423" <response-code> ("conference password required") is returned to Alice to indicate that her join has been refused and that she has to resend her request including the appropriate conference password in order to participate.

3. After getting the passcode through out-of-band mechanisms, Alice provides it in the proper <conference-password> request field of a new userRequest/create message and sends the updated request back to the server.

4. The conference server checks the provided password and then adds Alice to the protected conference. After that, a userResponse message with a "200" <response-code> ("success") is sent to Alice.

1. userRequest/create message (Alice tries to enter the conference without providing the password)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest
     xmlns:info="urn:ietf:params:xml:ns:conference-info"
           xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
           xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
    <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="ccmp:ccmp-user-request-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
        <operation>create</operation>
        <ccmp:userRequest/>
    </ccmpRequest>
</ccmp:ccmpRequest>
```

2. userResponse/create message ("423", "conference password required")

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
    xmlns:info="urn:ietf:params:xml:ns:conference-info"
    xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
    xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
   <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:type="ccmp:ccmp-user-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
        <operation>create</operation>
        <response-code>423</response-code>
        <response-string>conference password required</response-string>
        <ccmp:userResponse/>
    </ccmpResponse>
</ccmp:ccmpResponse>
```

3. userRequest/create message (Alice provides the password)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest
     xmlns:info="urn:ietf:params:xml:ns:conference-info"
           xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
           xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
    <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="ccmp:ccmp-user-request-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
        <operation>create</operation>
        <conference-password>8601</conference-password>
```

```
        <ccmp:userRequest/>
    </ccmpRequest>
</ccmp:ccmpRequest>
```

4. userResponse/create message
   (Alice has been added to the conference)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
    xmlns:info="urn:ietf:params:xml:ns:conference-info"
    xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
    xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
   <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:type="ccmp:ccmp-user-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
        <operation>create</operation>
        <response-code>200</response-code>
        <response-string>success</response-string>
        <version>10</version>
        <ccmp:userResponse/>
    </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 17: Password-Protected Conference Join Messages Details

7. Sidebars Scenarios and Examples

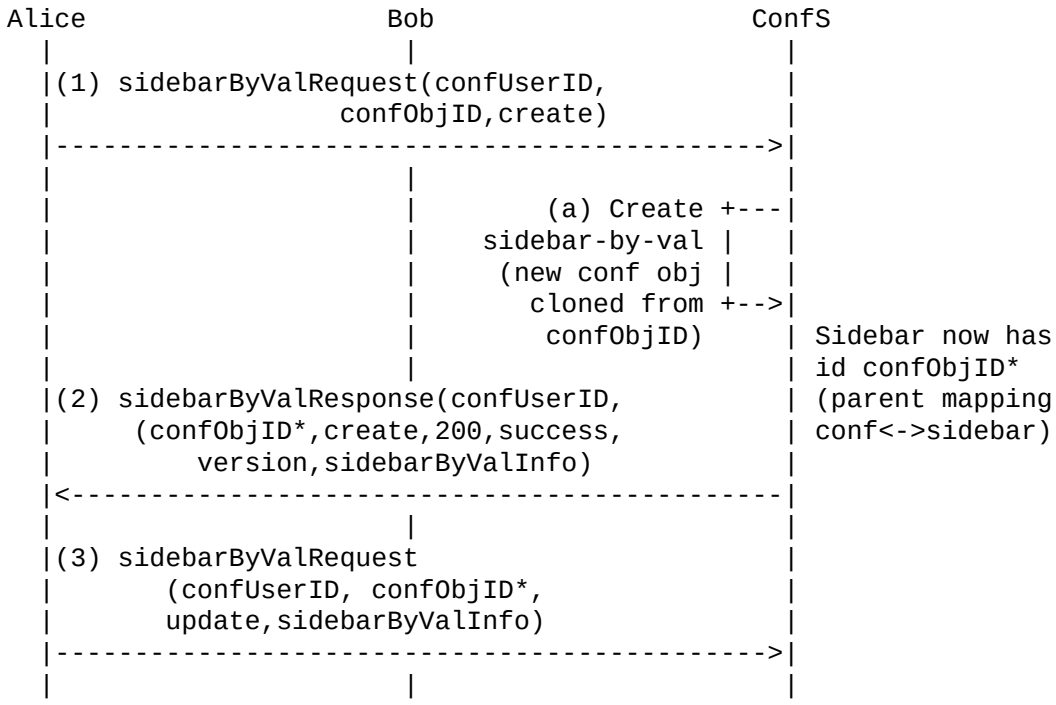
While creating conferences and manipulating users and their media are sufficient for many scenarios, there may be cases when more complex management is needed.

In fact, a feature typically required in conferencing systems is the ability to create sidebars. A sidebar is basically a child conference that usually includes a subset of the participants of the parent conference and a subset of its media as well. Sidebars are typically required whenever some of the participants in a conference want a private discussion, without interfering with the main conference.

This section deals with some typical scenarios using a sidebar, like whispering, private messaging, and coaching scenarios. The first subsections present some examples of how a generic sidebar can be created, configured, and managed.

7.1. Internal Sidebar

Figure 18 provides an example of one client, Alice, involved in an active conference with Bob and Carol. Alice wants to create a sidebar to have a side discussion with Bob while still viewing the video associated with the main conference. Alternatively, the audio from the main conference could be maintained at a reduced volume. Alice initiates the sidebar by sending a request to the ConfS to create a conference reservation based upon the active conference object. Alice and Bob would remain on the roster of the main conference, such that other participants could be aware of their participation in the main conference, while an internal-sidebar conference is occurring. Besides, Bob decides that he is not interested in still receiving the conference audio in background (not even at a lower volume as Alice configured) and so modifies the sidebar in order to make that stream inactive for him.



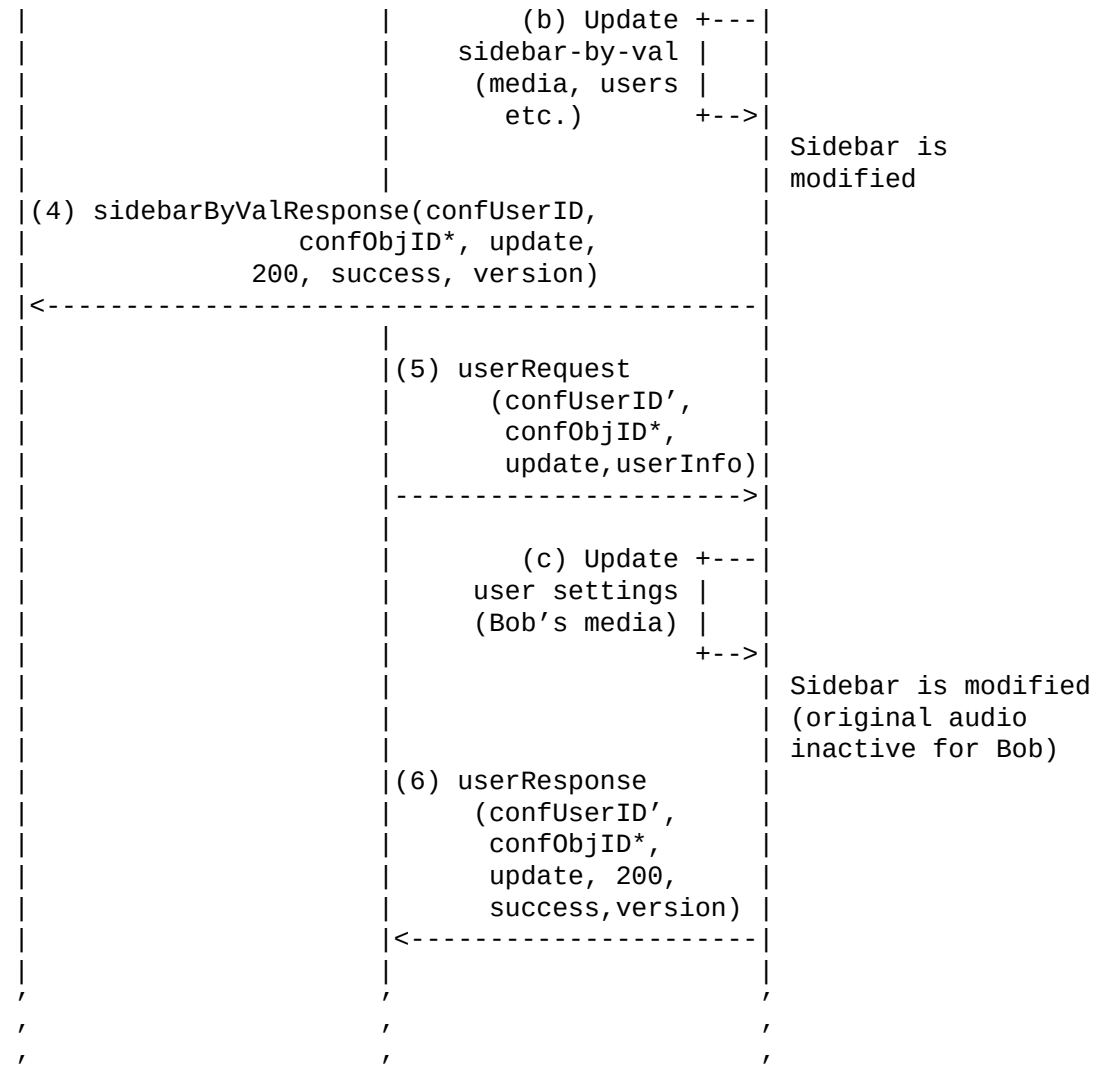


Figure 18: Client Creation of a Sidebar Conference

1. Upon receipt of CCMP sidebarByValRequest message to create a new sidebar based upon the conference whose XCON-URI is in the <confObjID> received in the request, the conference server uses such XCON-URI to clone a conference reservation for the sidebar. The sidebar reservation is NOT independent of the active main conference (i.e., parent). The conference server also allocates a new XCON-URI ("confObjID*" in Figure 18) for that sidebar to be used for any subsequent protocol requests from any of the members of the conference. The new XCON-URI is returned in the response message <confObjID> parameter.

2. The relationship information is provided in the sidebarByValResponse message, specifically in the <sidebar-parent> element. A dump of the complete representation of the main/parent conference is provided below as well to show how the cloning process for the creation of the sidebar could take place.

3. Upon receipt of the sidebarByValResponse message to reserve the conference, Alice can now create an active conference using that reservation or create additional reservations based upon the existing reservations. In this example, Alice wants only Bob to be involved in the sidebar; thus, she manipulates the membership so that only the two of them appear in the <allowed-users-list> section. Alice also wants both audio and video from the original conference to be available in the sidebar. For what concerns the media belonging to the sidebar itself, Alice wants the audio to be restricted to the participants in the sidebar (that is, Bob and herself). Additionally, Alice manipulates the media values to receive the audio from the main conference at a reduced volume, so that the communication between her and Bob isn't affected. Alice sends a sidebarByValRequest message with an operation of "update" along with the <sidebarByValInfo> containing the aforementioned sidebar modifications.

4. Upon receipt of the sidebarByValRequest message to update the sidebar reservation, the conference server ensures that Alice has the appropriate authority based on the policies associated with that specific conference object to perform the operation. The conference server must also validate the updated information in the reservation, ensuring that a member like Bob is already a user of this conference server. Once the data for the conference identified by the <confObjID> is updated, the conference server sends a sidebarByValResponse message to Alice. Depending upon the policies, the initiator of the request (i.e., Alice) and the participants in the sidebar (i.e., Bob) may be notified of his addition to the sidebar via the conference notification service.

5. At this point, Bob sends a userRequest message to the conference server with an operation of "update" to completely disable the background audio from the parent conference, since it prevents him from understanding what Alice says in the sidebar.

6. Notice that Bob's request only changes the media perspective for Bob. Alice keeps on receiving both the audio from Bob and the background from the parent conference. This request may be relayed by the conference server to the media server handling the mixing, if present. Upon completion of the change, the

conference server sends a userResponse message to Bob. Depending upon the policies, the initiator of the request (i.e., Bob) and the participants in the sidebar (i.e., Alice) may be notified of this change via the conference notification service.

The following conference object represents the conference in which the sidebar is to be created. It will be used by the conference server to create the new conference object associated with the sidebar.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <info:conference-info
                xmlns:info="urn:ietf:params:xml:ns:conference-info"
                xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
                xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
                entity="xcon:8977878@example.com">
     <info:conference-description>
        <info:display-text>MAIN CONFERENCE</info:display-text>
        <info:conf-uris>
            <info:entry>
               <info:uri>sip:8977878@example.com</info:uri>
               <info:display-text>conference sip uri</info:display-text>
            </info:entry>
        </info:conf-uris>
        <info:available-media>
          <info:entry label="123">
            <info:display-text>main conference audio</info:display-text>
            <info:type>audio</info:type>
            <info:status>sendrecv</info:status>
          </info:entry>
          <info:entry label="456">
            <info:display-text>main conference video</info:display-text>
            <info:type>video</info:type>
            <info:status>sendrecv</info:status>
            <xcon:controls>
                    <xcon:video-layout>single-view</xcon:video-layout>
           </xcon:controls>
          </info:entry>
        </info:available-media>
    </info:conference-description>
    <info:conference-state>
        <info:active>true</info:active>
    </info:conference-state>
    <info:users>
        <info:user entity="xcon-userid:Alice@example.com">
            <info:display-text>Alice</info:display-text>
            <info:endpoint entity="sip:Alice@example.com">
                <info:media id="1">
```

```
                    <info:label>123</info:label>
                    <info:status>sendrecv</info:status>
                </info:media>
                <info:media id="2">
                    <info:label>456</info:label>
                    <info:status>sendrecv</info:status>
                </info:media>
            </info:endpoint>
        </info:user>
        <info:user entity="xcon-userid:Bob@example.com">
            <info:display-text>Bob</info:display-text>
            <info:endpoint entity="sip:bob83@example.com">
                <info:media id="1">
                    <info:label>123</info:label>
                    <info:status>sendrecv</info:status>
                </info:media>
                <info:media id="2">
                    <info:label>456</info:label>
                    <info:status>sendrecv</info:status>
                </info:media>
            </info:endpoint>
        </info:user>
        <info:user entity="xcon-userid:Carol@example.com">
            <info:display-text>Carol</info:display-text>
            <info:endpoint entity="sip:carol@example.com">
                <info:media id="1">
                    <info:label>123</info:label>
                    <info:status>sendrecv</info:status>
                </info:media>
                <info:media id="2">
                    <info:label>456</info:label>
                    <info:status>sendrecv</info:status>
                </info:media>
            </info:endpoint>
        </info:user>
    </info:users>
</info:conference-info>
```

Figure 19: Conference with Alice, Bob, and Carol

The sidebar creation happens through a cloning of the parent conference. Once the sidebar is created, an update request makes sure that the sidebar is customized as needed. The following protocol dump makes the process clearer.

1. sidebarByValRequest/create message (Alice creates an internal sidebar)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest xmlns:info="urn:ietf:params:xml:ns:conference-info"
             xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
             xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
  <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="ccmp:ccmp-sidebarByVal-request-message-type">
      <confUserID>xcon-userid:Alice@example.com</confUserID>
      <confObjID>xcon:8977878@example.com</confObjID>
      <operation>create</operation>
      <ccmp:sidebarByValRequest/>
  </ccmpRequest>
</ccmp:ccmpRequest>
```

2. sidebarByValResponse/create message (sidebar returned)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
            xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
            xmlns:info="urn:ietf:params:xml:ns:conference-info"
            xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
  <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:type="ccmp:ccmp-sidebarByVal-response-message-type">
      <confUserID>xcon-userid:Alice@example.com</confUserID>
      <confObjID>xcon:8974545@example.com</confObjID>
      <operation>create</operation>
      <response-code>200</response-code>
      <response-string>success</response-string>
          <version>1</version>
      <ccmp:sidebarByValResponse>
          <sidebarByValInfo entity="xcon:8974545@example.com">
              <info:conference-description>
                  <info:display-text>
                       SIDEBAR CONFERENCE registered by Alice
                  </info:display-text>
                  <info:available-media>
                      <info:entry label="123">
                          <info:display-text>
                                main conference audio
                          </info:display-text>
                          <info:type>audio</info:type>
                          <info:status>sendrecv</info:status>
                      </info:entry>
                      <info:entry label="456">
                          <info:display-text>
                                main conference video
```

```
                            </info:display-text>
                            <info:type>video</info:type>
                            <info:status>sendrecv</info:status>
                        </info:entry>
                    </info:available-media>
                </info:conference-description>
                <info:conference-state>
                    <info:active>false</info:active>
                </info:conference-state>
                <info:users>
                    <xcon:allowed-users-list>
                        <xcon:target method="dial-in"
                              uri="xcon-userid:Alice@example.com"/>
                        <xcon:target method="dial-in"
                              uri="xcon-userid:Bob@example.com"/>
                        <xcon:target method="dial-in"
                              uri="xcon-userid:Carol@example.com"/>
                    </xcon:allowed-users-list>
                    <xcon:sidebar-parent>
                         xcon:8977878@example.com
                    </xcon:sidebar-parent>
                </info:users>
            </sidebarByValInfo>
        </ccmp:sidebarByValResponse>
    </ccmpResponse>
  </ccmp:ccmpResponse>
```

3. sidebarByValRequest/update message (Alice updates the created sidebar)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpRequest
            xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
            xmlns:info="urn:ietf:params:xml:ns:conference-info"
            xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
    <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:type="ccmp:ccmp-sidebarByVal-request-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8974545@example.com</confObjID>
        <operation>update</operation>
        <ccmp:sidebarByValRequest>
            <sidebarByValInfo entity="xcon:8974545@example.com">
                <info:conference-description>
                  <info:display-text>
                        private sidebar Alice - Bob
                  </info:display-text>
                  <info:available-media>
                        <info:entry label="123">
```

```
                        <info:display-text>
                            main conference audio
                        </info:display-text>
                        <info:type>audio</info:type>
                        <info:status>recvonly</info:status>
                        <xcon:controls>
                            <xcon:gain>-60</xcon:gain>
                        </xcon:controls>
                    </info:entry>
                    <info:entry label="456">
                        <info:display-text>
                            main conference video
                        </info:display-text>
                        <info:type>video</info:type>
                        <info:status>recvonly</info:status>
                    </info:entry>
                    <info:entry label="AUTO_GENERATE_1">
                        <info:display-text>
                            sidebar audio
                        </info:display-text>
                        <info:type>audio</info:type>
                        <info:status>sendrecv</info:status>
                    </info:entry>
                    <info:entry label="AUTO_GENERATE_2">
                        <info:display-text>
                            sidebar video
                        </info:display-text>
                        <info:type>video</info:type>
                        <info:status>sendrecv</info:status>
                    </info:entry>
                </info:available-media>
            </info:conference-description>
            <info:users>
                <xcon:allowed-users-list>
                    <xcon:target method="dial-out"
                          uri="xcon-userid:Alice@example.com"/>
                    <xcon:target method="dial-out"
                          uri="xcon-userid:Bob@example.com"/>
                </xcon:allowed-users-list>
            </info:users>
        </sidebarByValInfo>
    </ccmp:sidebarByValRequest>
</ccmpRequest>
</ccmp:ccmpRequest>
```

4. sidebarByValResponse/update message (sidebar's updates accepted)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
              xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
              xmlns:info="urn:ietf:params:xml:ns:conference-info"
              xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
  <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:type="ccmp:ccmp-sidebarByVal-response-message-type">
      <confUserID>xcon-userid:Alice@example.com</confUserID>
      <confObjID>xcon:8974545@example.com</confObjID>
      <operation>update</operation>
      <response-code>200</response-code>
      <response-string>success</response-string>
          <version>2</version>
      <ccmp:sidebarByValResponse/>
  </ccmpResponse>
</ccmp:ccmpResponse>
```

5. userRequest/update message (Bob updates his media)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest
         xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
         xmlns:info="urn:ietf:params:xml:ns:conference-info"
         xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
    <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:type="ccmp:ccmp-user-request-message-type">
      <confUserID>xcon-userid:Bob@example.com</confUserID>
      <confObjID>xcon:8974545@example.com</confObjID>
      <operation>update</operation>
      <ccmp:userRequest>
          <userInfo entity="xcon-userid:Bob@example.com">
              <info:endpoint entity="sip:bob83@example.com">
                  <info:media id="1">
                      <info:display-text>
                          main conference audio
                      </info:display-text>
                      <info:label>123</info:label>
                      <info:status>inactive</info:status>
                  </info:media>
              </info:endpoint>
          </userInfo>
      </ccmp:userRequest>
  </ccmpRequest>
</ccmp:ccmpRequest>
```

6. userResponse/update message (Bob's preferences are set)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse xmlns:info="urn:ietf:params:xml:ns:conference-info"
             xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
             xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
  <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:type="ccmp:ccmp-user-response-message-type">
      <confUserID>xcon-userid:Bob@example.com</confUserID>
      <confObjID>xcon:8974545@example.com</confObjID>
      <operation>update</operation>
      <response-code>200</response-code>
      <response-string>success</response-string>
      <version>3</version>
      <ccmp:userResponse/>
  </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 20: Internal Sidebar Messaging Details

7.2. External Sidebar

Figure 21 provides an example of a different approach towards sidebars. In this scenario, one client, Alice, is involved in an active conference with Bob, Carol, David, and Ethel. Alice gets an important text message via a whisper from Bob that a critical customer needs to talk to Alice, Bob, and Ethel. Alice creates a sidebar to have a side discussion with the customer Fred including the participants in the current conference with the exception of Carol and David, who remain in the active conference. The difference from the previous scenario is that Fred is not part of the parent conference: this means that different policies might be involved, considering that Fred may access information coming from the parent conference, in case the sidebar was configured accordingly. For this reason, in this scenario, we assume that Alice disables all the media from the original (parent) conference within the sidebar. This means that, while in the previous example Alice and Bob still heard the audio from the main conference in background, this time no background is made available. Alice initiates the sidebar by sending a request to the conference server to create a conference reservation based upon the active conference object. Alice, Bob and Ethel would remain on the roster of the main conference in a hold state. Whether or not the hold state of these participants is visible to other participants depends upon the individual and local policy. However, providing the hold state allows the participants in the main conference to see that others in the conference are busy. Note, that a separate conference could have been created by Alice to allow Bob and Ethel to talk to Fred. However, creating a sidebar has somewhat of an advantage by

allowing the conference to be created using some of the same settings (e.g., role, floor control, etc.) that Bob and Ethel had in the main conference and it would allow for updates such that the media could be updated, for example, to provide audio from the main conference.

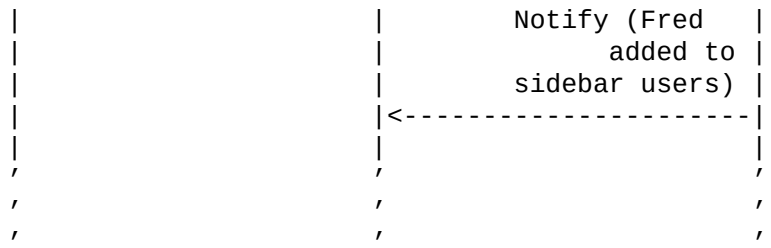


Figure 21: Client Creation of an External Sidebar

1. Upon receipt of the sidebarByRefRequest message to create a new sidebar conference, based upon the active conference specified by <confObjID> in the request, the conference server uses that active conference to clone a conference reservation for the sidebar. The sidebar reservation is NOT independent of the active conference (i.e., parent). The conference server, as before, allocates a new XCON-URI ("confObjID*" in Figure 21) to be used for any subsequent protocol requests toward the sidebar reservation. The mapping between the sidebar XCON-URI and the one associated with the main conference is maintained by the conference server and it is gathered from the <sidebar-parent> element in the sidebar conference object.

2. Upon receipt of the sidebarByRefResponse message, which acknowledges the successful creation of the sidebar object, Alice decides that only Bob and Ethel, along with the new participant Fred are to be involved in the sidebar. Thus, she manipulates the membership accordingly. Alice also sets the media in the <conference-info> such that the participants in the sidebar don't receive any media from the main conference. All these settings are provided to the conferencing server by means of a new sidebarByRefRequest message, with an "update" <operation>.

3. Alice sends the aforementioned sidebarByRefRequest message to update the information in the reservation and to create an active conference. Upon receipt of the sidebarByRefRequest/update message, the conference server ensures that Alice has the appropriate authority based on the policies associated with that specific conference object to perform the operation. The conference server also validates the updated information in the reservation. Since Fred is a new user for this conferencing system, a conference user identifier (XCON-USERID) is created for Fred. Specifically, Fred is added to the conference by only providing his SIP URI. Based upon the contact information provided for Fred by Alice, the call signaling to add Fred to the conference may be instigated through the focus (e.g., if Fred had

a "dial-out" value for the 'method' attribute in his <target> field under <allowed-users-list>) at the actual activation of the sidebar.

4. The conference server sends a sidebarByRefResponse message and, depending upon the policies, the initiator of the request (i.e., Alice) and the participants in the sidebar (i.e., Bob and Ethel) may be notified of his addition to the sidebar via the conference notification service.

1. sidebarByRefRequest/create message (Alice creates an external sidebar)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest xmlns:info="urn:ietf:params:xml:ns:conference-info"
             xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
             xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
  <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:type="ccmp:ccmp-sidebarByRef-request-message-type">
      <confUserID>xcon-userid:Alice@example.com</confUserID>
      <confObjID>xcon:8977878@example.com</confObjID>
      <operation>create</operation>
      <ccmp:sidebarByRefRequest/>
  </ccmpRequest>
</ccmp:ccmpRequest>
```

2. sidebarByRefResponse/create message (created sidebar returned)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
              xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
              xmlns:info="urn:ietf:params:xml:ns:conference-info"
              xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
  <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:type="ccmp:ccmp-sidebarByRef-response-message-type">
      <confUserID>xcon-userid:Alice@example.com</confUserID>
      <confObjID>xcon:8971212@example.com</confObjID>
          <operation>create</operation>
      <response-code>200</response-code>
      <response-string>success</response-string>
      <version>1</version>
      <ccmp:sidebarByRefResponse>
          <sidebarByRefInfo entity="xcon:8971212@example.com">
              <info:conference-description>
                  <info:display-text>
                      SIDEBAR CONFERENCE registered by Alice
                  </info:display-text>
```

```
                    <info:available-media>
                        <info:entry label="123">
                            <info:display-text>
                                 main conference audio
                            </info:display-text>
                            <info:type>audio</info:type>
                            <info:status>sendrecv</info:status>
                        </info:entry>
                        <info:entry label="456">
                            <info:display-text>
                                 main conference video
                            </info:display-text>
                            <info:type>video</info:type>
                            <info:status>sendrecv</info:status>
                        </info:entry>
                    </info:available-media>
                </info:conference-description>
                <info:conference-state>
                    <info:active>false</info:active>
                </info:conference-state>
                <info:users>
                    <xcon:allowed-users-list>
                        <xcon:target method="dial-in"
                              uri="xcon-userid:Alice@example.com"/>
                        <xcon:target method="dial-in"
                              uri="xcon-userid:Bob@example.com"/>
                        <xcon:target method="dial-in"
                              uri="xcon-userid:Carol@example.com"/>
                        <xcon:target method="dial-in"
                              uri="xcon-userid:David@example.com"/>
                        <xcon:target method="dial-in"
                              uri="xcon-userid:Ethel@example.com"/>
                    </xcon:allowed-users-list>
                    <xcon:sidebar-parent>
                        xcon:8977878@example.com
                    </xcon:sidebar-parent>
                </info:users>
            </sidebarByRefInfo>
        </ccmp:sidebarByRefResponse>
    </ccmpResponse>
  </ccmp:ccmpResponse>
```

3. sidebarByRefRequest/update message (Alice updates the sidebar)

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpRequest
                xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
                xmlns:info="urn:ietf:params:xml:ns:conference-info"
```

```
              xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
    <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:type="ccmp:ccmp-sidebarByRef-request-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8971212@example.com</confObjID>
        <operation>update</operation>
        <ccmp:sidebarByRefRequest>
            <sidebarByRefInfo entity="xcon:8971212@example.com">
                <info:conference-description>
                    <info:display-text>
                        sidebar with Alice, Bob, Ethel and Fred
                    </info:display-text>
                    <info:available-media>
                        <info:entry label="123">
                            <info:display-text>
                                 main conference audio
                            </info:display-text>
                            <info:type>audio</info:type>
                            <info:status>inactive</info:status>
                        </info:entry>
                        <info:entry label="456">
                            <info:display-text>
                                 main conference video
                        </info:display-text>
                            <info:type>video</info:type>
                            <info:status>inactive</info:status>
                        </info:entry>
                        <info:entry label="AUTO_GENERATE_1">
                            <info:display-text>
                                 sidebar audio
                            </info:display-text>
                            <info:type>audio</info:type>
                            <info:status>sendrecv</info:status>
                        </info:entry>
                        <info:entry label="AUTO_GENERATE_2">
                            <info:display-text>
                                 sidebar video
                            </info:display-text>
                            <info:type>video</info:type>
                            <info:status>sendrecv</info:status>
                            <xcon:controls>
                                 <xcon:video-layout>
                                       single-view
                                 </xcon:video-layout>
                            </xcon:controls>
                        </info:entry>
                    </info:available-media>
                </info:conference-description>
```

```
                <info:conference-state>
                    <info:active>false</info:active>
                </info:conference-state>
                <info:users>
                    <xcon:allowed-users-list>
                        <xcon:target method="dial-out"
                              uri="xcon-userid:Alice@example.com"/>
                        <xcon:target method="dial-out"
                              uri="xcon-userid:Bob@example.com"/>
                        <xcon:target method="dial-out"
                              uri="sip:fred@example.com"/>
                    </xcon:allowed-users-list>
                </info:users>
            </sidebarByRefInfo>
        </ccmp:sidebarByRefRequest>
    </ccmpRequest>
  </ccmp:ccmpRequest>
```

4. sidebarByRefResponse/update message (sidebar updated)

```
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <ccmp:ccmpResponse
                xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
                xmlns:info="urn:ietf:params:xml:ns:conference-info"
                xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
    <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:type="ccmp:ccmp-sidebarByRef-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8971212@example.com</confObjID>
        <operation>update</operation>
        <response-code>200</response-code>
        <response-string>success</response-string>
        <version>2</version>
        <ccmp:sidebarByRefResponse/>
    </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 22: External Sidebar Messaging Details

7.3. Private Messages

The case of private messages can be handled as a sidebar with just two participants, similar to the example in Section 7.1. Unlike the previous example, rather than using audio within the sidebar, Alice could just add an additional text-based media stream to the sidebar in order to convey her textual messages to Bob, while still viewing and listening to the main conference.

In this scenario, Alice requests to the conference server the creation of a private chat room within the main conference context (presented in Figure 19) in which the involved participants are just Bob and herself. This can be achieved through the following CCMP transaction (Figure 23).

1. Alice forwards a sidebarByValRequest/create message to the conference server with the main conference XCON-URI in the <confObjID> parameter and the desired sidebar conference object in the <sidebarByValInfo> field. In this way, a sidebar creation using user-provided conference information is requested from the conference server. Please note that, unlike the previous sidebar examples, in this case, a completely new conference object to describe the sidebar is provided: there is no cloning involved, while the <confObjID> still enforces the parent-child relationship between the main conference and the to-be-created sidebar.

2. The conference server, after checking Alice's rights and validating the conference object carried in the request, creates the required sidebar-by-val conference and a new XCON-URI for it. Instead of cloning the main conference object, as shown in Sections 7.1 and 7.2, the sidebar is created on the basis of the user-provided conference information. However, the parent relationship between the main conference and the newly created sidebar is still maintained by the conference server (as a consequence of the chosen CCMP request message type -- the sidebarByVal one) and it is reflected by the <sidebar-parent> element in the <sidebarByValInfo> element returned in the sidebarByValResponse message. Please notice that, according to the CCMP specification, the return of the created sidebar data in this kind of "success" response is not mandatory.

1. sidebarByValRequest/create message (Alice creates a private chat room between Bob and herself)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpRequest
            xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
            xmlns:info="urn:ietf:params:xml:ns:conference-info"
            xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
    <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:type="ccmp:ccmp-sidebarByVal-request-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977878@example.com</confObjID>
        <operation>create</operation>
        <ccmp:sidebarByValRequest>
            <sidebarByValInfo entity="xcon:AUTO_GENERATE_1@example.com">
```

```
                <info:conference-description>
                  <info:display-text>
                        private textual sidebar alice - bob
                  </info:display-text>
                  <info:available-media>
                        <info:entry label="123">
                            <info:display-text>
                                main conference audio
                            </info:display-text>
                            <info:type>audio</info:type>
                            <info:status>recvonly</info:status>
                        </info:entry>
                        <info:entry label="456">
                            <info:display-text>
                                main conference video
                            </info:display-text>
                            <info:type>video</info:type>
                            <info:status>recvonly</info:status>
                        </info:entry>
                        <info:entry label="AUTO_GENERATE_2">
                            <info:display-text>
                                sidebar text
                            </info:display-text>
                            <info:type>text</info:type>
                            <info:status>sendrecv</info:status>
                        </info:entry>
                    </info:available-media>
                </info:conference-description>
                <info:users>
                    <xcon:allowed-users-list>
                        <xcon:target method="dial-out"
                              uri="xcon-userid:Alice@example.com"/>
                        <xcon:target method="dial-out"
                              uri="xcon-userid:Bob@example.com"/>
                    </xcon:allowed-users-list>
                </info:users>
            </sidebarByValInfo>
        </ccmp:sidebarByValRequest>
    </ccmpRequest>
</ccmp:ccmpRequest>
```

2. sidebarByValResponse/create message (sidebar returned)

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpResponse
              xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
              xmlns:info="urn:ietf:params:xml:ns:conference-info"
              xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
```

```
    <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:type="ccmp:ccmp-sidebarByVal-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8974545@example.com</confObjID>
        <operation>create</operation>
        <response-code>200</response-code>
        <response-string>success</response-string>
            <version>1</version>
        <ccmp:sidebarByValResponse>
            <sidebarByValInfo entity="xcon:8974545@example.com">
                <info:conference-description>
                    <info:display-text>
                        private textual sidebar alice - bob
                    </info:display-text>
                    <info:available-media>
                        <info:entry label="123">
                            <info:display-text>
                                main conference audio
                            </info:display-text>
                            <info:type>audio</info:type>
                            <info:status>recvonly</info:status>
                        </info:entry>
                        <info:entry label="456">
                            <info:display-text>
                                main conference video
                            </info:display-text>
                            <info:type>video</info:type>
                            <info:status>recvonly</info:status>
                        </info:entry>
                        <info:entry label="789">
                            <info:display-text>
                                sidebar text
                            </info:display-text>
                            <info:type>text</info:type>
                            <info:status>sendrecv</info:status>
                        </info:entry>
                    </info:available-media>
                    <xcon:sidebar-parent>
                         xcon:8977878@example.com
                    </xcon:sidebar-parent>
                </info:conference-description>
                <info:users>
                    <xcon:allowed-users-list>
                        <xcon:target method="dial-out"
                              uri="xcon-userid:Alice@example.com"/>
                        <xcon:target method="dial-out"
                              uri="xcon-userid:Bob@example.com"/>
                    </xcon:allowed-users-list>
```

```
                </info:users>
            </sidebarByValInfo>
        </ccmp:sidebarByValResponse>
    </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 23: Sidebar for Private Messages Scenario

7.4. Observing and Coaching

"Observing and Coaching" is one of the most interesting sidebar-related scenarios. In fact, it highlights two different interactions that have to be properly coordinated.

An example of observing and coaching is shown in Figure 25. In this example, call center agent Bob is involved in a conference with customer Carol. Since Bob is a new agent and Alice sees that he has been on the call with Carol for longer than normal, she decides to observe the call and coach Bob as necessary. Of course, the conferencing system must make sure that the customer Carol is not aware of the presence of the coach Alice. This makes the use of a sidebar necessary for the success of the scenario.

Consider the following as the conference document associated with the video conference involving Bob (the call agent) and Carol (the customer) (Figure 24):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <info:conference-info
                xmlns:info="urn:ietf:params:xml:ns:conference-info"
                xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
                xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
                entity="xcon:8978383@example.com">
     <info:conference-description>
        <info:display-text>
                CUSTOMER SERVICE conference
        </info:display-text>
        <info:conf-uris>
            <info:entry>
               <info:uri>sip:8978383@example.com</info:uri>
               <info:display-text>conference sip uri</info:display-text>
            </info:entry>
        </info:conf-uris>
        <info:available-media>
          <info:entry label="123">
            <info:display-text>service audio</info:display-text>
            <info:type>audio</info:type>
            <info:status>sendrecv</info:status>
```

```
          </info:entry>
          <info:entry label="456">
            <info:display-text>service video</info:display-text>
            <info:type>video</info:type>
            <info:status>sendrecv</info:status>
            <xcon:controls>
                    <xcon:video-layout>single-view</xcon:video-layout>
             </xcon:controls>
          </info:entry>
        </info:available-media>
    </info:conference-description>
    <info:conference-state>
        <info:active>true</info:active>
    </info:conference-state>
    <info:users>
        <info:user entity="xcon-userid:bob@example.com">
            <info:display-text>Bob - call agent</info:display-text>
            <info:endpoint entity="sip:bob@example.com">
                <info:media id="1">
                    <info:label>123</info:label>
                    <info:status>sendrecv</info:status>
                </info:media>
                <info:media id="2">
                    <info:label>456</info:label>
                    <info:status>sendrecv</info:status>
                </info:media>
            </info:endpoint>
        </info:user>
        <info:user entity="xcon-userid:carol@example.com">
            <info:display-text>Carol - customer</info:display-text>
            <info:endpoint entity="sip:carol@example.com">
                <info:media id="1">
                    <info:label>123</info:label>
                    <info:status>sendrecv</info:status>
                </info:media>
                <info:media id="2">
                    <info:label>456</info:label>
                    <info:status>sendrecv</info:status>
                </info:media>
            </info:endpoint>
        </info:user>
    </info:users>
  </info:conference-info>
```

Figure 24: A Call-Center Conference Object Example

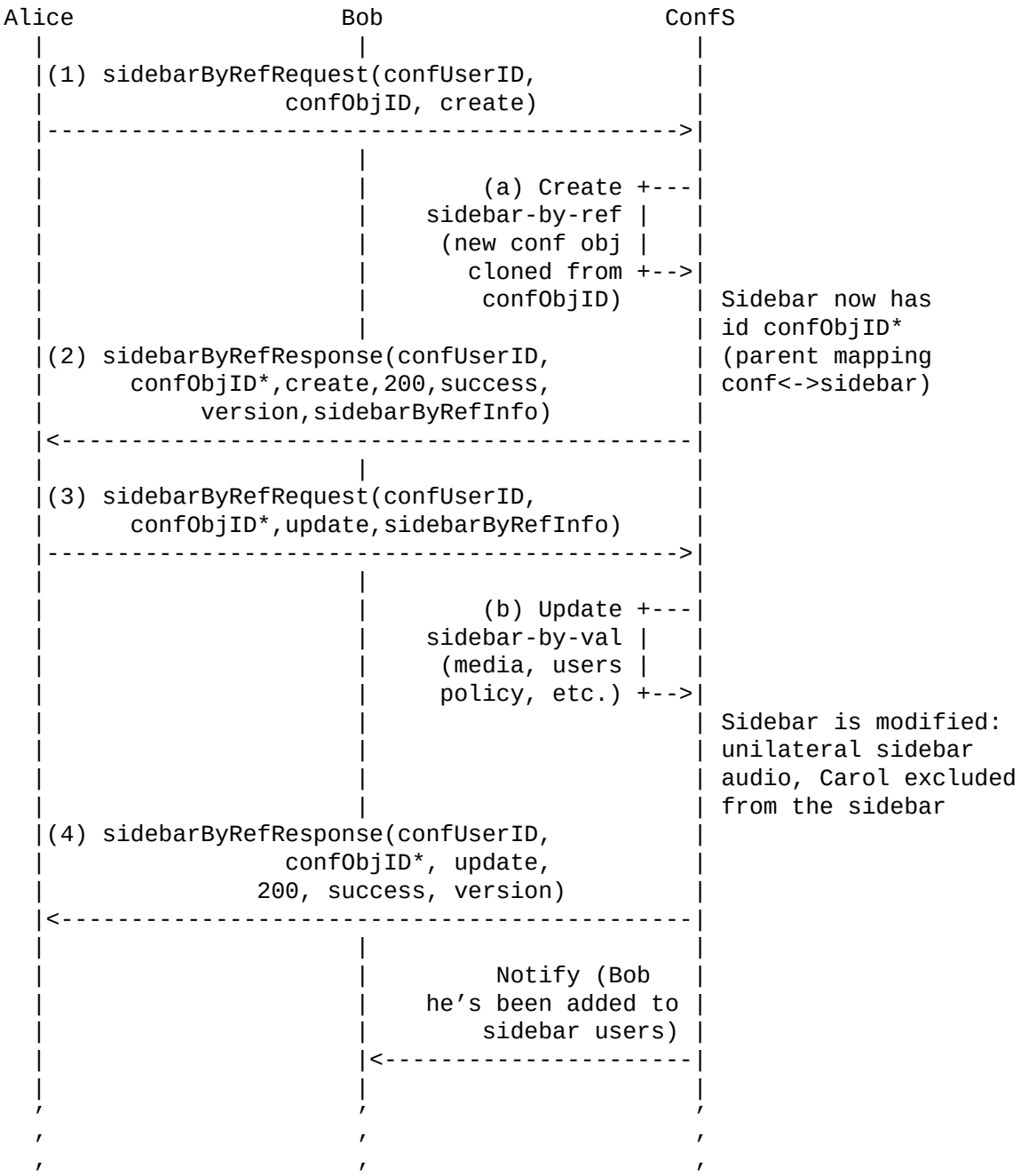


Figure 25: Supervisor Creating a Sidebar for Observing/Coaching

1. Upon receipt of the sidbarByRefRequest/create message from Alice to create a new sidebar conference from the <confObjID> received in the request, the conference server uses the received active conference to clone a conference reservation for the sidebar. The conference server also allocates a new XCON-URI to be used for any subsequent protocol requests directed to the new sidebar. The conference server maintains the mapping between this sidebar conference ID and the one associated with the main conference instance. The conference server sends a sidebarByRefResponse message with the new XCON-URI in the <confObjID> field and other relevant information in the <sidebarByRefInfo>.

2. Upon receipt of the sidebarByRefResponse message, Alice manipulates the data received in the <sidebarByRefInfo> in the response. Alice wants only Bob to be involved in the sidebar; thus, she updates the <allowed-users-list> to include only Bob and herself. Alice also wants the audio to be received by herself and Bob from the original conference, but wants any outgoing audio from herself to be restricted to the participants in the sidebar, whereas Bob’s outgoing audio should go to the main conference, so that both Alice and the customer Carol hear the same audio from Bob. Alice sends a sidebarByRefRequest message with an "update" <operation> including the updated sidebar information in the <sidebarByRefInfo> element.

3. Upon receipt of the sidebarByRefRequest/update message, the conference server ensures that Alice has the appropriate authority based on the policies associated with that specific conference object to perform the operation. In order to request the insertion of a further media stream in the sidebar (i.e., in this example an audio stream from Alice to Bob), the requester has to provide a new <entry> element in the <available-media> field of the <sidebarByRefInfo>. The mandatory ’label’ attribute of that new <entry> is filled with a dummy value "AUTO_GENERATE_1", but it will contain the real server-generated media stream identifier when the media stream is effectively allocated on the server side. Similarly, the mandatory ’id’ attribute in the <media> element referring to the new sidebar audio stream under both Alice’s and Bob’s <endpoint> contains a wildcard value, respectively, "AUTO_GENERATE_2" and "AUTO_GENERATE_3": those values will be replaced with the appropriated server-generated identifiers upon the creation of the referred media stream. We are assuming the conference server is able to recognize those dummy values as placeholders.

4. After validating the data, the conference server sends a sidebarByRefResponse message. Based upon the contact information provided for Bob by Alice, the call signaling to add Bob to the

sidebar with the appropriate media characteristics is instigated through the focus. Bob is notified of his addition to the sidebar via the conference notification service; thus, he is aware that Alice, the supervisor, is available for coaching him through this call.

1. sidebarByRefRequest/create message (Alice as coach creates a sidebar)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpRequest xmlns:info="urn:ietf:params:xml:ns:conference-info"
                xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
                xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
    <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:type="ccmp:ccmp-sidebarByRef-request-message-type">
        <confUserID>xcon-userid:alice@example.com</confUserID>
        <confObjID>xcon:8978383@example.com</confObjID>
        <operation>create</operation>
        <ccmp:sidebarByRefRequest/>
    </ccmpRequest>
</ccmp:ccmpRequest>
```

2. sidebarByRefResponse/create message (sidebar created)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpResponse
                xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
                xmlns:info="urn:ietf:params:xml:ns:conference-info"
                xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
    <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:type="ccmp:ccmp-sidebarByRef-response-message-type">
        <confUserID>xcon-userid:alice@example.com</confUserID>
        <confObjID>xcon:8971313@example.com</confObjID>
        <operation>create</operation>
        <response-code>200</response-code>
        <response-string>Success</response-string>
        <version>1</version>
        <ccmp:sidebarByRefResponse>
            <sidebarByRefInfo entity="xcon:8971313@example.com">
                <info:conference-description>
                    <info:display-text>
                        SIDEBAR CONFERENCE registered by alice
                    </info:display-text>
                    <info:available-media>
                        <info:entry label="123">
                            <info:display-text>
                                 main conference audio
                            </info:display-text>
                            <info:type>audio</info:type>
```

```
                        <info:status>sendrecv</info:status>
                    </info:entry>
                    <info:entry label="456">
                        <info:display-text>
                             main conference video
                        </info:display-text>
                        <info:type>video</info:type>
                        <info:status>sendrecv</info:status>
                    </info:entry>
                </info:available-media>
                <xcon:sidebar-parent>
                    xcon:8971313@example.com
                </xcon:sidebar-parent>
            </info:conference-description>
            <info:conference-state>
                <info:active>false</info:active>
            </info:conference-state>
            <info:users>
                <xcon:allowed-users-list>
                    <xcon:target method="dial-in"
                          uri="xcon-userid:alice@example.com"/>
                    <xcon:target method="dial-in"
                          uri="xcon-userid:bob@example.com"/>
                    <xcon:target method="dial-in"
                          uri="xcon-userid:carol@example.com"/>
                </xcon:allowed-users-list>
            </info:users>
        </sidebarByRefInfo>
    </ccmp:sidebarByRefResponse>
  </ccmpResponse>
</ccmp:ccmpResponse>
```

3. sidebarByRefRequest/update message (Alice introduces unilateral sidebar audio and excludes Carol from the sidebar)

```
<ccmp:ccmpRequest
            xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
            xmlns:info="urn:ietf:params:xml:ns:conference-info"
            xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
  <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:type="ccmp:ccmp-sidebarByRef-request-message-type">
    <confUserID>xcon-userid:alice@example.com</confUserID>
    <confObjID>xcon:8971313@example.com</confObjID>
    <operation>update</operation>
    <ccmp:sidebarByRefRequest>
        <sidebarByRefInfo entity="xcon:8971313@example.com">
```

```
                <info:conference-description>
                    <info:display-text>
                        Coaching sidebar Alice and Bob
                    </info:display-text>
                    <info:available-media>
                        <info:entry label="AUTO_GENERATE_1">
                            <info:display-text>
                                 Alice-to-Bob audio
                            </info:display-text>
                            <info:type>audio</info:type>
                            <info:status>sendrecv</info:status>
                        </info:entry>
                    </info:available-media>
                </info:conference-description>
                <info:conference-state>
                    <info:active>false</info:active>
                </info:conference-state>
                <info:users>
                    <info:user entity="xcon-userid:alice@example.com">
                      <info:endpoint entity="sip:alice@example.com">
                        <info:media id="AUTO_GENERATE_2">
                         <info:label>AUTO_GENERATE_1</info:label>
                         <info:status>sendonly</info:status>
                        </info:media>
                      </info:endpoint>
                    </info:user>
                    <info:user entity="xcon-userid:bob@example.com">
                      <info:endpoint entity="sip:bob@example.com">
                        <info:media id="AUTO_GENERATE_3">
                         <info:label>AUTO_GENERATE_1</info:label>
                         <info:status>recvonly</info:status>
                        </info:media>
                      </info:endpoint>
                    </info:user>
                    <xcon:allowed-users-list>
                        <xcon:target method="dial-in"
                              uri="xcon-userid:alice@example.com"/>
                        <xcon:target method="dial-out"
                              uri="xcon-userid:bob@example.com"/>
                    </xcon:allowed-users-list>
                </info:users>
            </sidebarByRefInfo>
        </ccmp:sidebarByRefRequest>
    </ccmpRequest>
  </ccmp:ccmpRequest>
```

4. sidebarByRefRequest/update message (updates accepted)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpResponse
                xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
                xmlns:info="urn:ietf:params:xml:ns:conference-info"
                xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
    <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:type="ccmp:ccmp-sidebarByRef-response-message-type">
        <confUserID>xcon-userid:alice@example.com</confUserID>
        <confObjID>xcon:8971313@example.com</confObjID>
        <operation>update</operation>
        <response-code>200</response-code>
        <response-string>success</response-string>
        <version>2</version>
        <ccmp:sidebarByRefResponse/>
    </ccmpResponse>
 </ccmp:ccmpResponse>
```

Figure 26: Coaching and Observing Messaging Details

8. Removing Participants and Deleting Conferences

The following scenarios detail the basic operations associated with removing participants from conferences and entirely deleting conferences. The examples assume that a conference has already been correctly established, with media, if applicable, per one of the examples in Section 5.

8.1. Removing a Party

Figure 27 provides an example of a client, Alice, removing another participant, Bob, from a conference. This example assumes an established conference with Alice, Bob, Claire, and Duck. In this example, Alice wants to remove Bob from the conference so that the group can continue in the same conference without Bob’s participation.

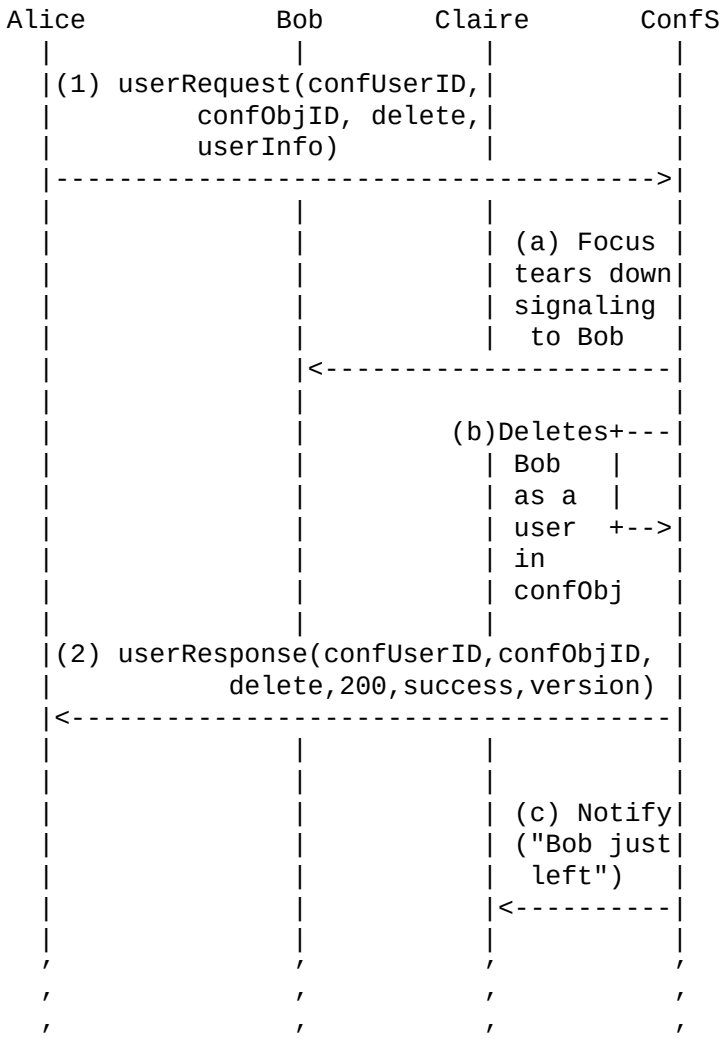


Figure 27: Client Manipulation of Conference - Remove a Party

1. Alice sends a userRequest message with a "delete" <operation>. The conference server ensures that Alice has the appropriate authority based on the policies associated with that specific conference object to perform the operation.

2. Based upon the contact and media information in the conference object for Bob in the <userInfo> element, the conferencing system starts the process to remove Bob (e.g., the call signaling to remove Bob from the conference is instigated through the focus). The conference server updates the data in the conference object, thus, removing Bob from the <users> list. After updating the data, the conference server sends a userResponse message to

Alice. Depending upon the policies, other participants (e.g., Claire) may be notified of the removal of Bob from the conference via the conference notification service.

1. userRequest/delete message (Alice deletes Bob)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest
       xmlns:info="urn:ietf:params:xml:ns:conference-info"
       xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
       xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
    <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:type="ccmp:ccmp-user-request-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
        <operation>delete</operation>
        <ccmp:userRequest>
            <userInfo entity="xcon-userid:Bob@example.com"/>
        </ccmp:userRequest>
    </ccmpRequest>
</ccmp:ccmpRequest>
```

2. userResponse/delete message (Bob has been deleted)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
       xmlns:info="urn:ietf:params:xml:ns:conference-info"
       xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
       xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
    <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:type="ccmp:ccmp-user-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
            <operation>delete</operation>
        <response-code>200</response-code>
        <response-string>success</response-string>
        <version>17</version>
        <ccmp:userResponse/>
    </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 28: Removing a Participant Messaging Details

8.2. Deleting a Conference

In this section, an example of a successful conference deletion is provided (Figure 29).

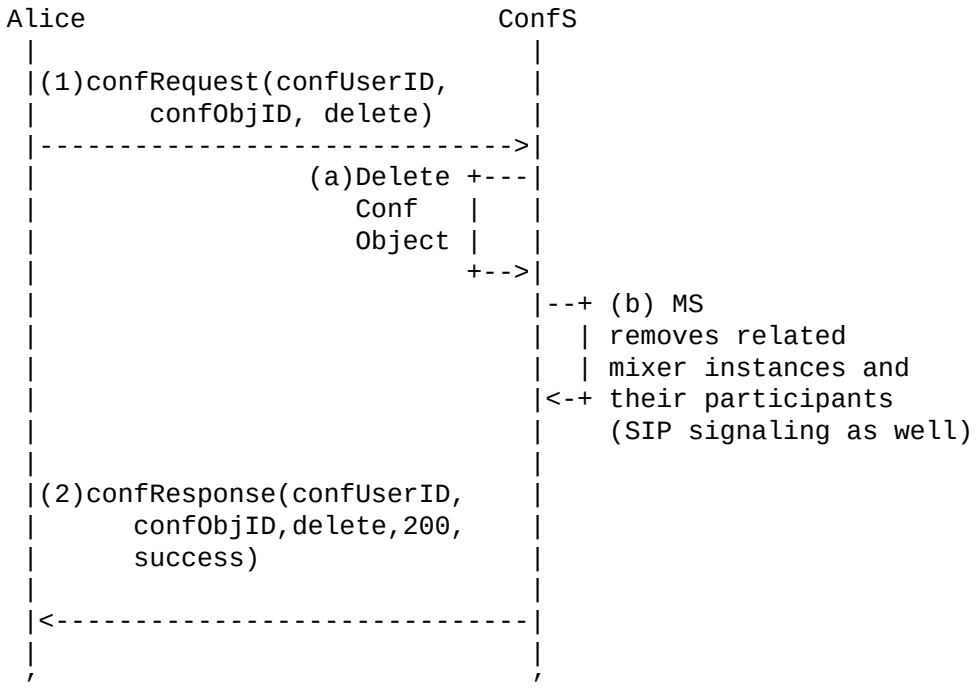


Figure 29: Deleting a Conference

1. The conferencing system client Alice sends a confRequest message with a "delete" operation to be performed on the conference identified by the XCON-URI carried in the <confObjID> parameter. The conference server, on the basis of the <confUserID> included in the receipt request, ensures that Alice has the appropriate authority to fulfill the operation.

2. After validating Alice's rights, the conference server instigates the process to delete the conference object, disconnecting participants and removing associated resources such as mixer instances. Then, the conference server returns a confResponse message to Alice with "200" as <response-code> and the deleted conference XCON-URI in the <confObjID> field.

1. confRequest/delete message (Alice deletes a conference)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest
       xmlns:info="urn:ietf:params:xml:ns:conference-info"
       xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
       xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
    <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:type="ccmp:ccmp-conf-request-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
        <operation>delete</operation>
        <ccmp:confRequest/>
    </ccmpRequest>
</ccmp:ccmpRequest>
```

2. confResponse/delete message ("200", "success")

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
       xmlns:info="urn:ietf:params:xml:ns:conference-info"
       xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
       xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
    <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:type="ccmp:ccmp-conf-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
        <operation>delete</operation>
        <response-code>200</response-code>
        <response-string>success</response-string>
        <ccmp:confResponse/>
    </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 30: Deleting a Conference Messaging Details

9. Security Considerations

The security considerations applicable to the implementation of these call flows are documented in the XCON framework, with additional security considerations documented in the CCMP document. Statements with regard to the necessary security are discussed in particular flows; however, this is for informational purposes only. The implementer is encouraged to carefully consider the security requirements in the normative documents.

10. Acknowledgements

The detailed content for this document is derived from the prototype work of Lorenzo Miniero, Simon Pietro Romano, Tobia Castaldi, and their colleagues at the University of Napoli.

11. References

11.1. Normative References

[RFC5239] Barnes, M., Boulton, C., and O. Levin, "A Framework for Centralized Conferencing", RFC 5239, June 2008.

[RFC6501] Novo, O., Camarillo, G., Morgan, D., and J. Urpalainen, "Conference Information Data Model for Centralized Conferencing (XCON)", RFC 6501, March 2012.

[RFC6502] Camarillo, G., Srinivasan, S., Even, R., and J. Urpalainen, "Conference Event Package Data Format Extension for Centralized Conferencing (XCON)", RFC 6502, March 2012.

[RFC6503] Barnes, M., Boulton, C., Romano, S., and H. Schulzrinne, "Centralized Conferencing Manipulation Protocol", RFC 6503, March 2012.

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

11.2. Informative References

[CALL-FLOWS] Amirante, A., Castaldi, T., Miniero, L., and S. Romano, "Media Control Channel Framework (CFW) Call Flow Examples", Work in Progress, July 2011.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.

[RFC4579] Johnston, A. and O. Levin, "Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents", BCP 119, RFC 4579, August 2006.

[RFC4582] Camarillo, G., Ott, J., and K. Drage, "The Binary Floor Control Protocol (BFCP)", RFC 4582, November 2006.

[RFC4597] Even, R. and N. Ismail, "Conferencing Scenarios", RFC 4597, August 2006.

[RFC5567] Melanchuk, T., "An Architectural Framework for Media Server Control", RFC 5567, June 2009.

[RFC6505] McGlashan, S., Melanchuk, T., and C. Boulton, "A Mixer Control Package for the Media Control Channel Framework", RFC 6505, March 2012.

Authors' Addresses

Mary Barnes
Polycom
TX
USA

EMail: mary.ietf.barnes@gmail.com

Lorenzo Miniero
Meetecho
Via Carlo Poerio 89/a
Napoli 80121
Italy

EMail: lorenzo@meetecho.com

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