XCON Working Group
Internet-Draft
Intended status: Informational
Expires: August 21, 2010

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

February 17, 2010

Centralized Conferencing Manipulation Protocol (CCMP) Call Flow Examples

draft-ietf-xcon-examples-03

Abstract

This document provides detailed call flows for the scenarios documented in the Centralized Conferencing (XCON) Framework and the XCON Scenarios. 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). The objective is to provide a base reference for both protocol researchers and developers.

Table of Contents

1. Introduction 4
2. Conventions 4
3. Terminology 4
4. Overview 5
5. Working with CCMP 6
5.1. CCMP and the Data Model 6
5.2. Using HTTP as a transport 8
5.3. Conference Notifications 11
6. Conference Creation 15
6.1. Basic Conference Creation 15
6.2. Conference Creation using Blueprints 20
6.3. Conference Creation using User-Provided Conference Information 27
6.4. Cloning an Existing Conference 32
7. Conference Users Scenarios and Examples 35
7.1. Adding a Party 36
7.2. Muting a Party 39
7.3. Conference Announcements and Recordings 43
7.4. Monitoring for DTMF 47
7.5. Entering a password-protected conference 47
8. Sidebars Scenarios and Examples 49
8.1. Internal Sidebar 50
8.2. External Sidebar 59
8.3. Private Messages 65
8.4. Observing and Coaching 69
9. Removing Participants and Deleting Conferences 76
9.1. Removing a Party 76
9.2. Deleting a Conference 79
10. IANA Considerations 80
11. Security Considerations 80
12. Change Summary 81
13. Acknowledgements 83
14. References 83
14.1. Normative References 83
14.2. Informative References 83
Authors' Addresses 84

1. Introduction

This document provides detailed call flows for the scenarios documented in the Framework for Centralized Conferencing (XCON Framework) [RFC5239] and 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)[I-D.ietf-xcon-ccmp].

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. Conventions

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations. In this document, these key words are used when describing normative functionality based on the XCON Framework and CCMP.

Note that due to RFC formatting conventions, this document often splits message details whose content would exceed 72 characters. A backslash character marks where this line folding has taken place. This backslash and its trailing CRLF and whitespace would not appear in the actual protocol contents.

3. Terminology

This document uses the same terminology as found in the referenced documents, 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 UAC as the client notation in the media control call flows [I-D.ietf-mediactrl-call-flows]. A CMCC differs from a generic Media Client in being an XCON-aware entity, thus being able to also issue CCMP requests.

Conferencing Server (ConfS): In this document, the "Conferencing Server" term is used interchangeably with the term Application Server (AS) as used in the Media Control Architectural Framework [RFC5567]. A Conferencing 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].

4. 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 [I-D.ietf-xcon-ccmp]. This is intended to be a simple guide for the use of the Conference Control Protocol between the Conferencing 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 Conferencing 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 [I-D.ietf-mediactrl-mixer-control-package], call signaling or binary floor control [RFC4582] protocols. This document references the scenarios in the Media Control call flows [I-D.ietf-mediactrl-call-flows], SIP Call Control Conferencing [RFC4579] and binary floor control protocol documents.

The rest of this document is organized as follows. Section 5 presents an overview on CCMP, together with some implementation-related details and related matters like HTTP transport and

notifications. Section 6 presents the reader with examples showing the different approaches CCMP provides to create a new conference. Section 7 more generally addresses the different user-related manipulations that can be achieved by means of CCMP, by presenting a number of interesting scenarios. Section 8 addresses the several scenarios that may involve the use of sidebars. Section 9 shows how CCMP can be used to remove conferences and users from the system. Finally, Section 11 provides a few details for what concerns the Security Considerations when it comes to implementing CCMP.

5. Working with CCMP

This section aims at being a brief introduction to how the Centralized Conferencing Manipulation Protocol (CCMP) [I-D.ietf-xcon-ccmp] works and how it can be transported across a network. Some words will be spent to describe a typical CCMP interaction, by focusing on relevant aspects of the client-server communication. Please notice that this section is by no means to be considered as a replacement for the CCMP document, which remains a mandatory read before approaching the following sections. It is just conceived to help the reader take the first steps toward the actual protocol interactions.

First of all, some lines will be devoted to the protocol by itself in Section 5.1, together with some recommendations from an implementation point of view. In Section 5.2, instead, an effective CCMP interaction will be presented by exploiting HTTP as a transport. Finally, a few words will be spent on notifications in Section 5.3.

Once done with these preliminary steps, some actual flows will be presented and described in detail in the sections to follow.

5.1. CCMP and the Data Model

CCMP is an XML-based protocol. It has been designed as a request/response protocol. Besides, it is completely stateless, which means implementations can safely handle transactions indipendently from each other.

The protocol allows for the manipulation of conference objects and related users. By manipulation it is implied, as the document specifies, that a Conferencing Client (briefly CMCC in all the following sections) can 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 to several different approaches;
- add/remove users to/from a conference;
- update a conference with respect to all of its aspects;

and so on.

It is worthwile to note that, while CCMP acts as the means to manipulate conference objects, its specification does not define these objects as well. In fact, a separate document has been written to specify how a conference object and all its components have to be constructed: the Conference Information Data Model for Centralized Conferencing (XCON) [I-D.ietf-xcon-common-data-model]. CCMP, according to 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 implementors take special care in carefully checking the Data Model handlers as well in order to avoid potential mistakes.

Of course, there are cases where a mandatory element in the Data Model cannot be assigned in a conference object by a CCMP user. For instance, a CMCC may be requesting the direct creation of a new conference: in this case, a conference object assumes an 'entity' element uniquely identifying the conference to be in place. Anyway, the CMCC has no way to a priori know what the entity will be like, considering it will only be generated by the ConfS after the request. For scenarios like this one, the CCMP specification envisages the use of a dedicated placeholder wildcard to make the conference object compliant with the Data Model: the wildcard would then be replaced by the ConfS with the right value.

5.2. Using HTTP as a transport

[I-D.ietf-xcon-ccmp] presents several ways by which CCMP requests and responses can be transported from a client to a server and viceversa. Nevertheless, more focus is given on HTTP as a solution for this transport matter, and a whole chapter is devoted in the document to how HTTP can be used for it. This document is agnostic with respect to the transport in use: this means that all the call flows herein presented will just show the CCMP interactions, without talking about how the messages could have gone across the network. Nevertheless, the following lines will provide a brief explanation, from a more practical point of view, of how HTTP might be exploited to carry CCMP messages.

In case HTTP is used as a transport, the specification is very clear with respect to how the interaction has to occur. Specifically, a CMCC is assumed to send his request as part of an HTTP POST message, and the Server would reply by means of an HTTP 200 message. In both cases, the HTTP messages would have the CCMP messages as payload, which would be reflected in the Content-Type message. Figure 1 presents a ladder diagram of such interaction, which is followed by a dump of the exchanged HTTP messages for further analysis:

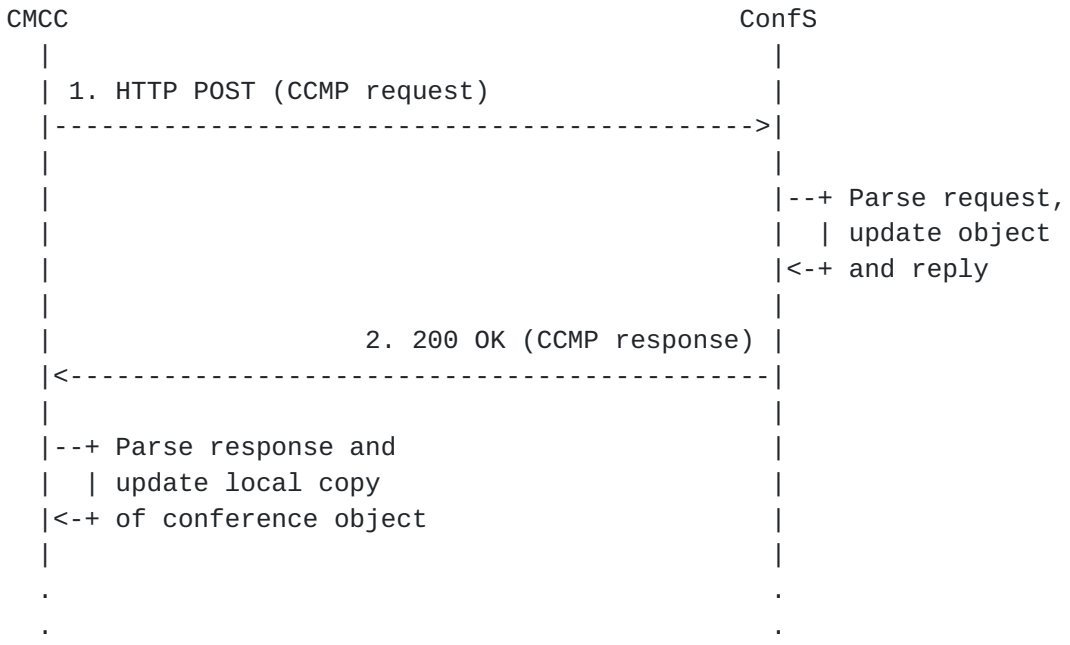


Figure 1: CCMP on HTTP

As it can be seen in the protocol dump in the following lines, the

CMCC has issued a CCMP request (a blueprintRequest with a 'retrieve' operation) towards the Conferencing Server (ConfS). The request has been carried as payload of an HTTP 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 with a 'success' response code) has been carried as payload of an HTTP 200 OK (message 2.). As before, the 'Content-Type' header has been correctly set (application/ ccmp+xml).

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

```
POST /Xcon/Ccmp HTTP/1.1
Content-Length: 657
Content-Type: application/ccmp+xml
Host: example.com:8080
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>success</response-code>
    <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>
```

Just for the sake of completeness, a few words will be spent about the occurred CCMP interaction as well. In fact, despite the simplicity of the request, this flow already provides some relevant information on how CCMP messages are built. Specifically, both the request and the 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 (in this case Alice, since the 'confUserID' has been set to 'xcon-userid:Alice@example.com') has also provided an additional element, 'blueprintRequest'. The name of that element varies according to the request type the CMCC is interested into. 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. As it will be clearer in the following sections, different request types may require different elements, and as a consequence different content.

Considering the request was a 'blueprintRequest', the ConfS has replied with a 'blueprintResponse' element. This element includes a complete dump of the conference object (compliant with the Data Model) describing the requested blueprint, together with an element addressing the result of the client's request (response-code=success).

This section won't delve in additional details for what concerns this interaction. It is just worth noticing that this was the example of the simplest CCMP communication that could take place between a CMCC and a ConfS, a blueprintRequest: this scenario will be described in more detail in Section 6.2.

5.3. Conference Notifications

[RFC5239] presents several different possible protocol interactions between a conferencing server and a conferencing client. One of those interactions is generically called "Notification Protocol", implementing a notification service for all clients interested in being informed by the server whenever something relevant happens in a conference. While at first glance it may appear that such a functionality should belong to a conference control protocol, such feature has been specifically marked as out of scope in CCMP. As a consequence, CCMP has been conceived as a request/answer protocol, and in fact no ways to provide notifications to clients have been introduced in the specification.

Nevertheless, the CCMP document by itself has a brief section presenting some typical ways notifications might be managed. This example document does not foster one rather than another, and all the

flows will always generically present a notification being involved, when it seems appropriate, but not providing any info on how the notification itself has been sent to the interested clients. Anyway, this section will briefly introduce some of the most typical ways a notification service could be implemented and integrated with the functionality provided by CCMP. It is by no means to be intended as a complete list of solutions: the aim of this section is to provide an overview of some of the possible solutions, together with indications on how they may be integrated into a CCMP-based platform.

The first approach that comes to mind is of course the XCON Event Package [I-D.ietf-xcon-event-package]. This specification extends the SIP Event Package for Conference State [RFC4575] and allows for the notification of conference notifications by means of the NOTIFY/ SUBSCRIBE mechanisms of SIP. Specifically, any SIP client who subscribed for notifications related to a specific conference would receive notifications via SIP describing all the changes to the document. An example ladder diagram is presented in Figure 2: in this figure, we assume a CMCC has updated a conference object, and the update is notified to a previously subscribed SIP client.

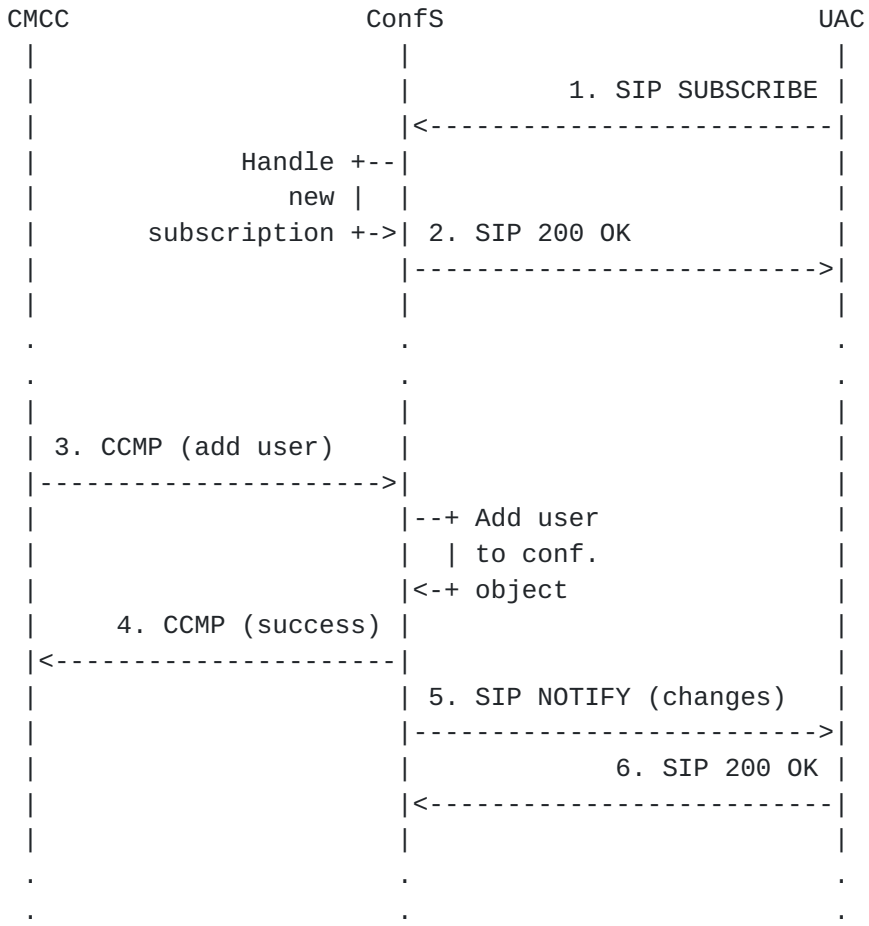


Figure 2: XCON Event Package: SIP notifications

While simple and effective, this solution has a drawback: it assumes that all clients to be notified have a SIP stack. In fact, the approach relies on the SIP SUBSCRIBE/NOTIFY mechanism. This means that a client without a SIP stack would be unable to receive notifications, in case no other means were available. Of course this is not a desired situation in a framework as XCON which has been conceived as being protocol-agnostic.

Considering CCMP is going to be probably most often deployed on HTTP, another way to achieve notifications might be by exploiting some sort of HTTP callbacks, as suggested in the CCMP specification itself. This would allow to overcome the previous limitation, since both the CMCC and the ConfS would already have an HTTP stack to make use of. Using this approach, an interested web client might provide the Conferencing System with an URL to contact whenever updates are available: the update could be part of the notification message itself, or it could be only implicitly referenced by the notification. At the same time, alternative notification means could

be exploited, e.g. by taking advantage of functionality provided by other protocols as XMPP. Figure 3 shows an example of different subscriptions which accordingly trigger notifications after the same relevant event happens.

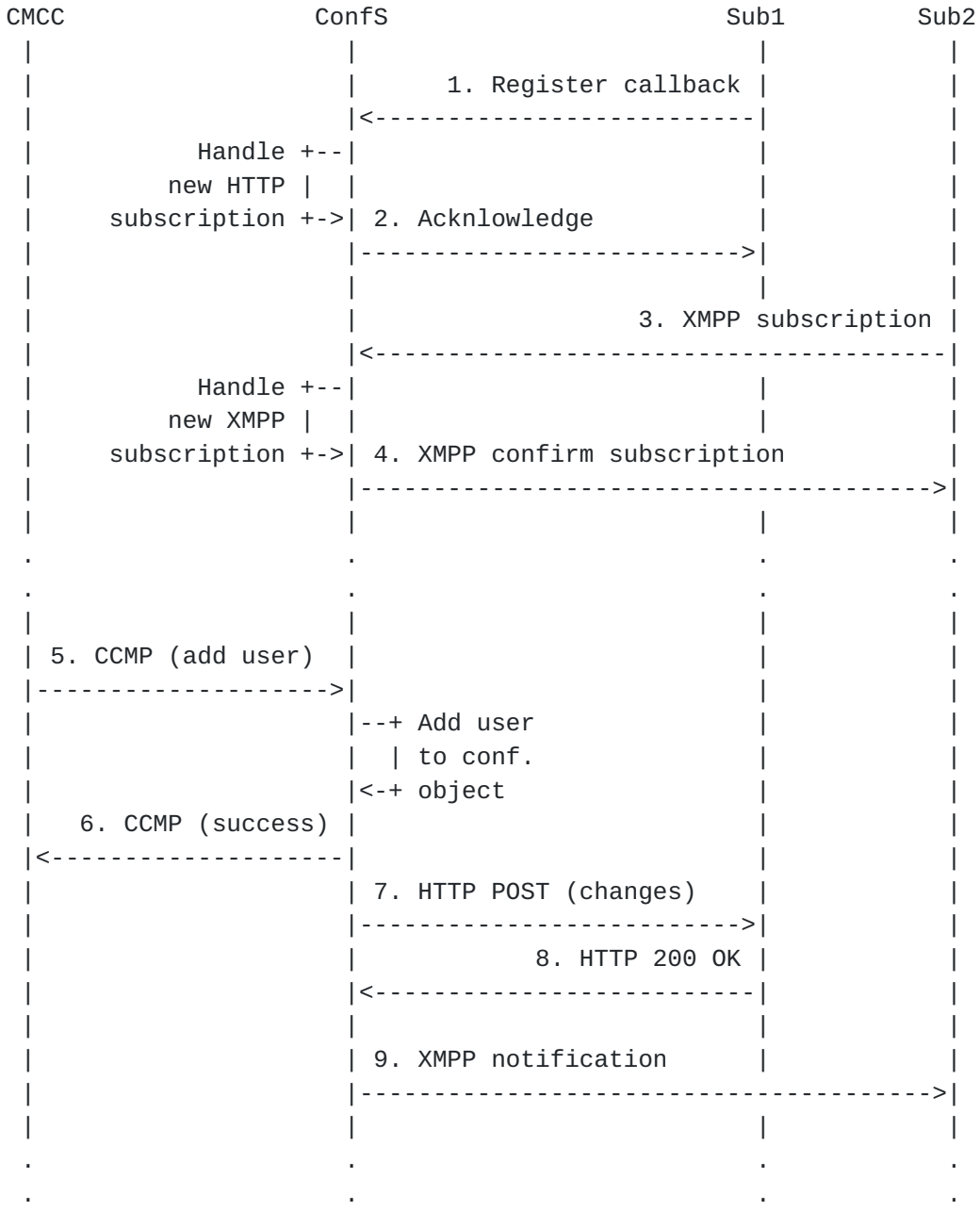


Figure 3: Alternative means for notifications

That said, there are actually many other ways to achieve

notifications in a conferencing system. A conferencing system may rely on several other solutions than the ones presented as periodic checks of a well known URL by interested clients, long polls, BOSH-based communications, Atom/RSS feeds and the like.

6. Conference Creation

This section starts the sequence of call flows of typical XCON-related scenarios provided in this document. Specifically, it 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. HTTP). However, for clarification purposes, the first example Section 6.1 provides the details of the media control messaging along with an example of the standard annotation used throughout the remainder of this document. In subsequent flows, only this annotation (identified by lower case 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 conferencing server vertical bar and those for the media control messaging are on the right side.

6.1. Basic Conference Creation

The simplest manner in which a conference can be created is accomplished by the client sending a "confRequest" message with the "create" operation 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" parameter, the XCON-USERID in the form of the "confUserID" parameter (the same already present in the request) and the data for the conference object in the "confInfo" parameter all returned in the "confResponse" message. According to the implementation of the framework, this example may also add the user that invoked the conference upon creation to the conference object (e.g., "method" attribute in the "target" element of "allowed-users-list" may be set to "dial out" for this client based on the particular conferencing systems default). This is exactly the case depicted in the figure, which is presented to enrich the scenario.

The specific data for the conference object are 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" parameter is not mandatory: if the "confInfo" parameter of the successful confResponse/create is void, a following confRequest/ retrieve 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] or other supported signaling protocols (being XCON agnostic with respect to them), using the signaling interface to the conference focus as described in [RFC4579]. 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 lower case letter items (i.e., (a)...(x)) along with the appropriate text for the processing done by the conferencing server.

As anticipated at the beginning of this section, this example also shows how the conferencing system may make use of other standard components to make available its functionality. An example of that is the MEDIACTRL specification, which allows the conferencing system to configure conference mixes, IVR dialogs and all sort of media-related interactions an application like this may need. So, just to provide the reader with some insight on these interactions, the conferencing system also configures and starts a mixer via MEDIACTRL 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 using MEDIACTRL in B1/B2).

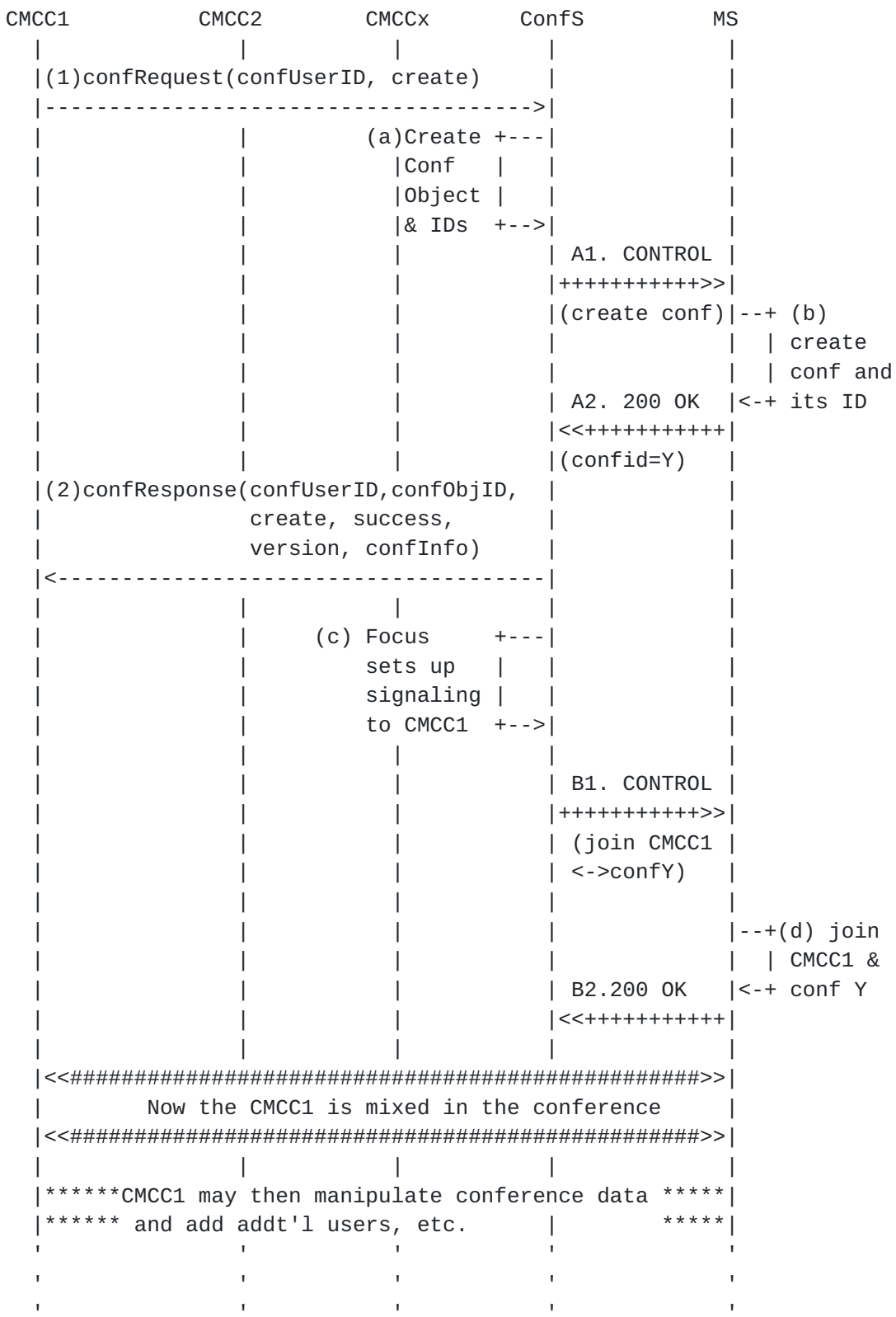

CMCC1
CMCC2
CMCCx
ConfS
MS
(1)confRequest(confUserID, create)
(a)Create Conf Object & IDs
A1. CONTROL
(create conf)
(b) create conf and its ID
A2. 200 OK
(confid=Y)
(2)confResponse(confUserID,confObjID, create, success, version, confInfo)
(c) Focus sets up signaling to CMCC1
B1. CONTROL
(join CMCC1 <->confY)
(d) join CMCC1 & conf Y
B2.200 OK
Now the CMCC1 is mixed in the conference
******CMCC1 may then manipulate conference data ***** and add addt'l users, etc. *****

Figure 4: Create Basic Conference - Complete flow

Figure 5: Create Basic Conference - Annotated 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>
   </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>success</response-code>
     <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-state>
                       <active>false</active>
                     </info:conference-state>
                </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:allowed-users-list>
                </info:users>
            </confInfo>
        </ccmp:confResponse>
     </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 6: Create Basic Conference (Annotated) Detailed Messaging

6.2. 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 like. Anyway, the XCON framework (and CCMP as a consequence) allows for the exploitation of templates. These templates are called "conference blueprints", and are basically conference objects with pre-defined settings except for the actual identifiers. This means that a client might get one of these blueprints, choose the one that more fits his needs, and use the chosen blueprint to create a new conference.

This section addresses exactly this scenario, and Figure 7 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 exploits the filter mechanism envisioned 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, success,
blueprintsInfo)
choose preferred
blueprint from the
list (blueprintName)
(3) blueprintRequest
(confUserID,confObjID,
retrieve)
4) blueprintResponse
(confUserID,confObjID,
retrieve,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, success)

. .

Figure 7: Client Creation of Conference using Blueprints

1. Alice first sends a "blueprintsRequest" message to the conferencing system identified by the conference server discovery process. This request contains the "confUserID" of the user issuing the request (Alice's XCON-USERID) and the "xpathFilter" parameter 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", the conferencing system would first authenticate Alice and then ensure 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" 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 conferencing system returns the "confInfo" associated with the specific blueprint as identified by the 'confObjID' in the "blueprintResponse" message.

5. Alice finally sends a "confRequest" with a "create" operation to the conferencing system 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" message with a "create" operation, the conferencing system uses the received blueprint to clone a conference, allocating a new XCON-URI (again called "confObjID*" in the example). The conferencing server then sends a "confResponse" message including the new "confObjID*" associated with the newly created conference instance. 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)

```
<?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="xcon: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)

```
   <?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>success</response-code>
        <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 ("success", "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>success</response-code>
      <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="audioLabel"></xcon:floor>
                    <xcon:floor id="videoLabel"></xcon:floor>
              </xcon:conference-floor-policy>
            </xcon:floor-information>
          </blueprintInfo>
        </ccmp:blueprintResponse>
      </ccmpResponse>
    </ccmp:ccmpResponse>
```

5. confRequest/create message (Alice clones the "AudioRoom" 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 ("success", 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>success</response-code>
       <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="11"/>
                      <xcon:floor id="12"/>
                    </xcon:conference-floor-policy>
```

```
                </xcon:floor-information>
            </confInfo>
        </ccmp:confResponse>
    </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 8: Create Conference (Blueprint) Detailed Messaging

6.3. 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" parameter for the conference to be created. The request also includes the "confUserID" of the requesting entity.

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

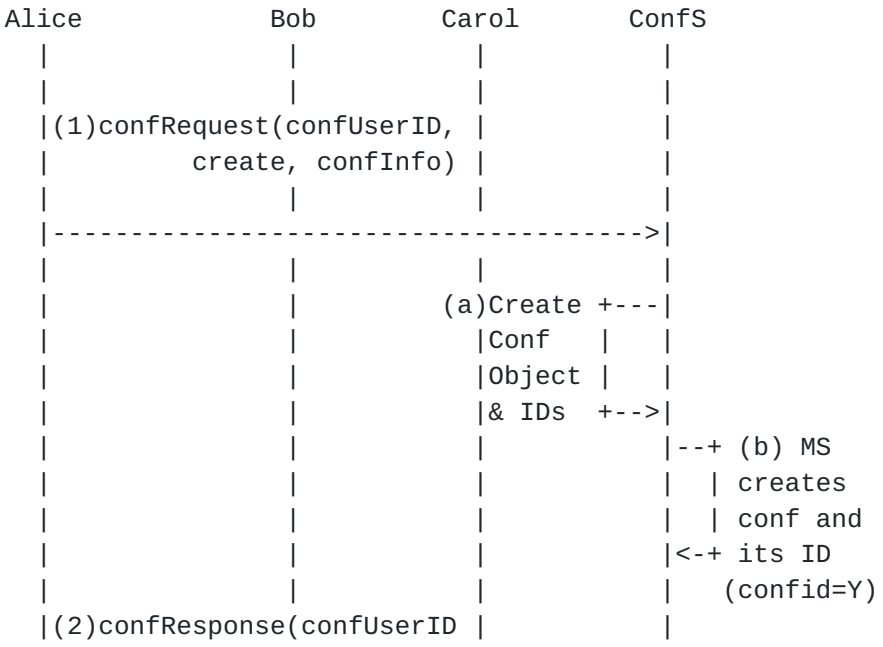
This "confInfo" parameter must comply of course with the Data Model specification. This means that its "entity" attribute is mandatory, and cannot be missing in the document. Nevertheless, considering in this example the client is actually requesting the creation of a new conference, which doesn't exist yet, this "entity" attribute cannot be set to a valid value. This is related to an issue already anticipated in Section 5.1. To cope with this, the CCMP protocol fosters the use of a wildcard placeholder: this placeholder ("xcon:AUTO_GENERATE_1@example.com" in the example) has the only aim of making the "confInfo" element compliant with the Data Model, and would subsequently be replaced by the conferencing system with the actual value. This means that, as soon as the conferencing system actually creates the conference, a valid "entity" value is created for it as well, which would take the place of the wildcard when completing the actual conference object provided by the client.

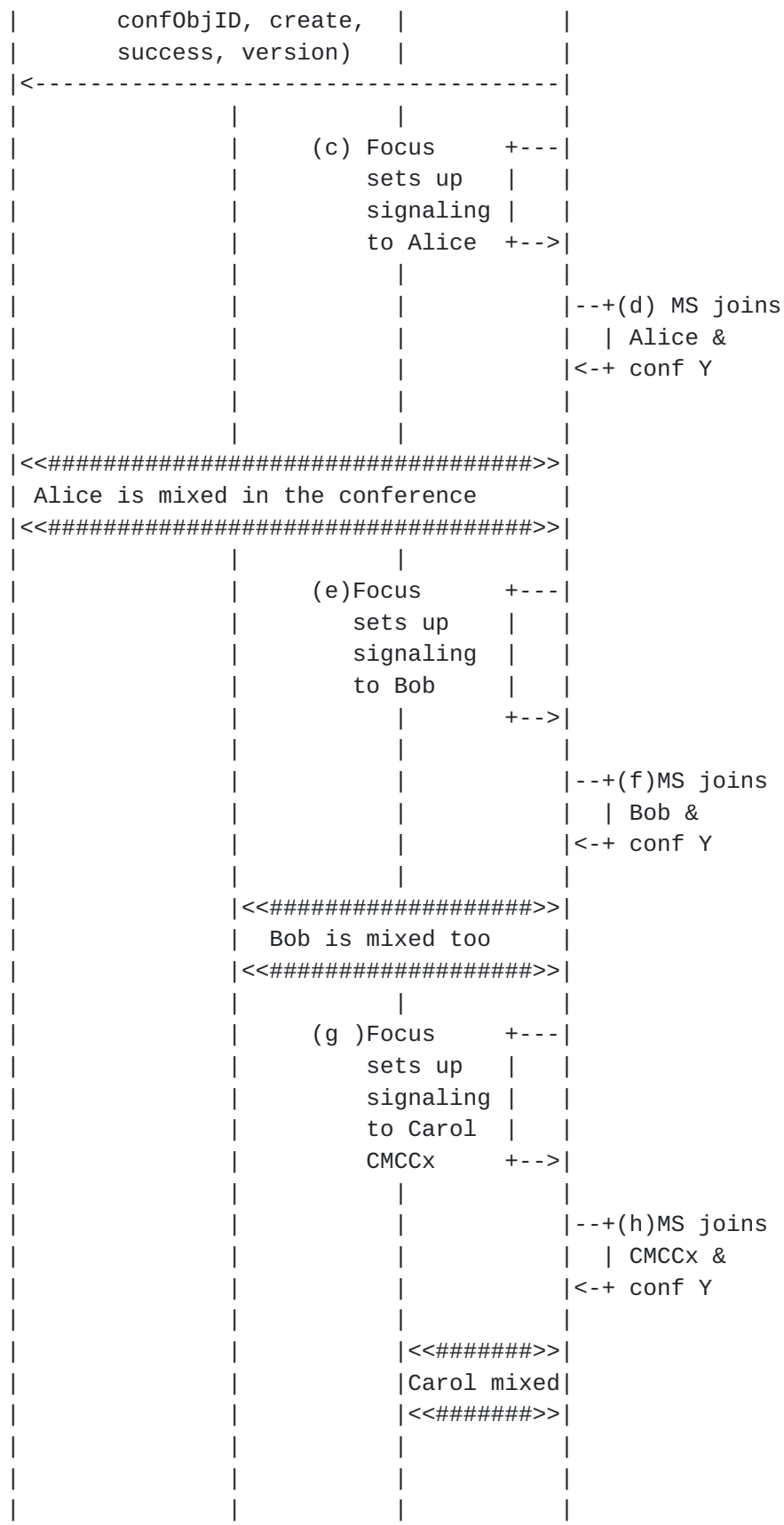
To give a flavour of what could be added to the conference object, we assume Alice is also interested in providing scheduling-related information. So, in this example, Alice also specifies by the <conference-time> element included in the "confInfo" that the conference she wants to create has to occur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

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 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, we choose not to return the created conference object in the successful "confResponse" in the "confInfo" parameter. Nevertheless, Alice could still retrieve the actual conference object by issuing a "confRequest" with a "retrieve" operation on the returned "confObjID". Such a request would show how, as we anticipated at the beginning of this section, the "entity" attribute of the conference object in "confInfo" is replaced with the actual information (i.e. "xcon:6845432@example.com").

Just for the sake of completeness and to provide the reader with some insight about how the CCMP conferencing server might interact with other related components, this example also assumes that the conference is activated upon creation: i.e., the "method" attribute is set to "dial out" for this client based on the particular conferencing systems default. This results (as shown in the ladder diagram) in Alice, Bob and Carol being called by the conferencing system. Just as before, this is not to be considered mandatory, since it depends on the implementation choices of the framework.




confObjID, create,
success, version)
(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
Bob is mixed too
(g)Focus sets up signaling to Carol CMCCx
(h)MS joins CMCCx & conf Y
Carol mixed

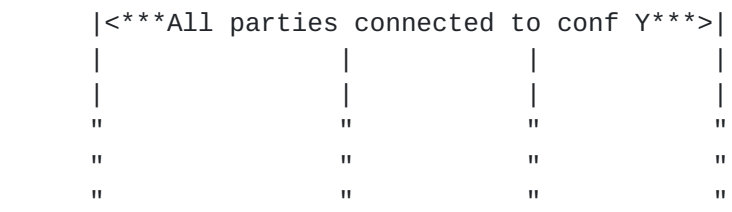


Figure 9: Create Basic Conference from user provided conference-info

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

```
  <?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 ("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>success<response-code>
      <version>1</version>
     </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 10: Create Basic Conference Detailed Messaging

6.4. 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 pre-defined 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 the "confUserID" and a specific "confObjID", from which a new conference is to be created by cloning an existing conference.

An example of how a client can create a conference based on a blueprint is provided in Section 6.2. 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 specifying querying for different types of conference objects supported by the specific conferencing system. For example, 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.

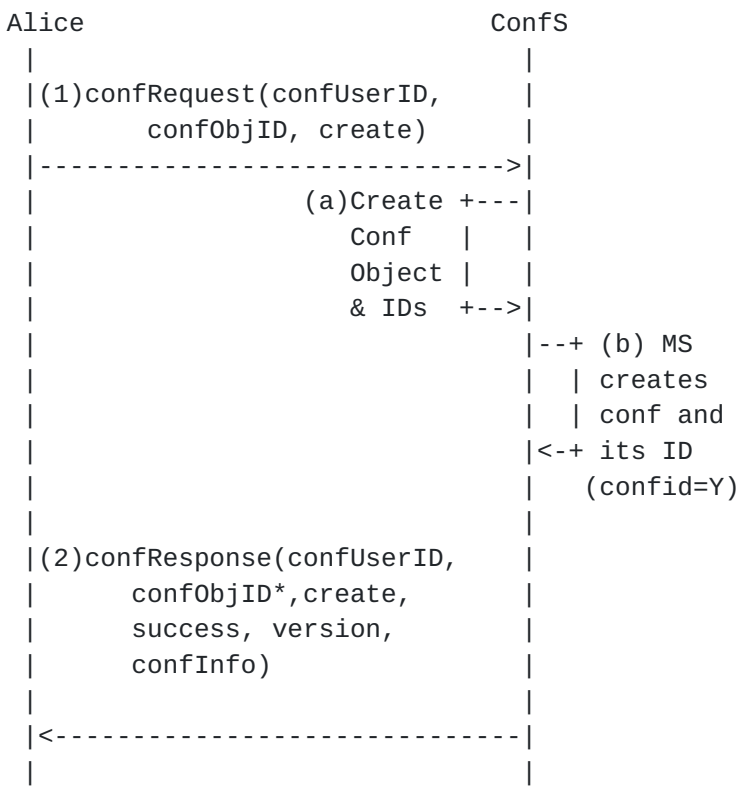


Figure 11: Create Basic Conference - Clone

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 "confObjID" in the request.

2. Upon receipt of the confRequest message containing a "create" operation and "confObjID", the conferencing system ensures that the "confObjID" received is valid. The conferencing system determines the appropriate read/write access of any users to be added to a conference based on this "confObjID" (using membership, roles, etc.). The conferencing system uses the received "confObjID" to clone a conference reservation. The conferencing system also reserves or allocates a new "confObjID" (called "confObjID*" in Figure 11) 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 conferencing system 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

```
<?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>
   </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>success</response-code>
     <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>
                <xcon:cloning-parent>
                    xcon:6845432@example.com
                </xcon:cloning-parent>
                 <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: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="11"/>
                   </xcon:conference-floor-policy>
                </xcon:floor-information>
            </confInfo>
        </ccmp:confResponse>
    </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 12: Create Basic Conference (Clone) Detailed Messaging

7. Conference Users Scenarios and Examples

Section 6 showed examples describing the several different ways a conference might be created using CCMP. This section instead 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 6.

7.1. 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 on.

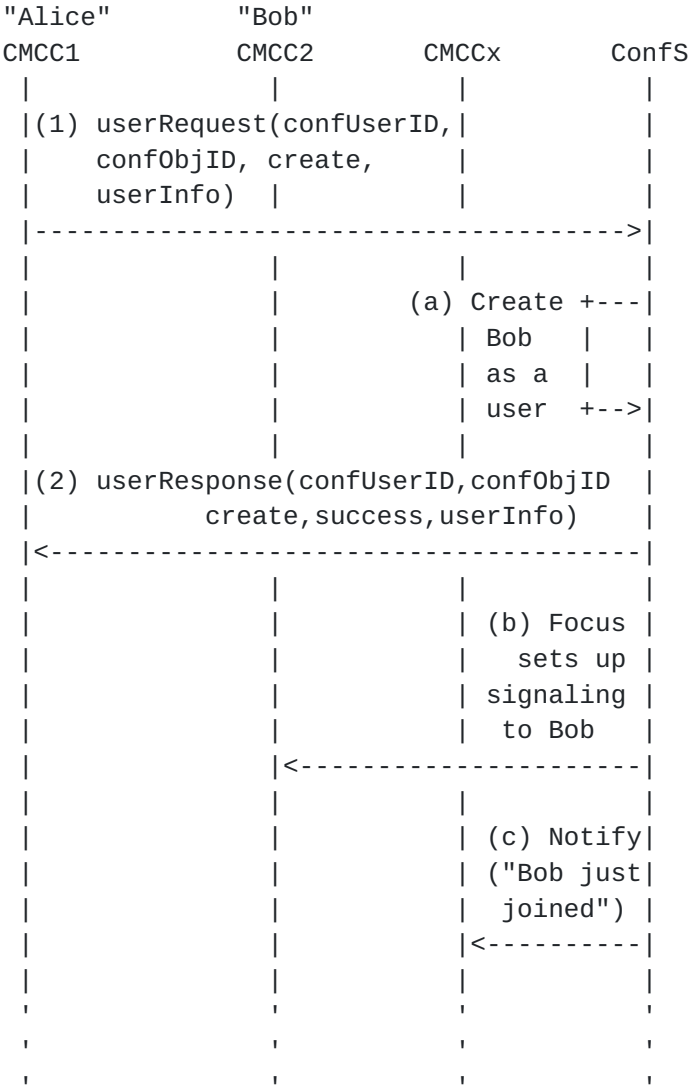


Figure 13: Client Manipulation of Conference - Add a party

1. Alice sends a userRequest message with an operation of "create" to add Bob to the specific conference as identified by the "confObjID". The "create" operation also makes sure that Bob is created as a user in the whole conferencing system. This is done by adding 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 Conference User Identifier is created for Bob, and the "userInfo" is used to update the conference object accordingly. As already seen in Section 6.3, a placeholder wildcard ("xcon-userid:AUTO_GENERATE@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 on;

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 part of the "entity" element 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. We again remind that 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 5.3).

To conclude the overview on this scenario, 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@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 ("success": 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>success</response-code>
       <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 14: Add Party Message Details

7.2. 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 as well, since it would involve an almost identical CCMP message flow anyway. Although, in case that any external floor control is involved, whether or not a particular conference 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: i.e., 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 15 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, whose role is "moderator" of the conference, wants to mute Bob on a medium-size 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.

From a protocol point of view, muting/unmuting an user basically consists in updating the conference object by modifying the settings related to the target user's media streams. Specifically, Bob's "userInfo" must be updated by modifying its audio <endpoint> information (e.g. setting it to "recvonly" in case of muting), identified by the right media id.

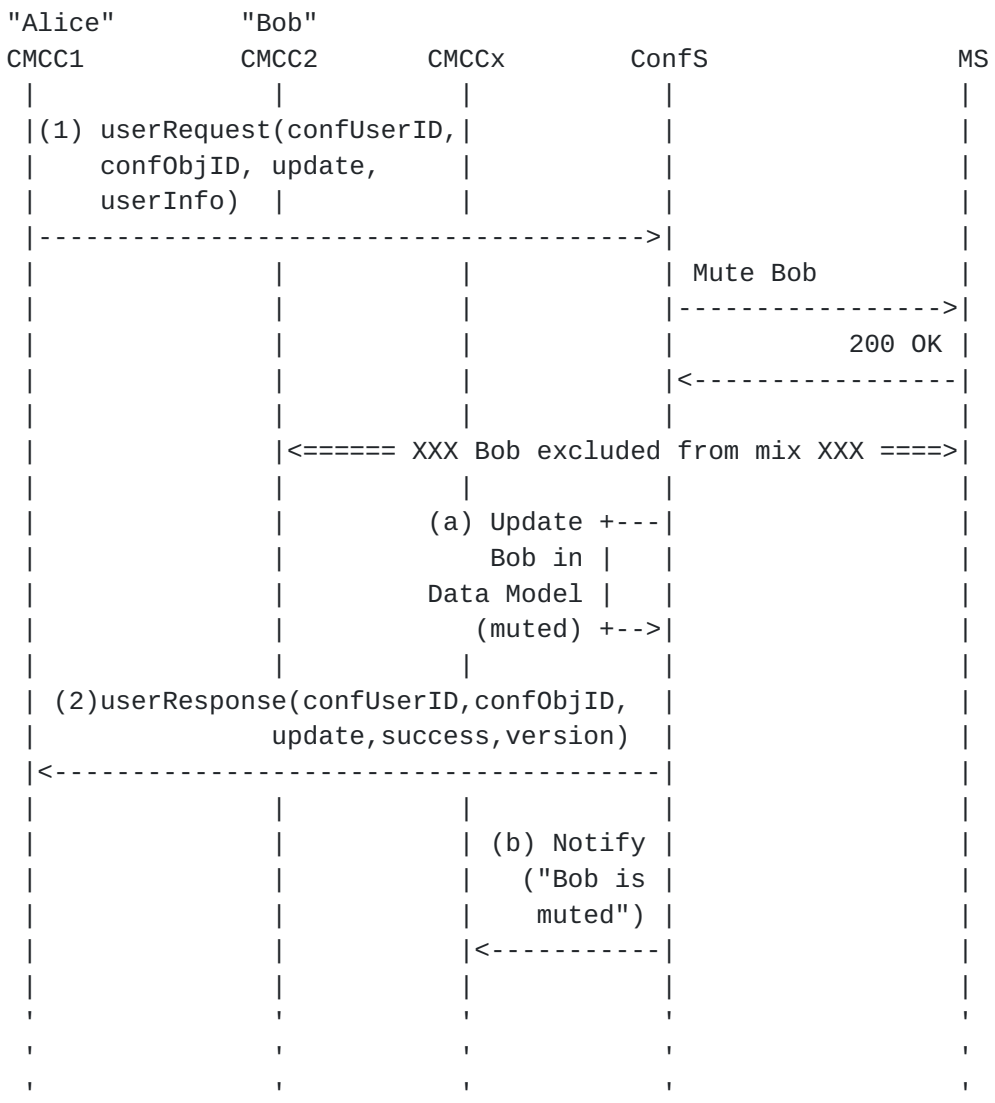


Figure 15: Client Manipulation of Conference - Mute a party

1. Alice sends a userRequest message with an "update" operation and the userInfo with the "status" field in the "media" element for Bob set to "revconly". The Conference Server ensures that Alice has the appropriate authority based on the policies associated with that specific conference object to perform the operation and updates the "userInfo" in the conference object reflecting that Bob's media is not to be mixed with the conference media. In case 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 to enforce the decision. An example describing a possible way of dealing with such a situation using the Media Server Control architecture is described in

[I-D.ietf-mediactrl-call-flows], at "Simple Bridging: Framework Transactions (2)".

2. A userResponse message with a response-code of "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">
      <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 ("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-user-response-message-type">
      <confUserID>xcon-userid:Alice@example.com</confUserID>
      <confObjID>xcon:8977878@example.com</confObjID>
      <operation>update</operation>
      <response-code>success</response-code>
          <version>7</version>
      <ccmp:userResponse/>
  </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 16: Mute Message Details

7.3. Conference Announcements and Recordings

This section deals with features that are typically required in a conferencing system, that are 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 7.1) it is an interesting scenario to address to see how the several components of an XCON-compliant architecture interact with each other to make it happen.

In this example, as shown in Figure 17 Alice is joining Bob's conference that requires that she first enters a pass code. After successfully entering the pass code, 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 [I-D.ietf-mediactrl-call-flows].

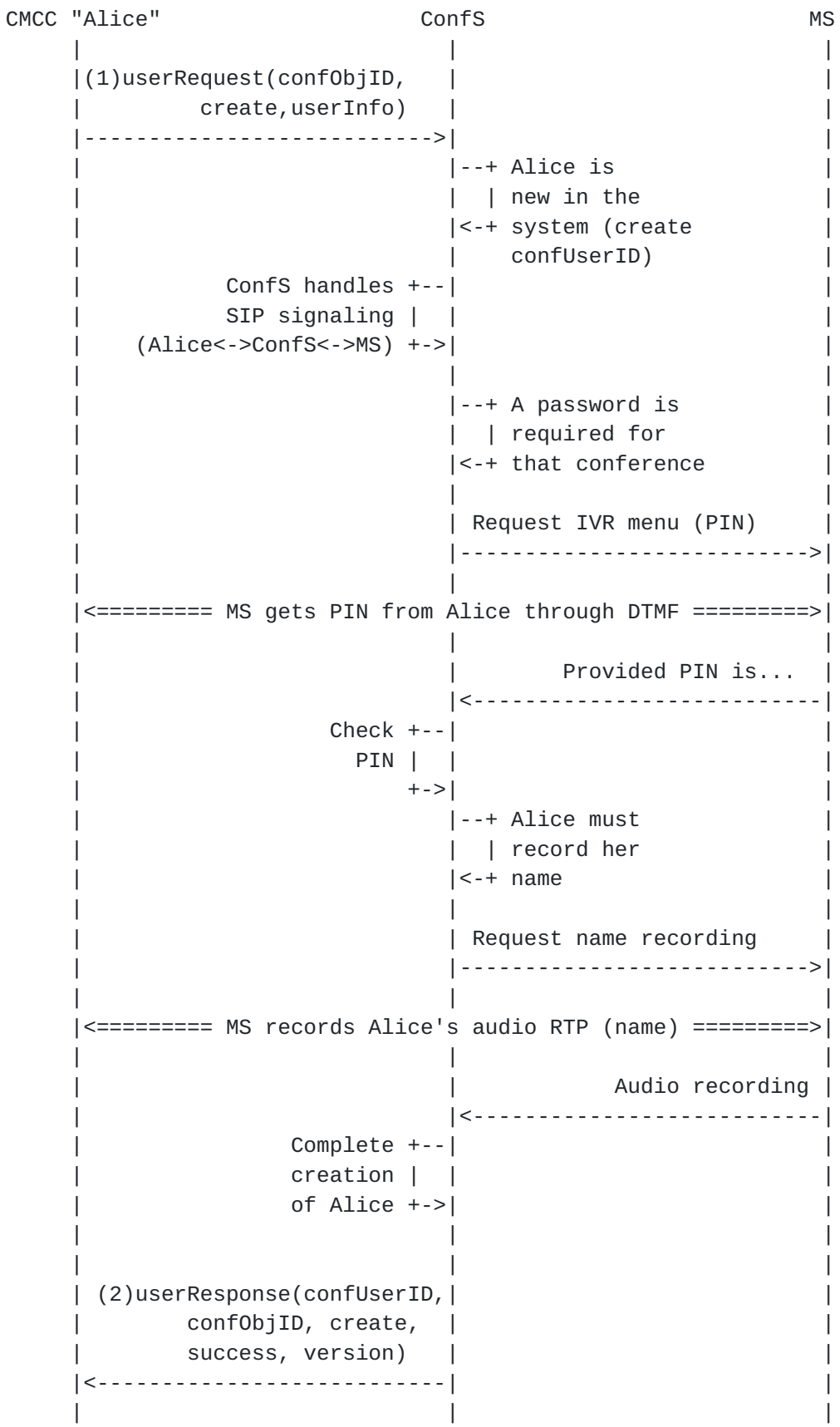

CMCC "Alice"
ConfS
MS
(1)userRequest(confObjID, create,userInfo)
--+ Alice is new in the
<-+ system (create confUserID)
ConfS handles SIP signaling (Alice<->ConfS<->MS)
--+ A password is required for
<-+ that conference
Request IVR menu (PIN)
<========= MS gets PIN from Alice through DTMF =========>
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
(2)userResponse(confUserID, confObjID, create, success, version)

Figure 17: Recording and Announcements

1. Upon receipt of the userRequest from Alice to be added to Bob's conference, the conferencing system 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 conferencing system then connects to a server which prompts and records Alice's name. The conferencing system 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 Conference User Identifier (i. e. an 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 which includes the "confUserID" 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 (Alice - a new conferencing system client - 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@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 ("success": Alice is provided with a new xcon-userid and 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:bobConf@example.com</confObjID>
        <operation>create</operation>
        <response-code>success</response-code>
        <version>5</version>
        <ccmp:userResponse/>
    </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 18: Announcement Messaging Details

7.4. Monitoring for DTMF

Conferencing systems often also need the capability to monitor for 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 often also exploited to achieve interaction between participants and the conference 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 17. A typical way for the conferencing system to be aware of all the DTMF interactions within the context of conferences it is responsible for, is making use of the MEDIACTRL architecture for what regards media manipulation. Examples in that sense (specifically for what concerns DTMF interception in conference instances) are presented in [I-D.ietf-mediactrl-call-flows].

7.5. Entering a password-protected conference

Some conferences may envision a password to be provided by a user who wants to manipulate the relative conference objects (e.g. join, update, delete) via CCMP. Such a password would be included in the <conference-password> element related to the conference XCON-URI in the appropriate <conference-uris> entry and must be then included in the apposite "conference-password" field in the CCMP request addressed to that conference.

In the following CCMP transactions, it is depicted a scenario in which 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 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 conferencing server detects that the indicated conference is not joinable without providing the relative pass code. Then a userResponse message with "confPasswordRequired" response-code is returned to Alice to indicate that her join has been refused and that she has to recast her request including the appropriate conference password in order to participate.

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

4. The conferencing server checks the provided password and then adds Alice to the protected conference. After that, a userResponse with a "success" response-code 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 ("passwordRequired")

```
<?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>
        <ccmp:response-code>confPasswordRequired</ccmp:response-code>
        <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 ("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-user-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
        <operation>create</operation>
        <response-code>success</response-code>
        <version>10</version>
        <ccmp:userResponse/>
    </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 19: Password-protected conference join messages details

8. Sidebars Scenarios and Examples

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

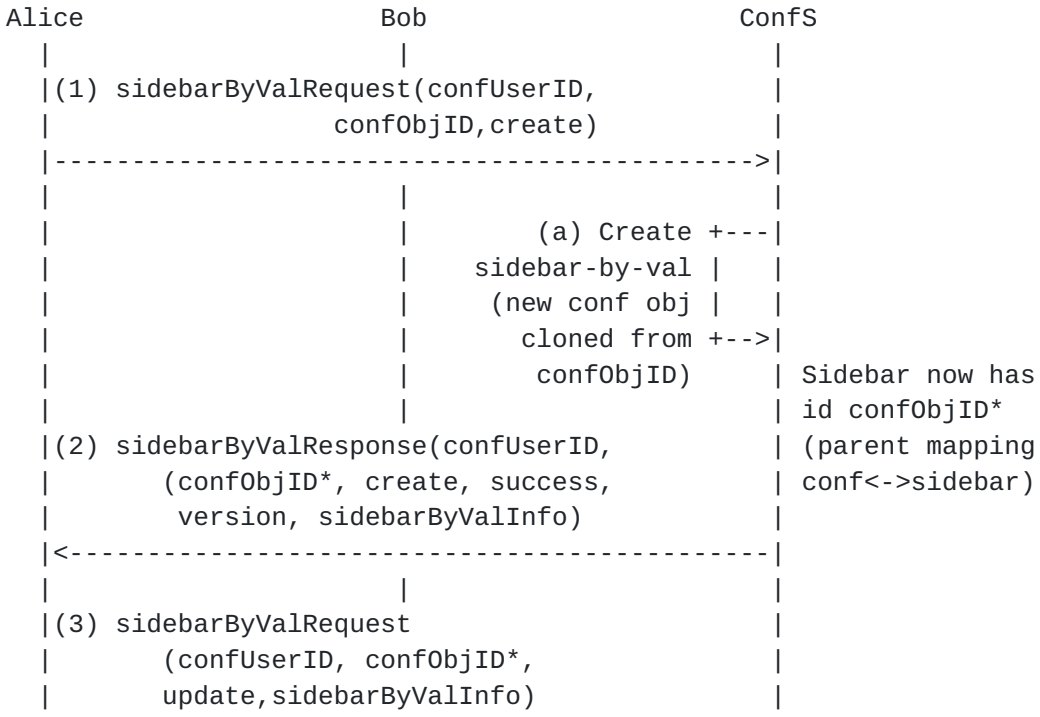
In fact, a feature typically required in conferencing systems is the ability to create sidebars. A sidebar is basically a child conference that usually includes a subset of the participants of the parent conference, and a subset of its media as well. Sidebars are

typically required whenever some of the participants in a conference want to discuss privately about something, without interfering with the main conference.

This section deals with some scenarios that typically envisage the use of a sidebar, like whispering, private messages and coaching scenarios. The first subsections, anyway, present some examples of how a generic sidebar can be created, configured and managed.

8.1. Internal Sidebar

Figure 20 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 conferencing system 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.



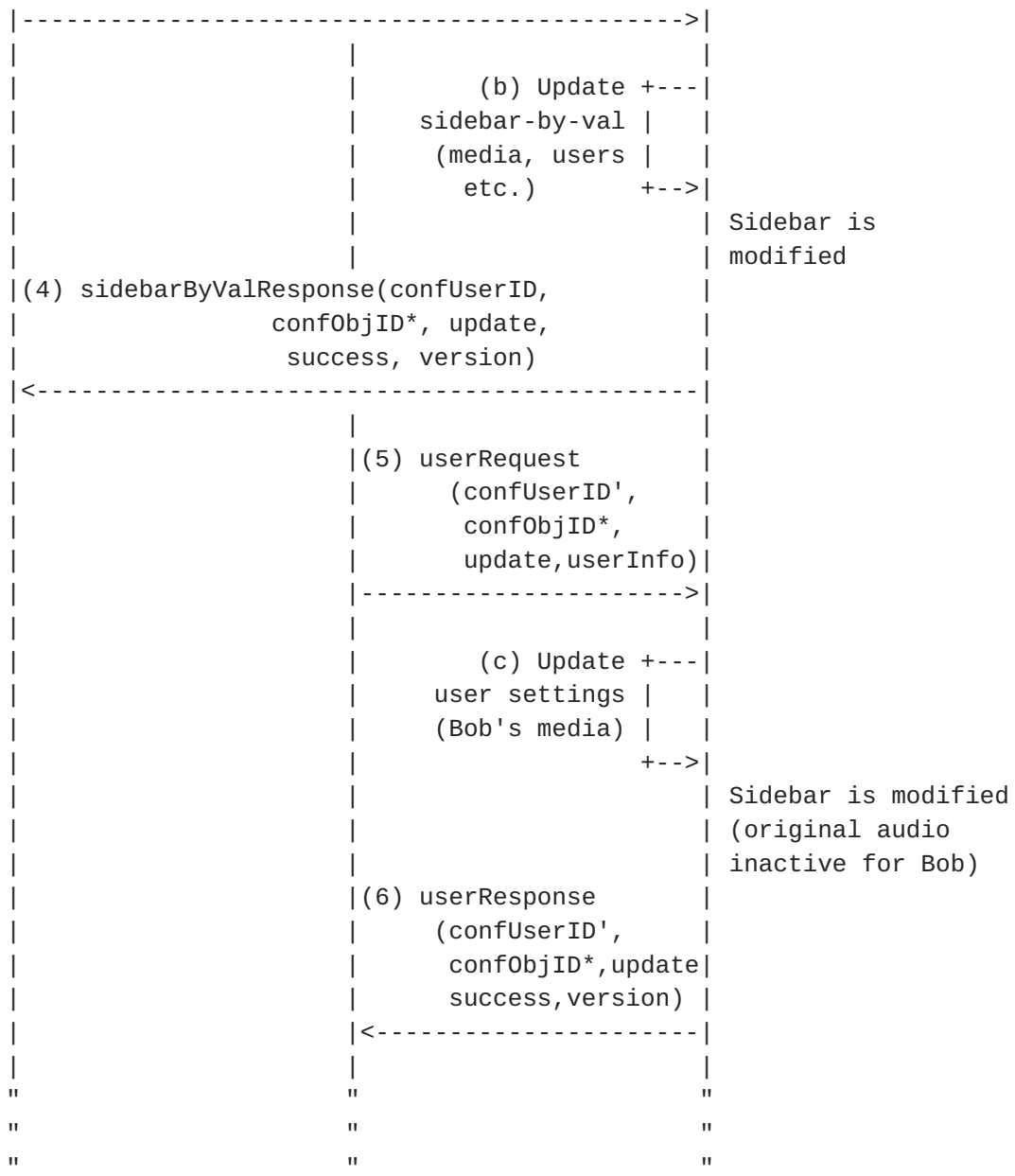


Figure 20: Client Creation of a Sidebar Conference

1. Upon receipt of CCMP sidebarByValRequest message to create a new sidebar-conference based upon the confObjID received in the request, the conferencing system uses the confObjID to clone a conference reservation for the sidebar. The sidebar reservation is NOT independent of the active conference (i.e., parent). The conferencing system also allocates a new XCON-URI for that sidebar to be used for any subsequent protocol requests from any of the members of the conference. The new sidebar identifier ("confObjID*" in Figure 20) 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 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 confObjID is updated, the conference server sends a sidebarByValResponse 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.

That said, let's consider the following conference object:

```
<?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 21: Conference with Alice, Bob and Carol

This is the representation of the conference the sidebar is going to be created in. As such, it will be used by the conferencing system in order to create the new conference object associated with the sidebar. In fact, the sidebar creation happens through a cloning of the parent conference. Once the sidebar is created, an "update" 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 ("success", 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>success</response-code>
          <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>
                  <xcon:sidebar-parent>
                       xcon:8977878@example.com
                  </xcon:sidebar-parent>
                  <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>
                </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 ("success", 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>success</response-code>
          <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 ("success", Bob's preferences setted)

```
<?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>success</response-code>
      <version>3</version>
      <ccmp:userResponse/>
  </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 22: Internal Sidebar Messaging Details

8.2. External Sidebar

Figure 23 provides an example of a different approach towards sidebar. 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 conferencing system 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.

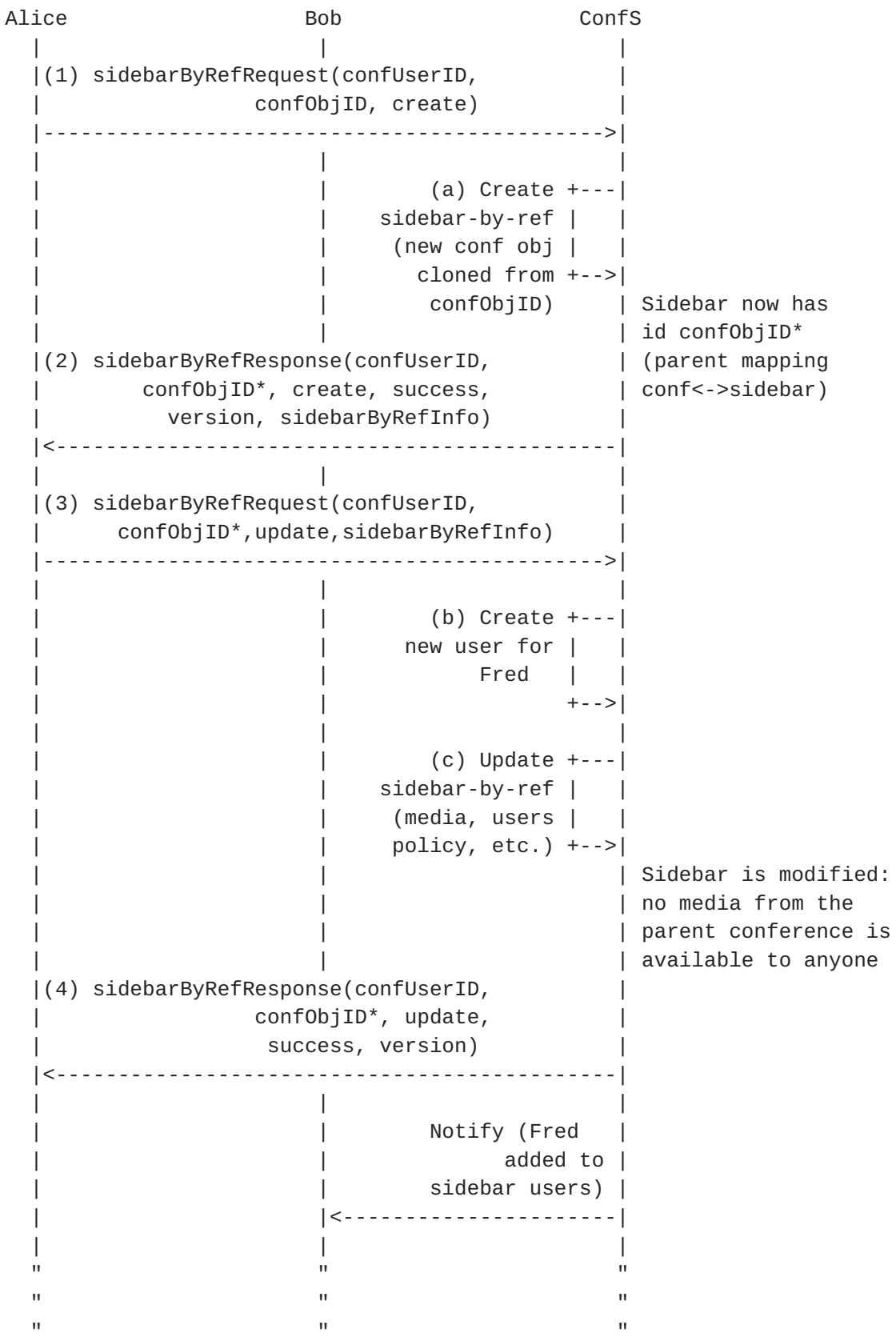

Alice
Bob
ConfS
(1) sidebarByRefRequest(confUserID, confObjID, create)
(a) Create sidebar-by-ref (new conf obj cloned from confObjID)
Sidebar now has id confObjID* (parent mapping conf<->sidebar)
(2) sidebarByRefResponse(confUserID, confObjID*, create, success, version, sidebarByRefInfo)
(3) sidebarByRefRequest(confUserID, confObjID*,update,sidebarByRefInfo)
(b) Create new user for Fred
(c) Update sidebar-by-ref (media, users policy, etc.)
Sidebar is modified: no media from the parent conference is available to anyone
(4) sidebarByRefResponse(confUserID, confObjID*, update, success, version)
Notify (Fred added to sidebar users)

Figure 23: 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 conferencing system 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 conferencing system, as before, allocates a conference ID (confObjID*) to be used for any subsequent protocol requests toward the sidebar reservation. The mapping between the sidebar conference ID and the one associated with the main conference is mantained by the conferencing system and it is gathered from the c<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 system by means of a new "sidebarByRefRequest" message, with an "update" operation.

3. Alice sends the aforementioned "sidebarByRefRequest" to update the information in the reservation and to create an active conference. Upon receipt of the "sidebarByRefRequest" with an operation of "update", the conferencing system ensures that Alice has the appropriate authority based on the policies associated with that specific conference object to perform the operation. The conferencing system also validates the updated information in the reservation. Since Fred is a new user for this conferencing system, a conference user identifier 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" method set as the target for him) 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 ("success", 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>success</response-code>
      <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>
                  <xcon:sidebar-parent>
                      xcon:8977878@example.com
                  </xcon:sidebar-parent>
                  <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>
            </info:users>
        </sidebarByRefInfo>
    </ccmp:sidebarByRefResponse>
  </ccmpResponse>
</ccmp:ccmpResponse>
```

3. sidebarByRefRequest/update message (Alice updates 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: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 & 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 ("success", sidebar updated)

```
  <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>success</response-code>
        <version>2</version>
        <ccmp:sidebarByRefResponse/>
    </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 24: External Sidebar Messaging Details

8.3. Private Messages

The case of private messages can be handled as a sidebar with just two participants, similarly to the example in Section 8.1. 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 conferencing system the creation of a private chat room within the main conference context (presented in Figure 21) in which the involved partecipants are just Bob and herself. This can be achieved through the following CCMP transaction (Figure 25).

1. Alice forwards a sidebarByValRequest/create to the Conferencing Control 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 to the conferencing system. Please note that, unlike the previous sidebar examples, in this case a comnpletely 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 Control 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 envisioned in Section 8.1 and Section 8.2, the sidebar is created on the basis of the user provided conference information (as anticipated before). However, the parent relationship between the main conference and the newly created sidebar is still mantained by the conferencing system (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 ("success", 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>success</response-code>
            <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>
                    <xcon:sidebar-parent>
                         xcon:8977878@example.com
                    </xcon:sidebar-parent>
                    <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>
                </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 25: Sidebar for Private Messages scenario

8.4. Observing and Coaching

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

An example of observing and coaching is shown in figure Figure 27. 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 26):

```
<?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 26: A call-center conference object example

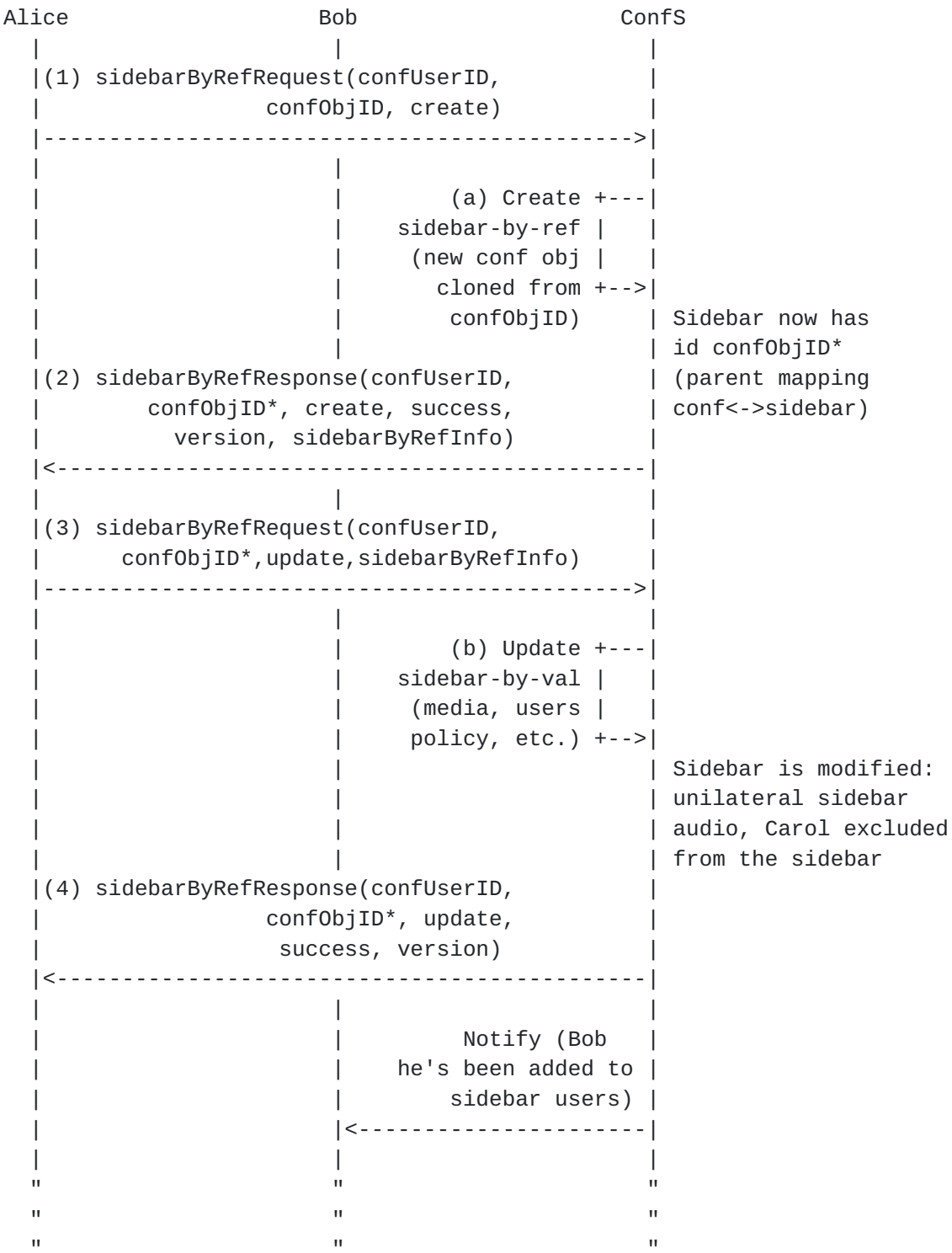


Figure 27: Supervisor Creating a Sidebar for Observing/Coaching

1. Upon receipt of the sidbarByRefRequest/create from Alice to "create" a new sidebar conference from the confObjID received in the request, the conferencing system uses the received active conference to clone a conference reservation for the sidebar. The conferencing system also allocates a conference ID to be used for any subsequent protocol requests from any of the members of the conference. The conferencing system maintains the mapping between this conference ID and the confObjID associated with the sidebar reservation through the conference instance. The conference server sends a sidebarByRefResponse message with the new confObjID and relevant 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.

3. Upon receipt of the sidebarByRefRequest message with an "update" operation, the conferencing system 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 requestor 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 <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 conferencing control server is able to recognize those dummy values as place-holders.

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:sidebarsByRefRequest/>
    </ccmpRequest>
</ccmp:ccmpRequest>
```

2. sidebarByRefResponse/create message ("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-sidebarByRef-response-message-type">
        <confUserID>xcon-userid:alice@example.com</confUserID>
        <confObjID>xcon:8971313@example.com</confObjID>
        <operation>create</operation>
        <ccmp:response-code>success</ccmp:response-code>
        <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>
                    <xcon:sidebar-parent>
                        xcon:8971313@example.com
                    </xcon:sidebar-parent>
                    <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>
                </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>
                    <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>
                    <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>
                    </user>
                    <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>
                    </user>
                </info:users>
            </sidebarByRefInfo>
        </ccmp:sidebarByRefRequest>
    </ccmpRequest>
  </ccmp:ccmpRequest>
```

4. sidebarByRefRequest/update message ("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-sidebarByRef-response-message-type">
        <confUserID>xcon-userid:alice@example.com</confUserID>
        <confObjID>xcon:8971313@example.com</confObjID>
        <operation>update</operation>
        <ccmp:response-code>success</ccmp:response-code>
        <version>2</version>
        <ccmp:sidebarByRefResponse/>
    </ccmpResponse>
  </ccmp:ccmpResponse>
```

Figure 28: Coaching and Observing Messaging details

9. 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 6.

9.1. Removing a Party

Figure 29 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.

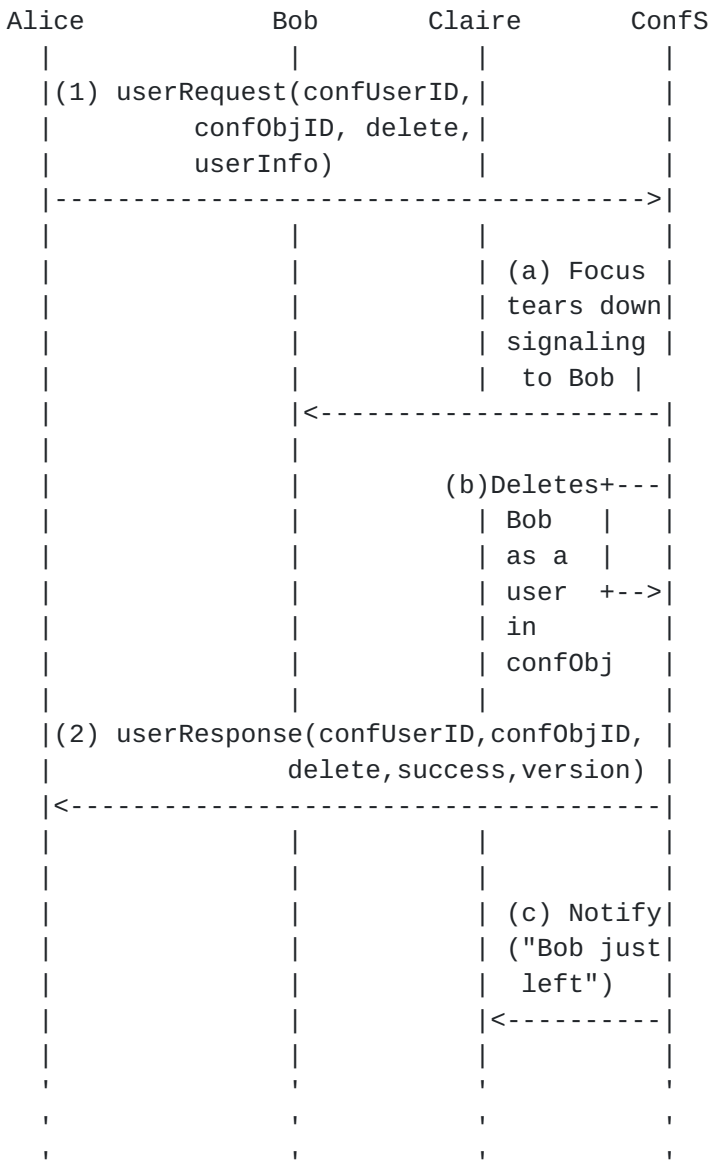


Figure 29: Client Manipulation of Conference - Remove a party

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 ("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-user-response-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:8977794@example.com</confObjID>
            <operation>delete</operation>
        <response-code>success</response-code>
        <version>17</version>
        <ccmp:userResponse/>
    </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 30: Removing a Participant Messaging Details

9.2. Deleting a Conference

In this section, an example of a successful conference deletion is provided (Figure 31).

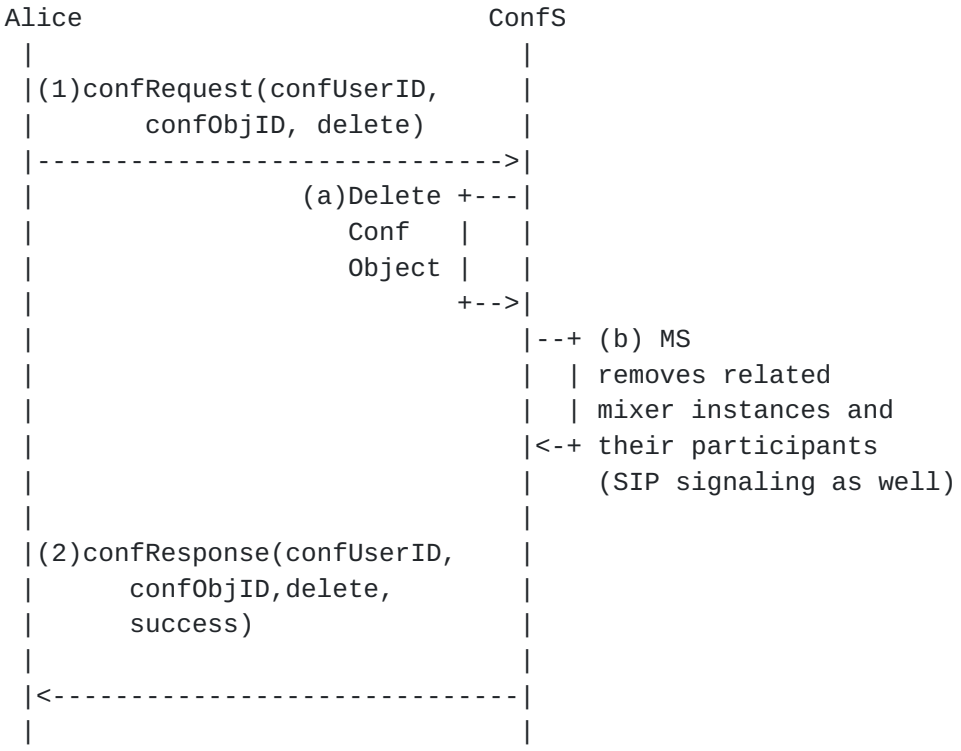


Figure 31: Deleting a conference

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 conferencing server instigates the process to delete the conference object, disconnetting participants and removing associated resources such as mixer instances. Then, the conference server returns a confResponse message to Alice with "success" 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 ("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>success</response-code>
        <ccmp:confResponse/>
    </ccmpResponse>
</ccmp:ccmpResponse>
```

Figure 32: Deleting a Conference Messaging Details

10. IANA Considerations

This document has no IANA considerations.

11. Security Considerations

The security considerations applicable to the implementation of these call flows is documented in the XCON Framework, with additional

security considerations documented in the CCMP document. Where applicable, statements with regards to the necessary security are discussed in particular flows, however, since this is only an informational document, readers are strongly recommended to carefully consider the security considerations defined in the XCON Framework and the CCMP document.

12. Change Summary

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

The following are the major changes between the 02 and the 03 versions of the draft:

- updated the call flows in order to take into account the changes on CCMP;

- added a completely new introductory section, addressing the protocol in general, the data model constraints, transport-related information, and notifications in a practical way;

- reorganized the chapters, grouping user-related scenarios in an users section, and doing the same for sidebars;

- added more verbose text to almost every section of the document;

The following are the major changes between the 01 and the 02 versions of the draft:

- updated the call flows in order to take into account the new versioning mechanism of the CCMP;

- clarified, per agreement in Stockholm, that cloning from a blueprint does not need a cloning-parent to be made available in the response;

- clarified that BFCP and CCMP-based media control are neither in conflict nor one the wrapper of the other; they act at different levels, and when both are involved, it is required that both grant a resource before it can be used by an interested participant;

- changed all the domains involved in the flows to make them compliant with [RFC2606];

- clarified that a successful creation of a new conference object may or may not contain the whole confInfo object in the response; in case it doesn't, a retrieve of the updated object can be achieved by issuing a confRequest/retrieve;

- clarified that the scenario in Section 7.3 only involves CCMP in adding the user to a conference; this includes requiring the use of a password only in adding the user to the conference object; the actual request for PIN/Password when joining thw conference is handled by means of out-of-band mechanisms (in this case at the media level, with the help of the MEDIACTRL framework);

- added and corrected Sidebars-related scenarios;

- added flows for some previously missing scenarios: Private Message/Whisper, Coaching Scenario, Removing a Party, Deleting a Conference;

-

The following are the major changes between the 00 and the 01 versions of the draft:

- Updates to reflect change of CCMP to HTTP transport model.

The following are the major changes between the individual 01 version to the WG 00:

- Updates to reflect most recent version of CCMP, including parameter names, etc.

- Added protocol details to many of the examples.

- Editorial: Simplifying intro, terms, etc.

13. 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.

14. References

14.1. Normative References

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

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

[I-D.ietf-xcon-ccmp] Barnes, M., Boulton, C., Romano, S., and H. Schulzrinne, "Centralized Conferencing Manipulation Protocol", draft-ietf-xcon-ccmp-05 (work in progress), December 2009.

14.2. Informative References

[RFC2606] Eastlake, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, June 1999.

[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.

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

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

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

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

[I-D.ietf-xcon-event-package] Camarillo, G., Srinivasan, S., Even, R., and J.

Urpalainen, "Conference Event Package Data Format Extension for Centralized Conferencing (XCON)", draft-ietf-xcon-event-package-01 (work in progress), September 2008.

[I-D.ietf-xcon-common-data-model]
Novo, O., Camarillo, G., Morgan, D., and J. Urpalainen, "Conference Information Data Model for Centralized Conferencing (XCON)", draft-ietf-xcon-common-data-model-16 (work in progress), February 2010.

[I-D.ietf-mediactrl-call-flows]
Amirante, A., Castaldi, T., Miniero, L., and S. Romano, "Media Control Channel Framework (CFW) Call Flow Examples", draft-ietf-mediactrl-call-flows-03 (work in progress), February 2010.

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

[I-D.ietf-mediactrl-mixer-control-package]
McGlashan, S., Melanchuk, T., and C. Boulton, "A Mixer Control Package for the Media Control Channel Framework", draft-ietf-mediactrl-mixer-control-package-10 (work in progress), January 2010.

[I-D.boulton-xcon-session-chat]
Barnes, M., Boulton, C., and S. Loreto, "Chatrooms within a Centralized Conferencing (XCON) System", draft-boulton-xcon-session-chat-04 (work in progress), July 2009.

Authors' Addresses

Mary Barnes
Nortel
2201 Lakeside Blvd
Richardson, TX

Email: mary.barnes@nortel.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