XCON
Internet-Draft
Expires: January 14, 2005

H. Khartabil
P. Koskelainen
A. Niemi
Nokia

July 16, 2004

The Conference Policy Control Protocol (CPCP)
draft-ietf-xcon-cpcp-xcap-01

Status of this Memo

Copyright Notice

Abstract

This document describes the Conference Policy Control Protocol (CPCP). It specifies an Extensible Markup Language (XML) Schema that enumerates the conference policy data elements that enable a user to define a conference policy. It also defines an XML Configuration Access Protocol (XCAP) application usage that may be used to store and manipulate a conference policy.

Table of Contents

1. Introduction

The SIP conferencing framework [13] defines the mechanisms for multi-party centralized conferencing in a SIP environment.

Existing SIP mechanisms allow users, for example, to join and leave a conference, as described in [9]. A centralised server, called focus, can expel and invite users, and may have proprietary access control lists and user privilege definitions. This document defines an XML Schema in Section 4 that enumerates the conference policy data elements that enable a user to define a conference policy. In some cases, such as some ad-hoc scenarios described in [9], there is a static conference policy which is not changed or manipulated during a conference. This policy document may be given to a focus using a number of transports. Mechanisms such as a web page or a voice response system can also be used to manipulate conference policy data.

However, in many cases it is useful to have standardised means to manipulate conference policy elements such as access control lists. The requirements for such protocol are defined in [8].

Section 6 of this document describes one such protocol for the real-time manipulation of conference policy. An XML Configuration Access Protocol (XCAP) [10] application usage is defined which meets the requirements in [8] to store and manipulate a conference policy object.

XCAP has many advantages in its use for conference policy control protocol. It is a HTTP 1.1 based protocol that allows clients to read, write, modify and delete application data stored in XML format at a server. XCAP maps XML document elements and attributes to HTTP URIs that can be directly accessed by HTTP. One application area which has already adopted XCAP is the manipulation of event lists [11].

A focus conforming to this specification MUST support the XML object defined in Section 4 . For manipulation of the the XML object, the system MAY support the XCAP usage defined in Section 6.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

3. Terminology

This document uses terminology from [13]. Some additional definitions are introduced here.

Conference authorization policy (CAP)

Conference authorization policy consists of an unordered set of rules, which control the permissions and privileges that are given to conference participants.

Conference Policy Server (CPS)

Conference Policy Server. See [13]

Conference participant

Conference participant is a user who has an on-going session (e.g. SIP dialog) with the conference focus.

Floor control

Floor control is a mechanism that enables applications or users to gain safe and mutually exclusive or non-exclusive access to the shared object or resource in a conference.

Dial-Out List (DL)

Dial-out list (DL) is a list of users who the focus needs to invite to the conference.

Privileged user

A privileged user is a user that has the right manipulate parts or all of the conference policy settings.

CPS XCAP URI

The URI of the XCAP server that is used to create the conference. The URI construction is specified in [10]. It is referred to in XCAP as the host part.

Conference Policy URI

The URI of conference policy. In XCAP, it is the CPS XCAP URI along with the abs_path. It identifies the XML document. The URI construction is specified in [10].

4. Structure of a Conference Policy document

The conference policy document is an XML [6] document that MUST be well-formed and MUST be valid. Conference policy documents MUST be based on XML 1.0 and MUST be encoded using UTF-8. This specification makes use of XML namespaces for identifying conference policy documents and document fragments. The namespace URI for elements defined by this specification is a URN [3], using the namespace identifier 'ietf' defined by [4] and extended by [15]. This URN is:

urn:ietf:params:xml:ns:conference-policy

4.1 MIME Type for CPCP XML Document

The MIME type for the CPCP XML document is "application/ conference-policy+xml".

4.2 Conference Root

A conference policy document begins with the root element tag <conference>. Other elements from different namespaces MAY be present for the purposes of extensibility. Elements or attributes from unknown namespaces MUST be ignored. The conference policy is build up using the following:

- The <settings> element: This element is mandatory and contains various conference settings. It contains the conference URI(s) and the maximum number of participants. It can occur only once in the document.

- The <info> element: This element is optional and includes information describing the conference, e.g. for search purposes. This information can also be used in the session description when the focus is sending invitations. It can occur only once in the document.

- The <time> element: This optional element defines conference time information, namely elements defining start and stop times for a conference.

- The <authorization> element: This optional element is the conference authorisation rules. It contains rules for users who can dial into the conference, users who are blocked from dialling in, amongst others.

- The <dialout-list> element: This optional element is for the dial-out list. It contains URIs for users that the focus will

invite to the conference.

- The <refer-list> element: This optional element is for the refer list. It contains URIs for users that the focus will refer to the conference.

- The <security-control> element: This optional element is for security control. It contains conference security level and passwords.

- The <ms> element: This optional element contains the media streams to be used in the conference.

- The <fp> element: This optional element is for the floor control policy.

The elements are described in more detail in the forthcoming sections.

A user may create a new conference at the CPS by placing a new conference policy document at the CPS. Depending on server policy and user privileges, the CPS may accept the creation.

A conference can be deleted permanently by removing the conference policy from the CPS, which consequently frees the resources. When the user deletes a conference, the CPS MUST also delete all its sub-conferences ("sidebars") at a server. Conference sidebars have unique URIs at the server.

4.3 XML Document Description

4.3.1 Conference Settings

The mandatory <settings> element contains 2 sub-elements; the <conference-uri> element and the <max-participant-count> element.

<conference-uri> is a mandatory element. It can occur more than once to accommodate multiple signaling protocols. Once a conference URI

is set, it MUST NOT be changed or removed for the duration of the conference. Only one URI per protocol MUST be set. URIs can be added at any time.

<max-participant-count> is optional. It carries the maximum number of participants allowed in the conference. When the maximum number of participants threshold is reached, no new users are not allowed to join until the number of participants decreases again. If using SIP, the server can reject a request to join (INVITE) with a "480 Temporarily Unavailable" response. Alternatively, the sever may

implement a waiting queue.

<allow-sidebars> is an optional element with a boolean value indicating if sidebars are allowed in this conference or not. The default value, if omitted, is "true" indicating that sidebars are allowed.

<sidebar> is an element identifying a side bar. Multiple <sidebar> elements can occur indicating multiple sidebars. No <sidebar> elements appearing in a conference policy indicates that there are no sidebars currently for this conference. A <sidebar> element contains a mandatory 'id' attribute that uniquely identifies the sidebar. It also contains an <uri> element that hold the sidebar URI. It can occur more than once to accommodate multiple signaling protocols. Once a sidebar URI is set, it MUST NOT be changed or removed for the duration of the conference. Only one URI per protocol MUST be set. URIs can be added at any time.

A sidebar MAY have its own policy. This policy is created exactly in the same manner as any other conference. The <policy> element in the <sidebar> element points to such policy. If the <policy> element is omitted, the sidebar inherits the policy of the conference it is a sidebar of.

A conference is identified by one or more conference URIs, one for each call signaling protocol that is supported. There must be at least one URI for a conference. Conference URIs can be proposed by the creator of the conference policy, as it may be useful to have human-friendly name in some cases, or can be assigned by the CPS. If the creator has proposed a conference URI, the server needs to decide whether it accept the name proposed by the client or not. It does this determination by examining if the conference URI already exists or not. If it exists, the CPS rejects the request to create the conference with that conference URI. Similarly, the CPS rejects the request to create a conference with a conference URI for a signalling protocol it does not support.

A Conference URI can be SIP, SIPS, TEL, or any supported URI scheme. The CPS MAY assign multiple conference URIs to a conference, one for each call signaling protocol that it supports.

Sidebar URIs are subject to the same behaviour.

4.3.2 Conference Information

The optional <info> element includes informative conference parameters which may be helpful describing the purpose of a conference, e.g. for search purposes or for providing host contact

information. The <info> element MUST have a special attribute 'xml:lang' to specify the language used in the contents of this element as defined Section 2.12 of [6].

Each conference has an optional <subject> element, which describes the current topic in a conference. The optional <display-name> element is the display name of the conference, which usually does not change over time.

<free-text> and <keywords> are optional elements. They provide additional textual information about the conference. This information can be made available to potential conference participants by means outside the scope of this document. Examples of usage could be searching for a conference based on some keywords. The optional <web-page> element points to a URI where additional information about the conference can be found.

The optional <host-info> element contains several elements. It gives additional information about the user hosting the conference. This information can, for example, be included into the SDP fields of the SIP INVITE requests sent by the focus. The <uri> element is optional and can occur more than once.

4.3.3 Conference Time

The information related to conference time and lifetime is contained in the <time> element. The conference may occur for a limited period of time (i.e. bounded), or the conference may be unbounded (i.e. it does not have a specified end time). Bounded conferences may occur multiple times(e.g. on weekly basis).

The <time> element contains one or more <occurrence> elements each defining the time information of a single conference occurrence. Multiple <occurrence> elements MAY be used if a conference is active at multiple times; each additional <occurrence> element contains time information for a specific occurrence.

For each occurrence, the <mixing-start-time> element specifies when a conference media mixing starts. the <mixing-stop-time> element specifies the time a conference media mixing stops. If the <mixing-start-time> element is not present, it indicates that the

conference media mixing starts immediately. If the <mixing-stop-time> element is not present, it indicates that the conference occurrence is not bounded, i.e. permanent, though media mixing will not become active until the <mixing-start-time>. <mixing-start-time> and <mixing-stop-time> elements both have the mandatory 'require-participant' attribute. This attribute has one of 3 values: "none", "key-participant", and "participant". For mixing

start time, this attribute allows a privileged user to define when media mixing starts based on the latter of the mixing start time, and the time the first participant or key participant arrives. If the value is set to "none", mixing starts according to the mixing start time. For mixing stop time, this attribute allows a privileged user to define when media mixing stops based on the earlier of the mixing stop time, and the time the last participant or key participant leaves. If the value is set to "none", mixing stops according to the mixing stop time.

Users can be allowed to join a conference before the media mixing time starts and after a certain time. A conference privileged user can indicate the time when users can join by populating the <can-join-after> element. Similarly, a conference privileged user can define the time after which new users are not allowed to join the conference anymore. This is done by populating the <must-join-before> element.

It is possible to define the time when users or resources on the dial-out list and on the refer-list are requested to join the conference by using the <request-users> element. It is also possible to define that the users and resources on the dial-out list and the refer-list are requested to join the conference only after the first a participant or key participant has joined. This is achieved with the 'require-participant' attribute. A value of "none" indicates that the focus sends the requests immediately after the specified time has lapsed.

The absence of this conference time information indicates that a conference starts immediately and terminates when the conference policy is removed. See Section 4.2 for more details.

A running conference instance can be extended or stopped by modifying the conference time information. Note that those conference times do not guarantee resources for the conference to occur.

If a conference is in progress when deleted or stopped, the focus issues signalling requests to terminate all conference related sessions it has with participants. In SIP, the focus issues BYE requests.

4.3.4 Conference Authorization Rules

One of the key components of conference policy is the set of authorization rules that specify who is allowed to join a conference, see floors and request/grant them, subscribe to conference-information notifications and so on. The unordered list of authorization rules together define the conference authorization

policy

The conference authorization rules are enclosed in the <authorization-rules> element and are formatted according to the XML schema defined in the common policy framework [1]. In <authorization-rules> element, there can be multiple rules, each rule is represented by the <rule> element, each of which consist of three parts: conditions, actions and transformations. Conditions determine whether a particular rule applies to a request. Each action or transformation in the applied rule is a positive grant of permission to the conference participant. The details of each specific element and attribute is described in [1].

Asking the focus to allow certain users to join the conference is achieved by modifying an existing authorization rule or creating a new one. The CPS then informs the focus of such change.

If the conference is long-lasting, it is possible that new rules are added all the time but old rules are almost never removed (some of them are overwritten, though). This leads easily to the situation that the conference policy contains many unnecessary rules which are not really needed anymore. Therefore, there is a need to delete rules. This can be achieved by removing that portion of the policy.

Conflicting rules may exist (for example, both allowed and blocked action is defined for same target). The common policy directives [1] dictate the behaviour in such situations.

This section outlines the new conditions, actions and transformations for conference authorization policy.

4.3.4.1 Conditions

4.3.4.1.1 Identity

4.3.4.1.1.1 Interpreting the <id> Element

The <identity> element is already defined in the common policy framework [1]. However, the rules for interpreting the identities in <id> elements are left for each application to define separately.

This document, however, does not define the rules for interpreting identities in <id> elements in conferencing applications since those interpretation rules are signalling protocol specific.

OPEN ISSUE: Do we need to state more than this? How are identities derived from users that join using POTS, H.323, etc.?

4.3.4.1.1.2 Matching Any Identity

The <any> element is used to match any participant. This allows a conference to be open to anyone.

4.3.4.1.1.3 Matching Unauthenticated Identities

The <unauthenticated> element is used to match unauthenticated participants. That is, participants that have provided no authenticated identity to the conference focus.

4.3.4.1.1.4 Matching AnonymousIdentities

The <anonymous> element is used to match participants that have provided an authenticated identity to the conference focus, but have requested anonymity in the conference itself.

4.3.4.1.1.5 Matching Referred Identities

The <has-been-referred> element can be used to match those participants that the focus has referred to the conference.

4.3.4.1.1.6 Matching Invited Identities

The <has-been-invited> element can be used to match those participants that the focus has invited into the conference.

4.3.4.1.1.7 Matching Identities of Former Conference Participants

The <has-been-in-conference> element can be used to match those participants that have joined the conference in the past.

4.3.4.1.1.8 Matching Identities Currently in the Conference

The <is-in-conference> element can be used to match those participants that are currently participating in the conference.

4.3.4.1.1.9 Matching Key Participant Identities

The <key-participant> element can be used to match those participants that are key participants of a conference.

4.3.4.1.1.10 Matching Identities on the Dial-out List

The <is-on-dialout-list> element can be used to match those participants that are on the dial-out list.

4.3.4.1.1.11 Matching Identities on the Refer List

The <is-on-refer-list> element can be used to match those participants that are on the refer list.

4.3.4.1.1.12 Floor ID

The <floor-id> element can be used to assign users as floor moderators. It MUST be used in conjunction with the <id> element that identifies the floor moderator. The <floor-id> element carries the floor ID of the floor that the user is a moderator of. The transformation <is-floor-moderator> is used to assert that the user identified using the <id> condition is the floor moderator of the floor identified in the <floor-id> condition.

4.3.4.1.1.13 Matching PIN Codes

The <pin> element can be used to match those participants that are have knowledge on a PIN code for the conference. For example:

```
<rule id="1">
        <conditions>
                <pin>12345</pin>
        </conditions>
        <actions>
                <join-handling>allow</join-handling>
        </actions>
        <transformations/>
</rule>
```

So the condition is the PIN. If any user knows the PIN, ignoring their identity, the user is allowed to join.

A combination of the <identity> condition and the <pin> condition creates the possibility of assigning users personal PIN codes to enable them to join a conference. For example:

```
<rule id="2">
        <conditions>
                <identity>
                        <id>358401234567</id>
                </identity>
                <pin>67890</pin>
        </conditions>
        <actions>
                <join-handling>allow</join-handling>
        </actions>
        <transformations/>
</rule>
```

4.3.4.1.1.14 Matching Passwords

The <password> element can be used to match those participants that are have knowledge on a password for the conference. For example:

```
<rule id="3">
        <conditions>
                <password>pass1</password>
        </conditions>
        <actions>
                <join-handling>allow</join-handling>
        </actions>
        <transformations/>
</rule>
```

So the condition is the password. If any user knows the password, ignoring their identity, the user is allowed to join.

A combination of the <identity> condition and the <password> condition creates the possibility of assigning users personal passwords to enable them to join a conference. For example:

```
<rule id="4">
        <conditions>
                <identity>
                        <id>alice@example.com</id>
                </identity>
                <password>pass2</password>
        </conditions>
        <actions>
                <join-handling>allow</join-handling>
        </actions>
        <transformations/>
</rule>
```

4.3.4.2 Actions

4.3.4.2.1 Conference State Events

The <allow-conference-state> element represents a boolean action. If set to TRUE, the focus is instructed to allow the subscription to conference state events, such as the SIP Event Package for Conference State [14]. If set to FALSE, the subscription to conference state events would be rejected.

If this element is undefined it has a value of TRUE, causing the subscription to conference state events to be accepted.

OPEN ISSUE: Is a simple block/allow sufficient here, or should the subscription handling be similar to e.g. presence, and have three states (block, confirm, allow), or possibly even four states (block, confirm, polite-block, allow)?

4.3.4.2.2 Floor Control Events

The <allow-floor-events> element represents a boolean action. If set to TRUE, the focus is instructed to accept the subscription to floor control events. If set to FALSE, the focus is instructed to reject the subscription.

If this element is undefined, it has a value of FALSE, causing the subscription to floor control events to be rejected.

OPEN ISSUE: Is a simple block/allow sufficient here, or should the subscription handling be similar to e.g. presence, and have three states (block, confirm, allow), or possibly even four states

(block, confirm, polite-block, allow)?

4.3.4.2.3 Conference Join Handling

The "join-handling" element defines the actions used by the conference focus to control conference participation. This element defines the action that the focus is to take when processing a particular request to join a conference. This element is an enumerated integer type, with defined values of:

block: This action instructs the focus to deny access to the conference. This action has a value of zero and it is the lowest value of the "join-handling" element. This action is the default action taken in the absence of any other actions.

confirm: This action instructs the focus to place the participant on a pending list (e.g., by parking the call on a music-on-hold server), awaiting moderator input for further actions. This action has a value of one.

allow: This action instructs the focus to accept the conference join request and grant access to the conference within the instructions specified in the transformations of this rule. This action has a value of two.

Note that placing a value of block for this element doesn't guarantee that a participant is blocked from joining the conference. Any other rule that might evaluate to true for this participant that carried an action whose value was higher than block would automatically grant confirm/allow permission to that participant.

4.3.4.2.4 Dynamically Referring Users

The <allow-refer-users-dynamically> element represents a boolean action. If set to TRUE, the identity is allowed to instruct the focus to refer a user to the conference without modifying the refer-list (in SIP terms, the identity is allowed to send a REFER request to the focus which results in the focus sending a REFER request to the user the referrer wishes to join the conference). If

set to FALSE, the refer request is rejected.

If this element is undefined it has a value of FALSE, causing the refer to be rejected.

4.3.4.2.5 Dynamically Inviting Users

The <allow-invite-users-dynamically> element represents a boolean action. If set to TRUE, the identity is allowed to instruct the focus to invite a user to the conference without modifying the dial-out list (in SIP terms, the identity is allowed to send a REFER request to the focus which results in the focus sending an INVITE requested to the user the referrer wishes to join the conference). If set to FALSE, the refer request is rejected.

If this element is undefined it has a value of FALSE, causing the refer to be rejected.

4.3.4.2.6 Modifying Conference setting

The <allow-modify-settings> element represents a boolean action. If set to TRUE, the identity is allowed to modify the conference settings in the conference policy. If set to FALSE, any modifications to the conference settings are rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.7 Modifying Conference Information

The <allow-modify-information> element represents a boolean action. If set to TRUE, the identity is allowed to modify the conference information in the conference policy. If set to FALSE, any modifications to the conference information are rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.8 Modifying Conference Time

The <allow-modify-time> element represents a boolean action. If set to TRUE, the identity is allowed to modify the conference time in the conference policy. If set to FALSE, any modifications to the conference time are rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.9 Modifying Authorization rules

The <allow-modify-authorization-rules> element represents a boolean action. If set to TRUE, the identity is allowed to modify the authorization rules of a conference in the conference policy. If set

to FALSE, any modifications to the rules are rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.10 Modifying Conference Dial-out List

The <allow-modify-dol> element represents a boolean action. If set to TRUE, the identity is allowed to modify the conference dial-out list in the conference policy. If set to FALSE, any modifications to the dial-out list are rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.11 Modifying Conference Refer List

The <allow-modify-rl> element represents a boolean action. If set to TRUE, the identity is allowed to modify the conference refer list in the conference policy. If set to FALSE, any modifications to the refer list are rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.12 Modifying Conference Security Control

The <allow-modify-sc> element represents a boolean action. If set to TRUE, the identity is allowed to modify the conference security control settings in the conference policy. If set to FALSE, any modifications to the security control settings are rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.13 Modifying Conference Floor Policy

The <allow-modify-fp> element represents a boolean action. If set to TRUE, the identity is allowed to modify the conference floor policy in the conference policy. If set to FALSE, any modifications to the floor policy are rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.14 Modifying Conference media streams

The <allow-modify-ms> element represents a boolean action. If set to

TRUE, the identity is allowed to modify the conference media streams in the conference policy. If set to FALSE, any modifications to the media streams are rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.15 Creating Sidebars

The <allow-sidebar> element represents a boolean action. If set to TRUE, the identity is allowed to create and manipulate a sidebar by creating and modifying a <sidebar> element in a conference policy. If set to FALSE, any sidebar creation and manipulation is rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.16 Modifying Conference Dial-in List

The conference dial-in list is virtual and is not represented by a physical list in the conference policy. It is rather a collection of authorization rules that allow users to join a conference. The <allow-modify-dil> element represents a boolean action. If set to TRUE, the identity is allowed to create an authorization rule in the conference policy that give a user a join handling of "allow" (See Section 4.3.4.2.3. If set to FALSE, any modifications to authorization rules are rejected.

If this element is undefined it has a value of FALSE, causing the modifications to be rejected.

4.3.4.2.17 Authenticating a User

The <authenticate> element defines the mechanism used by the conference focus to authenticate a user. This element is an enumerated integer type, with defined values of:

none: This action instructs the focus not to authenticate the user.

This action has a value of zero and it is the lowest value of the <authenticate-user> element. This action is the default action taken in the absence of any other actions.

asserted-id: This action instructs the focus to authenticate the user by asserting their identity using means outside the scope of this document (for example, using digest-AKA). This action has a value of one.

shared-secret: This action instructs the focus to authenticate the user using a shared secret (for example, using digest). This action has a value of two.

certificate: This action instructs the focus to authenticate the user using a certificate (for example, using PGP). This action has a value of three.

4.3.4.3 Transformations

4.3.4.3.1 Key Participant

When the <is-key-participant> element is set to TRUE, the joining participant is denoted as a key participant. If set to FALSE, the participant is not denoted as a key participant.

If this element is undefined, it has a value of FALSE, causing no key participant status to be given to the participant.

4.3.4.3.2 Floor Moderator

When the <is-floor-moderator> element is set to TRUE, the joining conference participant is denoted as floor moderator, meaning that they are privileged to control the floor in the conference. If set to FALSE, floor moderator privileges are not given to the conference participant.

If this element is undefined, it has a value of FALSE, causing no floor moderator privileges to being granted.

4.3.4.3.3 Conference Information

The <show-conference-info> element is of type boolean transformation. If set to TRUE, conference information is shown to the conference participant. If set to FALSE, conference information is not shown to the participant.

The <show-conference-info> element controls whether information in the <settings>, <time> and <info> elements may be made available publicly. For example, an application at a conference server might list the ongoing conferences on web page, or it may allow searching for conferences based on the keywords listed in the <Conference-info> element. Not setting this transformation to any users instructs the application not to reveal any such information to any user. However, information in other elements, such as <dialout-list>, should not be seen by anyone else other than a privileged user, even with this transformation enabled for a user.

If this element is undefined, it has a value of FALSE, causing no conference information to being shown.

OPEN ISSUE: Do we require more granularity for this element? Perhaps an enumerated integer type, with defined levels of information about the conference, or a set of boolean transformations, each granting a single piece of conference information, like the ability to see "sidebar" elements?

4.3.4.3.4 Floor Holder

The <show-floor-holder> element is of type boolean transformation. If set to TRUE, the conference participant is able to see who is currently holding the floor. If set to FALSE, the participant is not able to see the floor holder.

If this element is undefined, it has a value of FALSE, causing the floor holder not be shown to the participant.

4.3.4.3.5 Floor Requests

The <show-floor-requests> element is of type boolean transformation. If set to TRUE, the conference participant is able to see the floor requests. If set to FALSE, the conference participant is not able to see floor requests.

If this element is undefined, it has a value of FALSE, causing the floor requests to not being seen by the conference participant.

4.3.5 Conference Dial-Out List

The dial-out list (DL) is a list of user URIs that the focus uses to learn who to invite to join a conference. This list can be created at conference policy creation time or updated during the conference lifetime so it can be used for mid-conference invites (and mass-invites) as well.

Asking the focus to invite (add) a user into the conference is achieved by adding that user's URI to the Dial-Out List (DL). The CPS then triggers the focus to send the conference invitation, eg: SIP INVITE as needed. Similarly, a user can be removed from the

Dial-out list by removing the URI from the dial-out list.

The <dialout-list> element is optional and includes zero or more <target> elements. The <target> element includes the mandatory 'uri' attribute. The <target> element can be extended.

4.3.6 Conference Refer List

The refer list (RL) contains a list of resources that the focus needs to refer to the conference. In SIP, this is achieved by the focus sending a REFER request to those potential participants. This list can be updated during the conference lifetime so it can be used for mid-conference refers as well.

The <refer-list> element is optional and identical to the <dialout-list> element in Section 4.3.5.

4.3.7 Conference Security Control

The conference security currently encompasses one aspects: the integrity and confidentiality of the signalling messages.

The conference security settings start with the optional <security-control> element.

We define two mechanisms for securing the signaling between users and the focus: TLS and S/MIME. TLS is used to provide transport layer security on a hop-by-hop basis. According to SIP [5], using SIPS URI scheme in a request signifies that TLS must be used to secure each hop over which the request is forwarded until the request reaches the SIP entity responsible for the domain portion of the Request-URI.

The <security-mechanism> element inside the <security-control> element has 2 boolean attributes: 'tls' and 's-mime'. When the 'tls' attribute is set to "true" (thus implying the use of SIPS URI scheme, if SIP is used as the signaling protocol), it is required that TLS is used end-to-end. In other words, TLS must be used also on the last hop between the entity responsible for the domain portion of the Request-URI and the conference policy server.

If end-to-end confidentiality of entire signalling protocol messages is not required by the conference policy, but it is required that the message bodies within the signalling protocol messages are encrypted, the 's-mime' attribute must have a value "true".

TLS and S/MIME may be required independent of each other. In other

words, it may be required to use neither, one, or both depending on the settings of these attributes.

4.3.8 Conference Floor Policy

The absence of the <floor-policy> element from an XML document indicates that the conference does not have a floor.

One or more <floor> elements can appear in the <floor-policy> element. The number of those elements indicates how many floors the conference can have. The <floor> element contains the required 'floor-control' attribute that uniquely identifies a floor and indicates the floor control protocol URI. It also contains the required boolean attribute 'moderator-controlled' that indicates if the floor is moderator controlled or not.

A floor can be used for one or more media streams; the mandatory <media-streams> element can contain zero or more of the <video>, <audio>, <application>, <data> ,<control>, <message>, and <text> elements indicating the media of the floor. Other media types can be defined by extensions. Each media stream is identified with the 'media-id' attribute. This attribute is mandatory and MUST be unique for all media streams in a conference. It is used to correlate the conference media stream in Section 4.3.9 with the ones for a floor. It is also used to correlate the media streams used in the signalling protocols with those in the conference policy, used, for example, in SDP "i" field [19].

A floor can be controlled using many algorithms; the mandatory <algorithm> element MUST contain one and only of the <moderator-controlled>, <fcfs>, and <random> elements indicating the algorithm.

The <max-floor-users> element in the <floor> element is optional and, if present, dictates the maximum number of users who can have the floor at one time. The optional <moderator-uri> indicates the URI of the moderator. It MUST be set if the attribute moderator-controlled is set to "true".

4.3.9 Conference Media Streams

Media policy is an integral part of the conference policy. It defines e.g. what kind of media topologies exist in the conference. Media policy is documented in [18].This document does not define media policy, but instead enables the user to specify the media streams a conference has. This is used by the focus to know what media streams to invite users with and what media streams it should accept from dialling in users. The details of media manipulation are defined elsewhere. User with sufficient privileges is allowed to create, modify and delete the media policy (e.g. add new media types).

The definition starts with the optional <media-streams> element. This element lists the media streams allowed for this conference. The format of this mirrors that of the <media-streams> element in floor policy in Section 4.3.8. The absence of this element indicates

that the conference will use media according to the focus local policy.

4.4 XML Schema Extensibility

The schema as be extended at multiple places:

- The <conference> element to enable more conference policy information to be added
- The <settings> element to allow for future conference settings to be defined
- The <info> element to allow further conference and host information to be conveyed
- The <occurrence> element to allow further conference timing information
- The <target> element in <dialout-list> and <refer-list> to allow extensions on the behaviour of the focus. For example, how many times to retry inviting a user
- The <security-control> element to allow new security setting for a conference to be introduced into
- The <algorithm> element in <floor-policy> to allow new algorithms to be introduced into how a floor is granted
- The <floor> element in <floor-policy> to allow extensions to floor policy for a floor
- The <media-streams> element to allow introduction of new media streams
- The <sidebar> element to allow introduction of new sidebar information

4.5 XML Schema

```
   <?xml version="1.0" encoding="UTF-8"?>
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:conference-policy"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="urn:ietf:params:xml:ns:conference-policy"
elementFormDefault="qualified">
        <!-- This import brings in the XML language attribute xml:lang-->
        <xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
        <!-- The root Conference Element -->
        <xs:element name="conference">
```

```
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="settings"
type="ConferenceSettings"/>
                                <xs:element name="info" type="ConferenceInfo"
minOccurs="0"/>
                                <xs:element name="time" type="ConferenceTime"
minOccurs="0"/>
                                <xs:element name="authorization-rules"
type="ConferenceAuthorizationRules" minOccurs="0"/>
                                <xs:element name="dailout-list" type="Target"
minOccurs="0"/>
                                <xs:element name="refer-list" type="Target"
minOccurs="0"/>
                                <xs:element name="security-control"
type="ConferenceSC" minOccurs="0"/>
                                <xs:element name="floor-policy"
type="ConferenceFloorPolicy" minOccurs="0"/>
                                <xs:element name="media-streams"
type="ConferenceMediaStreams" minOccurs="0"/>
                                <xs:any namespace="##other"
processContents="lax" minOccurs="0"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <!-- Conference Settings -->
        <xs:complexType name="ConferenceSettings">
                <xs:sequence>
                        <xs:element name="conference-uri" type="xs:anyURI"
minOccurs="1" maxOccurs="unbounded"/>
                        <xs:element name="max-participant-count"
type="xs:nonNegativeInteger" minOccurs="0"/>
                        <xs:element name="allow-sidebars" type="xs:boolean"
default="true" minOccurs="0"/>
                        <xs:element name="sidebar" type="Sidebar" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:any namespace="##other" processContents="lax"
minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <!-- Conference Info -->
        <xs:complexType name="ConferenceInfo">
                <xs:sequence>
                        <xs:element name="subject" type="xs:string"
minOccurs="0"/>
                        <xs:element name="display-name" type="xs:string"
minOccurs="0"/>
```

```
                        <xs:element name="free-text" type="xs:string"
minOccurs="0"/>
                        <xs:element name="keywords" minOccurs="0">
                                <xs:simpleType>
                                        <xs:list itemType="xs:string"/>
                                </xs:simpleType>
                        </xs:element>
                        <xs:element name="web-page" type="xs:anyURI"
minOccurs="0"/>
                        <xs:element name="host-info" minOccurs="0">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="uri"
type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
                                                <xs:element name="e-mail"
type="xs:anyURI" minOccurs="0"/>
                                                <xs:element name="web-page"
type="xs:anyURI" minOccurs="0"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                        <xs:any namespace="##other" processContents="lax"
minOccurs="0"/>
                </xs:sequence>
```

```
                <xs:attribute ref="xml:lang"/>
        </xs:complexType>
        <!-- Conference time -->
        <xs:complexType name="ConferenceTime">
                <xs:sequence>
                        <xs:element name="occurrence" minOccurs="0"
maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="mixing-start-
time" type="StartStopTime" minOccurs="0"/>
                                                <xs:element name="mixing-stop-
time" type="StartStopTime" minOccurs="0"/>
                                                <xs:element name="can-join-
after" type="xs:dateTime" minOccurs="0"/>
                                                <xs:element name="must-join-
before" type="xs:dateTime" minOccurs="0"/>
                                                <xs:element name="request-
users" type="StartStopTime" minOccurs="0"/>
                                                <xs:any namespace="##other"
processContents="lax" minOccurs="0"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <!-- Conferenece Authorisation -->
        <xs:complexType name="ConferenceAuthorizationRules">
                <xs:sequence>
                        <xs:element name="rule" type="ruleType" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ruleType">
                <xs:sequence>
                        <xs:element name="conditions" minOccurs="0">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element ref="condition"
minOccurs="0" maxOccurs="unbounded"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="actions" minOccurs="0">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element ref="action"
```

```
minOccurs="0" maxOccurs="unbounded"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="transformations" minOccurs="0">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element
ref="transformation" minOccurs="0" maxOccurs="unbounded"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
```

```
                </xs:sequence>
                <xs:attribute name="id" type="xs:string" use="required"/>
        </xs:complexType>
        <xs:element name="condition" abstract="true"/>
        <xs:element name="action" abstract="true"/>
        <xs:element name="transformation" abstract="true"/>
        <xs:element name="identity" substitutionGroup="condition">
                <xs:complexType>
                        <xs:choice>
                                <xs:element name="id" type="xs:anyURI"/>
                                <xs:sequence>
                                        <xs:element name="domain"
type="xs:string"/>
                                        <xs:sequence minOccurs="0">
                                                <xs:element name="except"
type="xs:anyURI" maxOccurs="unbounded"/>
                                        </xs:sequence>
                                </xs:sequence>
                                <xs:sequence>
                                        <xs:element name="any"
type="xs:string"/>
                                        <xs:sequence minOccurs="0">
                                                <xs:element name="except"
type="xs:anyURI" maxOccurs="unbounded"/>
                                        </xs:sequence>
                                </xs:sequence>
                        </xs:choice>
                </xs:complexType>
        </xs:element>
        <xs:element name="unauthenticated" type="xs:string"
substitutionGroup="condition"/>
        <xs:element name="anonymous" type="xs:string"
substitutionGroup="condition"/>
        <xs:element name="has-been-referred" type="xs:string"
substitutionGroup="condition"/>
        <xs:element name="has-been-invited" type="xs:string"
substitutionGroup="condition"/>
        <xs:element name="has-been-in-conference" type="xs:string"
substitutionGroup="condition"/>
        <xs:element name="is-in-conference" type="xs:string"
substitutionGroup="condition"/>
        <xs:element name="key-participant" type="xs:string"
substitutionGroup="condition"/>
        <xs:element name="is-on-dialout-list" type="xs:string"
substitutionGroup="condition"/>
        <xs:element name="is-on-refer-list" type="xs:string"
substitutionGroup="condition"/>
```

```
        <xs:element name="floor-id" type="xs:anyURI"
substitutionGroup="condition"/>
        <xs:element name="pin" type="xs:anyURI" substitutionGroup="condition"/>
        <xs:element name="password" type="xs:anyURI"
substitutionGroup="condition"/>


        <xs:element name="allow-conference-state" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-floor-events" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="join-handling" substitutionGroup="action">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:enumeration value="block"/>
                                <xs:enumeration value="allow"/>
                                <xs:enumeration value="confirm"/>
                        </xs:restriction>
                </xs:simpleType>
```

```
        </xs:element>
        <xs:element name="allow-refer-users-dynamically" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-invite-users-dynamically" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-modify-settings" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-modify-information" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-modify-time" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-modify-authorization-rules" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-modify-dol" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-modify-rl" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-modify-sc" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-modify-fp" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-modify-ms" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-sidebar" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="allow-modify-dil" type="xs:boolean"
substitutionGroup="action"/>
        <xs:element name="authenticate" substitutionGroup="action">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:enumeration value="none"/>
                                <xs:enumeration value="digest-aka"/>
                                <xs:enumeration value="digest"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>


        <xs:element name="is-key-participant" type="xs:boolean"
substitutionGroup="transformation"/>
        <xs:element name="is-floor-moderator" type="xs:boolean"
substitutionGroup="transformation"/>
        <xs:element name="show-conference-info" type="xs:boolean"
substitutionGroup="transformation"/>
        <xs:element name="show-floor-holder" type="xs:boolean"
substitutionGroup="transformation"/>
        <xs:element name="show-floor-requests" type="xs:boolean"
```

```
substitutionGroup="transformation"/>
        <!-- Target -->
        <xs:complexType name="Target">
                <xs:sequence>
                        <xs:element name="target" minOccurs="0"
maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:any namespace="##other"
processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                                        </xs:sequence>
                                        <xs:attribute name="uri"
type="xs:anyURI" use="required"/>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <!-- Security Control (SC) -->
        <xs:complexType name="ConferenceSC">
                <xs:sequence>
                        <xs:element name="security-mechanism">
                                <xs:complexType>
                                        <xs:attribute name="tls"
type="xs:boolean" default="false"/>
```

```
                                        <xs:attribute name="s-mime"
type="xs:boolean" default="false"/>
                                </xs:complexType>
                        </xs:element>
                        <xs:any namespace="##other" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <!-- Conference Floor Control Policy -->
        <xs:complexType name="ConferenceFloorPolicy">
                <xs:sequence>
                        <xs:element name="floor" maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="media-
streams" type="ConferenceMediaStreams"/>
                                                <xs:element name="algorithm">
                                                        <xs:complexType>
                                                                <xs:sequence>

<xs:element name="moderator-controlled" minOccurs="0"/>

<xs:element name="fcfs" minOccurs="0"/>

<xs:element name="random" minOccurs="0"/>
                                                                                <xs:any
namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                                                                </xs:sequence>
                                                        </xs:complexType>
                                                </xs:element>
                                                <xs:element name="max-floor-
users" type="xs:nonNegativeInteger" minOccurs="0" default="1"/>
                                                <xs:element name="moderator-
URI" type="xs:anyURI" minOccurs="0"/>
                                                <xs:any namespace="##other"
processContents="lax" minOccurs="0"/>
                                        </xs:sequence>
                                        <xs:attribute name="floor-control"
type="xs:anyURI"/>
                                        <xs:attribute name="moderator-
controlled" type="xs:boolean" default="false"/>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <!-- Conference Media Streams -->
        <xs:complexType name="ConferenceMediaStreams">
```

```
                <xs:sequence>
                        <xs:element name="video" type="Media" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="audio" type="Media" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="application" type="Media"
minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="data" type="Media" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="control" type="Media" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="message" type="Media" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="text" type="Media" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:any namespace="##other" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <!-- Start/Stop time -->
        <xs:complexType name="StartStopTime">
```

```
                <xs:simpleContent>
                        <xs:extension base="xs:dateTime">
                                <xs:attribute name="required-participant"
use="required">
                                        <xs:simpleType>
                                                <xs:restriction
base="xs:string">
                                                        <xs:enumeration
value="key-participant"/>
                                                        <xs:enumeration
value="participant"/>
                                                        <xs:enumeration
value="none"/>
                                                </xs:restriction>
                                        </xs:simpleType>
                                </xs:attribute>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <!-- Sidebar -->
        <xs:complexType name="Sidebar">
                <xs:sequence>
                        <xs:element name="uri" type="xs:anyURI" minOccurs="1"
maxOccurs="unbounded"/>
                        <xs:element name="policy" type="xs:anyURI"
minOccurs="0"/>
                        <xs:any namespace="##other" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="id" type="xs:string" use="required"/>
        </xs:complexType>
        <!-- Media -->
        <xs:complexType name="Media">
                <xs:attribute name="media-id" type="xs:string" use="required"/>
        </xs:complexType>
   </xs:schema>
```

5. Conference Policy Manipulation and Conference Entity Behaviour

5.1 Overview of Operation

This document assumes that the user knows the location of conference policy serve, the details of that discovery are beyond the scope of this document.

CPCP allows clients to manipulate the conference policy at conference policy server (CPS). CPS is able to inform the focus about changes in conference policy, if necessary. For example, if new users are added to the dial-out list, then conference policy server informs the focus which makes the invitations as requested.

Some assumptions about the conferencing architecture are made. Clients always connect to the conference policy server (CPS) when they perform manipulation operations. It is assumed that the CPS informs other conferencing entities, such as focus, the floor control

server and the mixer directly or via the focus. For example, if user A wants to expel user B from an ongoing conference, user A must first manipulate the conference policy data. The CPS then communicates that change to the focus to perform the operation.

5.2 Use of External Lists

External lists MAY be used in a conference policy. They can be used in the dial-out list, the refer-list and the authorization policy. An external list is a list of resources created by means outside the scope of this document.

A privileged user of the conference policy uses an external list by placing its manipulation URI in an element that carries a URI. At the time the focus needs to activate the policy surrounding the URI, the focus fetches the URIs for the members of the external list using the list URI. For example, a conference creator creates a conference and places the URI of an external list in the dial-out list. At some point, the focus needs to invite using on the dial-out list to join the conference. It is at that moment that the focus retrieves the members of the external list. It then sends INVITE (in SIP terms) to the members of that external list. This results in all participants connected to one focus.

In can happen that the external list is not accessible at the time the focus requires it. In this case, the external list is ignored, and in the case of an authorization rule, that rule fails.

There are also cases where the external list has been manipulated. It is outside the scope of this document how the focus can learn of such manipulation. But if is does, it reacts in a similar manner as it would have if the list was local and has been modified.

If an external list contains a reference to yet another list, that reference is ignored.

5.3 Communication Between Conference Entities

The communication between different (logical) conferencing elements is beyond the scope of this document. It can be expected that in

most cases CPS includes also those logical functions.

5.4 Manipulating Participant Lists

A user with sufficient privileges is allowed to perform user management operations, such as adding a new user to the conference or expelling a user from the conference. These operations are performed by modifying the conference policy at the conference policy server.

After authorising the user to do such manipulations, the conference policy server communicates the change to the focus. The focus reacts by performing singlling operations such as sending SIP INVITE, BYE or REFER.

5.4.1 Expelling a Participant

Expelling a user is performed by a privileged user creating or manipulating an existing authorization rule and setting that user's <join-handling> action to "block>. The focus reacts by terminating the session with that participant, such as a sending SIP BYE request.

Care must be taken since if one rules allows a user to join and one blocks a user from joining, the result in that the user is allowed to join. For example, Bob can join a conference since an authorization rule has been defined to allow everyone at example.com:

```
<rule id="1">
        <conditions>
                <identity>
                        <domain>example.com</domain>
                </identity>
        </conditions>
        <actions>
                <join-handling>allow</join-handling>
        </actions>
        <transformations/>
</rule>
```

Setting the following rule will not block Bob from joining nor will it expel him since the above rule overrides it:

```
<rule id="2">
        <conditions>
                <identity>
                        <uri>bob@example.com</uri>
```

```
        </identity>
    </conditions>
    <actions>
        <join-handling>block</join-handling>
    </actions>
```

```
        <transformations/>
    </rule>
```

So, in order to expel Bob, the original rule has to be modified using the <except> element:

```
    <rule id="1">
        <conditions>
            <identity>
                <domain>example.com</domain>
                <except>bob@domain.com</except>
            </identity>
        </conditions>
        <actions>
            <join-handling>allow</join-handling>
        </actions>
        <transformations/>
    </rule>
```

5.5 Re-joining a Conference

Participants can drop out of a conference for many reasons including: client crash, out of coverage, had to leave for a while. It might be of interest to enable that user to re-join the conference. To allow that, participants that have departed the conference gracefully can only re-join if a privileged user has added an authorization rule allowing them to join. Participants that have departed the conference ungracefully (eg: crash) require a special behaviour from the focus . The focus is aware when a user has not gracefully departed a conference (for example; it did not receive a SIP BYE request and media is no longer being received). If this is the case, the focus is required to re-issue the invitation or referral to that user after a pre-configured unit of time.

5.6 Floor Control Policy vs. Floor Control Protocol

Conference floor control is an optional feature provided by a

separate floor control protocol (FCP). However, creating a floor and defining a floor policy belongs to CPCP. Moreover, setting some key floor parameters, such as floor moderator in moderator controlled floor policy, belongs to CPCP. FCP only defines how to request, grant, deny and revoke a floor within given floor policy.

For example, in a typical conference the privileged conference user uses CPCP for creating a floor for audio plane, defining the floor policy as "moderator-controlled" and appointing one user - possibly himself - to act as a floor moderator governing the access to the floor.

When the floor has been created and a floor moderator has been assigned, the floor moderator gets notifications from the focus and is able to accept or deny floor requests from the conference users. Note that FCP does not create media streams (just the virtual floor attached to media), as media streams are created using CPCP. The details of FCP are beyond the scope of this draft.

6. An XCAP Usage for Conference Policy Manipulation

6.1 Application Unique ID

XCAP requires application usages to define a unique application usage ID (AUID) in either the IETF tree or a vendor tree. This specification defines the "conference-policy" AUID within the IETF tree, via the IANA registration in Section 9.

6.2 Resource Interdependencies

The conference policy server MAY fill the conference URI(s), but the client MUST propose a conference URI. If the CPS does not allow assignments of URIs by the client, it rejects the request with a "409" response and SHOULD include a body in the response detailing the error. XCAP Base document [10] section 7.2.1 explains how such a response body is constructed. The CPS MAY assign multiple conference URIs to a conference, one for each call signaling protocol that it supports. Section 4.3.1 discusses this is more detail.

Sidebar URIs are subject to the same behaviour.

6.3 Additional Constraints

These are defined within the XML structure definition.

6.4 Naming Conventions

There are no naming conventions that need to be defined for this application usage.

6.5 Authorization Policies

A server can allow privileged users to modify documents that they don't own. The establishment and indication of such policies is done

by setting the authorization rules as described in Section 4.3.4.

6.6 MIME Type for CPCP XML Document

The MIME type for the CPCP XML document is defined in Section 4.1

7. Examples

The following is an example of a document compliant to the schema:

Below is an example how to create a conference:

7.1 An Example CPCP Document

Alice creates a conference with the follows policy:

- Conference URIs are suggested to be sip:myconference@example.com and tel:+3581234567.
- Maximum number of participants in the conference is 10.
- The conference allows side-bars
- Media mixing starts at the latter of 9:30 am and the first participant arrives
- Media mixing sends at 12:30 pm. The conference does not need a key participant to continue.
- Users can join 5 minutes before media mixing starts and cannot join half an hour before media mixing ends.
- Users are requested to join a conference (invited and referred) 5 minutes before the conference starts and no participant nor key-participant is needed for this action to take place.

- Everyone at the domain example.com is allowed to join and can subscribe to the conference state event package.

- Alice is a key participant

- Alice will be invited to join the conference while Sarah will be referred to the conference.

- No TLS, will be used but S/MIME is required.

- PIN code is set to 13579 and password is set to abcd1234.

- One floor is created for audio and a first-come-first-serve policy.

- Two media are made available in the conference:audio and video.

The resulting CPCP document looks like

```
<?xml version="1.0" encoding="UTF-8"?>
<conference xmlns="urn:ietf:params:xml:ns:conference-policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <settings>
                <conference-uri>sip:myconference@example.com</conference-uri>
                <max-participant-count>10</max-participant-count>
                <allow-sidebars>true</allow-sidebars>
        </settings>
        <info xml:lang="en-us">
                <subject>What's happening tonight</subject>
                <display-name>Party Goer's</display-name>
                <free-text>John and Peter will join the conference soon</free-
text>
                <keywords>party nightclub beer</keywords>
                <host-info>
                        <uri>sip:Alice@example.com</uri>
                        <uri>tel:+3581234567</uri>
                        <e-mail>mailto:Alice@example.com</e-mail>
                        <web-page>http://www.example.com/users/Alice</web-page>
                </host-info>
        </info>
        <time>
                <occurrence>
                        <mixing-start-time required-
participant="participant">2004-12-17T09:30:00-05:00</mixing-start-time>
                        <mixing-stop-time required-
participant="none">2004-12-17T12:30:00-05:00</mixing-stop-time>
                        <can-join-after>2001-12-17T09:25:00-05:00</can-join-
after>
                        <must-join-before>2004-12-17T12:00:00-05:00</must-join-
before>
                        <request-users required-
participant="none">2001-12-17T09:30:00-05:00</request-users>
                </occurrence>
```

```
        </time>
        <authorization-rules>
                <rule id="1">
                        <conditions>
                                <identity>
                                        <domain>example.com</domain>
                                </identity>
                        </conditions>
                        <actions>
                                <allow-conference-state>true</allow-conference-
state>
```

```
                        <join-handling>allow</join-handling>
                    </actions>
                    <transformations/>
                </rule>
                <rule id="2">
                    <conditions>
                        <identity>
                            <id>alice@example.com</id>
                        </identity>
                    </conditions>
                    <actions>
                        <allow-sidebar>true</allow-sidebar>
                    </actions>
                    <transformations>
                       <is-key-participant>true</is-key-participant>
                    </transformations>
                </rule>
        </authorization-rules>
        <dailout-list>
                <target uri="sip:bob@example.com"/>
        </dailout-list>
        <refer-list >
                <target uri="sip:sarah@example.com"/>
        </refer-list >
        <security-control>
                <security-mechanism tls="false" s-mime="true"/>
                <pin>13579</pin>
                <password>abcd1234</password>
        </security-control>
        <floor-policy>
                <floor floor-control="fcp://example.com/floorabc" moderator-
controlled="false">
                    <media-streams>
                        <audio media-id="2"/>
                    </media-streams>
                    <algorithm>
                        <fcfs/>
                    </algorithm>
                    <max-floor-users>1</max-floor-users>
                </floor>
        </floor-policy>
        <media-streams>
                <video media-id="1"/>
                <audio media-id="2"/>
        </media-streams>
   </conference>
```

7.2 CPCP Manipulations Using XCAP

1. Creating a Conference

Continuing with the example from Section 7.1, Alice's client uses XCAP to transport the conference policy to the conference policy server

PUT http://xcap.example.com/services/conferences/users/Alice/conference.xml HTTP/1.1 Content-Type:application/conference-policy+xml

Content-Type: application/conference-policy+xml

[conference policy from Section 7.1 goes here.

At exactly 2004-12-17T09:30:00-05:00, the focus sends SIP INVITE request to Alice and a SIP REFER request to Sarah. At 2004-12-17T09:25:00-05:00, SIP INVITE requests can be accepted from anyone at domain example.com. Any attempts to join the conference by users in other domains are rejected.

2. Expelling a User

After the conference has started, Alice decides to expel Bob who has joined the conference. So she modifies the authorization rule that allows everyone at example.com to join:

PUT http://xcap.example.com/services/conferences/users/Alice/conference.xml/~~/conference/authorization-rules/rule[@id=""]/conditions/identity/ HTTP/1.1

Content-Type:text/plain

```
<identity>
     <domain>example.com</domain>
     <except>bob@example.com</except>
</identity>
```

At this point, the focus sends a SIP BYE request to Bob ending Bob's participation in the conference. This also guarantees that Bob cannot rejoin the conference since he is explicitly blocked. Any attempt Bob makes in rejoining the conference will fail.

3. Allowing An Expelled Participant To Join Again

Continuing with the example above, Alice now decides to allow Bob to join again after a period of time. She does so by rewriting parts of the rule that blocks him from joining.

```
   PUT
   http://xcap.example.com/services/conferences/users/Alice/conference.xml/
~~/conference/authorization-rules/rule[@id=""]/conditions/identity/ HTTP/1.1

   Content-Type:text/plain


<identity>
     <domain>example.com</domain>
</identity>
```

Bob can now rejoin the conference by sending a SIP INVITE request.

4. Allowing Sarah to Refer Users

Alice now decides that Sarah can ask the focus to refer users to the conference:

```
   PUT
   http://xcap.example.com/services/conferences/users/Alice/conference.xml/
~~/conference/authorization-rules/rule[@id="3"] HTTP/1.1

   Content-Type:text/plain


<rule id="3">
     <conditions>
            <identity>
                    <uri>sarah@example.com</uri>
            </identity>
     </conditions>
```

```
        <actions>
                <allow-refer-users-dynamically>true</allow-refer-users-
dynamically>
        </actions>
        <transformations/>
   </rule>
```

5. Removing A Conference

Alice now decides she no longer wants this conference to exist and therefore deletes the conference:

```
DELETE
http://xcap.example.com/services/conferences/users/Alice/conference.xml
```

As a result of this action, the focus sends SIP BYE requests to all current participants in the conference. The conference server terminates the focus thereafter.

8. Security Considerations

A conference document may contain information that is highly sensitive. Its delivery to the conference server needs to happen strictly, paying special attention to integrity and confidentiality. Reading the document is also a security concern since the conference policy contains sensitive information like the PIN code, password of the conference, the topic of the conference, who is allowed to join and the URIs of the users that can participate.

Manipulations of the conference policy have similar security issues. Users with relevant privileges can manipulate parts of the conference policy giving themselves and others privileges to manipulate the conference policy, including the dial-out list and the security control settings for a conference. This can happen because the conference policy it self carries the identities and the authorization rules that apply to those identities. Those authorization rules carry the privileges that certain identities have. If an unauthorized user gets access to this document (pretending to be someone else), s/he can manipulate those rules giving himself and other unauthorized users access to the conference policy. S/he can also manipulate other parts of the conference policy under a false identity. Some of the things that a malicious user can do include: denying users certain privileges, giving himself floor moderation, removing users from lists, removing rules for certain identities, giving privileges to other malicious users, changing the media streams and changing conference time. Therefore,

it is very important that only authorized clients are able to manipulate the conference policy. Any conference policy transport protocol MUST provide authentication, confidentiality and integrity.

In the case that XCAP is used to create and manipulate a conference policy, the XCAP base specification mandates that all XCAP servers

MUST implement HTTP Authentication: Basic and Digest Access Authentication [16]. Furthermore, XCAP servers MUST implement HTTP over TLS [17]. It is recommended that administrators of XCAP servers use an HTTPS URI as the XCAP root services URI, so that the digest client authentication occurs over TLS. By using these means, XCAP client and server can ensure the confidentiality and integrity of the XCAP created conference policy document and its manipulation operations, and that only authorized clients are allowed to perform them.

9. IANA Considerations

9.1 XCAP Application Usage ID

This section registers a new XCAP Application Usage ID (AUID) according to the IANA procedures defined in..

Name of the AUID: conference-policy
Description: Conference policy application manipulates conference policy at a server.

9.2 application/conference-policy+xml MIME TYPE

MIME media type: application

MIME subtype name: conference-policy+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [7].

Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [7].

Security considerations: See section 10 of RFC 3023 [7] and section Section 9 of this document.

Interoperability considerations: none.

Published specification: This document.

Applications which use this media type: This document type has been used to support conference policy manipulation for SIP based conferencing.

Additional information:

Magic number: None

File extension: .cl or .xml

Macintosh file type code: "TEXT"

Personal and email address for further information: Petri Koskelainen (petri.koskelainen@nokia.com)

Intended Usage: COMMON

Author/change controller: The IETF

9.3 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:conference-policy

This section registers a new XML namespace, as per guidelines in URN document [15].

URI: The URI for this namespace is urn:ietf:params:xml:ns:conference-policy.

Registrant Contact: IETF, XCON working group, Petri Koskelainen (petri.koskelainen@nokia.com)

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
     "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <meta http-equiv="content-type"
   content="text/html;charset=iso-8859-1"/>
 <title>Conference Policy Namespace</title>
</head>
<body>
```

```
    <h1>Namespace for Conference Policy</h1>
    <h2>application/conference-policy+xml</h2>
    <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
  </body>
  </html>
  END
```

10. Contributors

Jose Costa-Requena

Simo Veikkolainen

Teemu Jalava

11. Acknowledgements

The authors would like to thank Markus Isomaki, Adam Roach, Eunsook Kim, Roni Evan and the IETF XCON working group for their feedback and suggestions.

12. References

12.1 Normative References

[1] Schulzrinne, H., Tschofenig, H., Cuellar, J., Polk, J. and J. Rosenberg, "Common Policy", Internet-Draft I-D.ietf-geopriv-common-policy, February 2004.

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

[3] Moats, R., "URN Syntax", RFC 2141, May 1997.

[4] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

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

[6] Bray, T., Paoli, J., Sperberg-McQueen, C. and E. Maler,

"Extensible Markup Language (XML) 1.0 (Second Edition)", W3C REC REC-xml-20001006, October 2000.

[7] Murata, M., Laurent, S. and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[8] Koskelainen, P. and H. Khartabil, "Requirements for conference policy control protocol", draft-ietf-xcon-cpcp-req-01 (work in progress), January 2004.

[9] Johnston, A. and O. Levin, "Session Initiation Protocol Call Control - Conferencing for User Agents",

draft-ietf-sipping-cc-conferencing-03 (work in progress), February 2004.

[10] Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-02 (work in progress), February 2004.

[11] Rosenberg, J., "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Presence Lists", draft-ietf-simple-xcap-list-usage-02 (work in progress), February 2004.

[12] Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Modification Events for the Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Managed Documents", draft-ietf-simple-xcap-package-01 (work in progress), February 2004.

[13] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", draft-ietf-sipping-conferencing-framework-01 (work in progress), October 2003.

[14] Rosenberg, J., Shulzrinne, H. and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", draft-ietf-sipping-conference-package-03, February 2004.

[15] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004.

[16] Franks, J., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[17] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

12.2 Informative References

[18] Jennings, C. and B. Rosen, "Media Mixer Control for XCON", draft-jennings-xcon-media-control-00 (work in progress), February 2004.

[19] Handly, M., Eriksson, G., Jacobson, V. and C. Perkins, "Grouping of Media Lines in SDP", draft-ietf-mmusic-sdp-new-18 (work in progress), June 2004.

Authors' Addresses

Hisham Khartabil
Nokia
P.O. Box 321
Helsinki FIN-00045
Finland

EMail: hisham.khartabil@nokia.com

Petri Koskelainen
Nokia
P.O. Box 100 (Visiokatu 1)
Tampere FIN-33721
Finland

EMail: petri.koskelainen@nokia.com

Aki Niemi
Nokia
P.O. Box 100
NOKIA GROUP, FIN 00045
Finland

Phone: +358 50 389 1644
EMail: aki.niemi@nokia.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.