XCON Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 28, 2008

M. Barnes
Nortel
C. Boulton
Avaya
H. Schulzrinne
Columbia University
February 25, 2008

Centralized Conferencing Manipulation Protocol
draft-barnes-xcon-ccmp-04

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on August 28, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2008).

Abstract

The Centralized Conferencing Manipulation Protocol (CCMP) defined in this document provides the mechanisms to create, change and delete objects related to centralized conferences, including participants, their media and their roles. The protocol relies on web services and SIP event notification as its infrastructure, but can control

conferences that use any signaling protocol to invite users. CCMP is based on the Simple Object Access Protocol (SOAP), with the data necessary for the interactions specified via Web Services Description Language (WSDL).

Table of Contents

1. Introduction

The Framework for Centralized Conferencing (XCON) [7]defines a signaling-agnostic framework, naming conventions and logical entities required for constructing advanced conferencing systems. A primary concept introduced in the XCON framework is the existence of a conference object. The framework introduces the conference object as a logical representation of a conference instance which represents the current state and capabilities of a conference.

The Centralized Conferencing Manipulation Protocol (CCMP) defined in this document allows the creation, manipulation and deletion of a conference object by authenticated and authorized clients. This includes adding and removing participants, changing their roles, as well as adding and removing media streams and associated end points.

CCMP implements a client-server model. The server is the Conference Control Server defined in the XCON framework, while clients can either be signaling end points, such as Session Initiation Protocol (SIP) RFC 3261 [10] user agents, or control-only agents that do not contribute media to the conference.

CCMP manipulates conferences based on their semantic properties and is based on a client-server Remote Procedure Call (RPC) mechanism, with the Simple Object Access Protocol (SOAP) [5] and [6] used to carry out the appropriate client-server protocol transactions.

The common information contained in conference objects is defined using an XML representation based on the schema in the XCON data model [8]. These data structures are used as the basis for the Web Services Description Language (WSDL) [4] definition and XML schema.

This document first provides some background on the motivations associated with the design of CCMP in Section 4 followed by a brief discussion of the system architecture in Section 5. The protocol operations are then detailed in Section 7, with a discussion of the key elements in the conference object in Section 6. The practical sequence of protocol operations is discussed in Section 8, with examples provided in Section 10. An XML schema is provided in Section 12. WSDL information is detailed in Section 13.

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 [1] and indicate requirement levels for

compliant implementations.

3. Terminology

This document reuses the terminology defined in the Framework for Centralized Conferencing [7]. In addition, the following acronyms and terms are used in this document:

SOAP: Simple Object Access Protocol[5][6].
WSDL: Web Services Description Language[4]. WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.
W3C: World Wide Web Consortium. The organization that developed the SOAP and WSDL specifications referenced within this document.

4. Motivation

SOAP is chosen as the RPC mechanism due to its compatibility with the requirements for the conference control protocol as introduced in the framework for centralized conferencing. SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML-based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP allows the re-use of libraries, servers and other infrastructure and provides a convenient mechanism for the formal definition of protocol syntax using Web Services Description Language (WSDL).

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP.

It is likely that implementations and future standardization work will add more conference attributes and parameters. There are three types of extensions. The first and simplest type of extension adds elements to the overall conference description, media descriptions or

descriptions of users. The XML namespace mechanism makes such extensions relatively easy, although implementations still have to deal with implementations that may not understand the new namespaces. The Options operation (Section 7.1) allows clients to determine the capabilities of a specific server, reflected by the specific blueprints supported by that server.

A second type of extension replaces the conference, user or media objects with completely new schema definitions, i.e., the namespaces for these objects themselves differ from the basic one defined in this document. As long as the Options request remains available and keeps to a mutually-understood definition, a compatible client and server will be able to bootstrap themselves into using these new objects.

Finally, it is conceivable that new object types are needed beyond the core conference, user and media objects and their children. These would also be introduced by namespaces.

5. System Architecture

CCMP supports the framework for centralized conferencing. Figure 1 depicts a subset of the 'Conferencing System Logical Decomposition' architecture from the framework for centralized conferencing document. It illustrates the role that CCMP assumes within the overall centralized architecture.

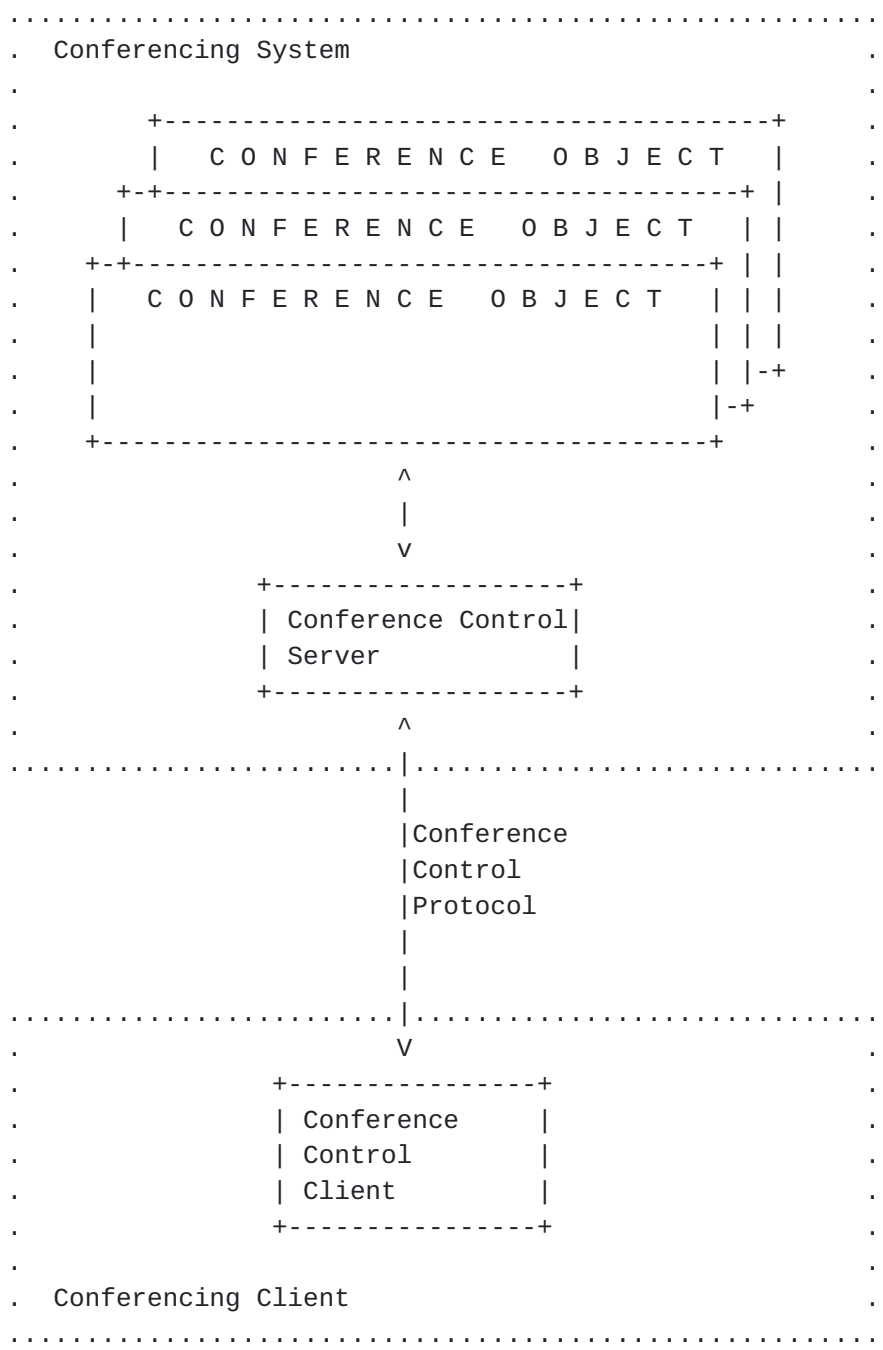

Figure 1: Conference Client Interaction

CCMP serves as the Conference Control Protocol, allowing the conference control client to interface with the conference object maintained by the conferencing system, as represented in Figure 1. Conference Control is one part of functionality for advanced conferencing supported by a conferencing client. Other functions are discussed in the framework for centralized conferencing document and

related documents.

6. Conference Object and User Identifiers

This section provides an overview of the conference object and conference users in relation to the CCMP protocol. The identifiers used in CCMP for the conference object (XCON-URI) and conference user (XCON-USERID) are introduced in the XCON framework and defined in the XCON data model [8]

6.1. Conference Object

Conference objects feature a simple dynamic inheritance-and-override mechanism. Conference objects are linked into a tree, where each tree node inherits attributes from its parent node. The roots of these inheritance trees are also known as "blueprints". Nodes in the inheritance tree can be active conferences or simply descriptions that do not currently have any resources associated with them. An object can mark certain of its properties as unalterable, so that they cannot be overridden.

The schema for the conference object is defined in the XCON data model. Conference objects are uniquely identified by the XCON-URI. A client MAY specify a parent element that indicates the parent from which the conference is to inherit values. When creating conferences, the XCON-URI included by the client is only a suggestion. To avoid identifier collisions and to conform to local server policy, the conference control server MAY choose a different identifier.

6.2. Conference Users and Participants

Each conference can have zero or more users. All conference participants are users, but some users may have only administrative functions and do not contribute or receive media. Users are added one user at a time to simplify error reporting. Users are inherited as well, so that it is easy to set up a conference that has the same set of participants or a common administrator. The Conference Control Server creates individual users, assigning them a unique Conference User Identifier (XCON-USERID).

A variety of elements defined in the common <conference-info> element as specified in the XCON data model are used to determine how a specific user expects and is allowed to join a conference as a participant, or users with specific privileges (e.g., observer). For example, the <method> attribute defines how the caller joins the conference, with a set of defined XML elements, namely <dial-in> for

users that are allowed to dial in and <dial-out> for users that the conference focus will be trying to reach. <dial-in> is the default.

If the conference is currently active, dial-out users are contacted immediately; otherwise, they are contacted at the start of the conference. The conference control server assigns a unique Conference User Identifier (XCON-USERID) to each user. The conference control server uses the XCON-USERID to change or delete <user> elements. Depending upon policies and privileges, specific users MAY also manipulate <user> elements.

In many conferences, users can dial in if they know the XCON-URI and an access code shared by all conference participants. This type of user is initially represented in the data by a <user> element without an entity attribute. Only the (default) type of <dial-in> is permitted for this type of user. The users are identified, in the entity attribute, by their call signaling URL, such as their SIP URL or tel URI [12]. In cases where there is no such URI, e.g., because a PSTN caller has blocked caller-ID delivery, the server assigns a locally-unique URI, such as a locally-scoped tel URI. The conference control server assigns a unique Conference User Identifier (XCON-USERID) to these users when they dial in to join the conference. If the user supports the notification event package [13], they can receive their XCON-USERID, thus allowing them to also manipulate the <user> attribute in the conference object.

7. Protocol Operations

The primary function of the protocol defined within this document is to provide a conference control client with the ability to carry out specific operations on a conference object. This section describes the generic behavior of the core protocol operations on conference objects. Each object has four basic operations: retrieve, create, change and delete. The XCON-URI as discussed in Section 6.1 is the target for each of these operations.

To simplify operations, a conference control server treats certain parameters as suggestions, as noted in the object description. If the conference control server cannot set the parameter to the values desired, it picks the next best value, according to local policy and returns the values selected in the response. If the client is not satisfied with these values, it simply deletes the object.

There is also a querying mechanism ("options") to ascertain the namespaces understood by the server. Any elements with namespaces not understood by the server are to be ignored by the server. This allows a client to include optional elements in requests without

having to tailor its request to the capabilities of each server.

A conference control client and conference control server MUST provide the ability to action all of the protocol operations in this section and MUST fully implement the SOAP WSDL schema defined in Section 13 which uses HTTP operations as the transport mechanism.

7.1. Options

The "options" operation is used by a client to query a system for its capabilities and doesn't pertain to a particular conference object. In this document, the response returns the XML namespaces that the server understands and the namespaces to be used in responses that it requires the client to understand. Within the conferencing system, the namespaces correlate with blueprints, as specified in the XCON framework. The blueprints are comprised of conference information initialized to specific values and ranges.

7.2. Retrieve

The "retrieve" operation is used by a client to query a system for a specific template in the form of a blueprint prior to the creation of a conference. In this case, the "retrieve" operation often follows an "options" operation, although a conferencing control client may be pre-configured to perform the "retrieve" operation on a specific blueprint.

The "retrieve" operation is also used to get the current representation of a specific conference object for a conference reservation or an active conference. In this case, the the unique conference identifier (XCON-URI) MUST be included in the CCMP request.

The "retrieve" operation returns the full XML document describing the conference object in its current state including all inherited values. Elements may be marked by attributes, in particular, whether they are specific to this instance or have been inherited from the parent node.

To simplify operations, HTTP GET can also be used directly on XCON-URIs, so that simple systems that need to only obtain data about conference objects do not need a full SOAP implementation.

7.3. Create

The "create" operation is used by a client to create and reserve a conference object. The creation of the conference object can be explicit by requesting it to be created based upon a specific

blueprint. When the creation of a conference object is implicit, with no specific blueprint specified, the creation and reservation of the conference instance is based on the default conference object. The default conference object is specific to a conference control server and its specification is outside the scope of this document.

When creating conferences, any XCON-URI included by the client is considered as a suggestion. To avoid identifier collisions and to conform to local server policy, the conference control server MAY choose a different identifier. The identifier is returned in the response.

In addition, the conference description MAY contain a calendar element, in the iCal format in XML rendition defined in CPL [11] or (preferable, if available as stable reference) xCal [14]. This description indicates when the conference is active.

The "create" operation may also be used to create a new conference user, in which case an XCON-USERID is included in the request. The response to this operation includes an XCON-USERID, which may be different than the one sent by the client in the request.

To simplify operations, HTTP PUT can also be used to create a new objects as idenfied by the XCON-URI or XCON-URI.

7.4. Change

The "change" operation updates the conference object as referenced by the XCON-URI included in the request. A request which attempts to change a non-existing object is an error, as is a request which attempts to change a parameter that is inherited from a protected element.

During the lifetime of a conference, this operation is used by a conference control client to manipulate a conference object. This includes the ability to pass relevant fragments of the conference object, to manipulate specific elements in the conference object. [Editor's note: the mechanism for manipulation of specific elements in the conference object (i.e., partial updates) requires further consideration and detail.]

Upon receipt of a "change" operation, the conference control server updates the specific elements in the referenced conference object. Object properties that are not explicitly changed, remain as-is. This approach allows a conference control client to manipulate objects created by another application even if the manipulating application does not understand all object properties.

To simplify operations, HTTP POST can also be used to change the conference objectidentified by the XCON-URI.

7.5. Delete

This conference control operation is used to delete the current representation of a conference object and requires the unique conference identifier (XCON-URI) be provided by the client.

A request which attempts to delete a conference object that is being referenced by a child object is an error.

To simplify operations, HTTP DELETE can also be used to delete conference objects identified by the XCON-URI.

8. Protocol Operations on Conference Objects

The primary function of CCMP is to provide a conference control client with the ability to carry out specific operations on a conference object. As mentioned previously, SOAP is used as the XML RPC mechanism to fulfill such operations.

A conference client must first discover the conference control server as described in Section 8.1. The conference control server is the target for the CCMP requests.

The conference control operations as described in Section 7 are enveloped in SOAP requests and responses in the form of CCMP Requests (Section 8.2 and CCMP Responses (Section 8.3 respectively.

8.1. Locating a Conference Control Server

If a conference control client is not pre-configured to use a specific conference control server for the requests, the client MUST first discover the conference control server before it can send any requests.

There are several options for discovery of the conference control server.

[Editor's note: need to add more detail in this section!]

8.2. Constructing a CCMP Request

The construction of the SOAP envelope associated with a CCMP request message complies fully with the WSDL, as defined in Section 13. Construction of a valid CCMP request is based upon the operations

defined in Section 7, depending upon the function and associated information desired by the conference control client.

8.3. Handling a CCMP Response

As with the CCMP request message, the CCMP response message is enclosed in a SOAP envelope. A response to the CCMP request MUST contain a response code and may contain other elements depending upon the type of request and the value of the response code. A summary of the response codes is provided in Section 8.3.1 followed by the handling of responses and specific response codes for each of the operations in Section 8.3.2.

8.3.1. Response codes

All response codes are application-level, and MUST only be provided in successfully processed transport-level responses. For example where HTTP is used, CCMP Response messages MUST be accompanied by a 200 OK HTTP response.

The set of CCMP Response codes currently contain the following tokens:

success: This code indicates that the request was successfully processed.
pending: This code indicates that the notification is to follow.
modified: This code indicates that the object was created, but may differ from the request.
badRequest: This code indicates that the request was badly formed in some fashion.
unauthorized: This code indicates that the user was not authorized for the specific operation on the conference object.
forbidden: This code indicates that the specific operation is not valid for the target conference object.
objectNotFound: This code indicates that the specific conference object was not found.
operationNotAllowed: This code indicates that the specific operation is not allowed for the target conference object (e.g.., due to policies, etc.)
deleteFailedParent: This code indicates that the conferencing system cannot delete the specific conference object because it is a parent for another conference object.
modifyFailedProtected: This code indicates that the target conference object cannot be changed (e.g., due to policies, roles, privileges, etc.).

requestTimeout: This code indicates that the request could not be processed within a reasonable time, with the time specific to a conferencing system implementation.
serverInternalError: This code indicates that the conferencing system experienced some sort of internal error.
notImplemented: This code indicates that the specific operation is not implemented on that conferencing system.

8.3.2. Operation Responses

The following sections detail the operation specific handling of the response codes.

8.3.2.1. Options Response

A CCMP Response for an "options" operation, containing a response code of "success", MUST include the XML namespaces that the server understands and the namespaces to be used in subsequent responses that it requires the client to understand. Future work may add more global capabilities rather than conferencing system specific. Within the conferencing system, the namespaces correlate with blueprints, as specified in the XCON framework. The blueprints are comprised of conference information initialized to specific values and ranges.

Upon receipt of a successful CCMP response, a conference control client may then perform a "retrieve" operation per Section 7.2 to get a specific conference blueprint.

In the case of a response code of "requestTimeout", a conference control client MAY re-attempt the request within a period of time that would be specific to a conference control client or conference control server.

The response codes of "modified", "deleteParentFailed" and "modifyFailedProtected" are not applicable to the "options" operation and should be treated as "serverInternalError", the handling of which is specific to the conference control client.

A CCMP response containing any other response code is an error and the handling is specific to the conference control client. Typically, an error for an "options" operation indicates a configuration problem in the conference control server or in the client.

8.3.2.2. Retrieve Response

The CCMP response for a "retrieve" operation containing a response code of "success" MUST contain the full XML document describing the

conference object in its current state including all inherited values. Elements may be marked by attributes, in particular, whether they are specific to this instance or have been inherited from the parent node.

If a response code of "objectNotFound" is received in the CCMP response, it is RECOMMENDED that a conference control client attempt to retrieve another conference blueprint, if more than one had been received in response to the "options" operation.

If a response code of "requestTimeout" is received in the CCMP response, a conference control client MAY re-attempt the request within a period of time that would be specific to a conference control client or conference control server.

Response codes such as "notImplemented" and "forbidden" indicate that a subsequent "retrieve" would not likely be sucessful. Handling of these and other response codes is specific to the conference control client. For example, in the case of some clients an "options" operation might be performed again or another conference control server may be accessed.

The response codes of "modified", "deleteParentFailed" and "modifyFailedProtected" are not applicable to the "retrieve" operation and SHOULD be treated as "serverInternalError", the handling of which is specific to the conference control client.

8.3.2.3. Create Response

If the CCMP response to the "create" operation contains a response code of "success", the response MUST also contain either the XCON-URI for the conference object or the XCON-USERID if the request was to create a conference user.

If the CCMP response to the "create" operation contains a response code of "modified", the response MUST also contain the XCON-URI for the conference object and the XML document associated with that conference object. For example, in the case where the conference object contained a calendar element, the conference server may only offer a subset of the dates requested, thus the updated dates are included in the returned XML document.

In the case of a response code of "requestTimeout", a conference control client MAY re-attempt the request within a period of time that would be specific to a conference control client or conference control server.

Response codes such as "unauthorized", "forbidden" and

"operationNotAllowed" indicate the client does not have the appropriate permissions, there is an error in the permissions, or there is a system error in the client or conference control server, thus re-attempting the request would likely not succeed.

The response codes of "deleteParentFailed" and "modifyFailedProtected" are not applicable to the "create" operation and SHOULD be treated as "serverInternalError", the handling of which is specific to the conference control client.

Any other response code indicates an error in the client or conference control server (e.g., "forbidden", "badRequest") and the handling is specific to the conference control client.

8.3.2.4. Change Response

If the CCMP response to the "change" operation contains a response code of "success", the response also contains the XCON-URI for the conference object that was changed.

If the CCMP response to the "change" operation contains a response code of "modified", the response MUST contain the XCON-URI for the conference object and the XML document associated with that conference object. For example, a conferencing system may not have the resources to support specific capabilities that were changed, such as <codecs> in the <available-media>, thus the <codecs> supported are included in the returned XML document.

If the CCMP response code of "requestTimeout" is received, a conference control client MAY re-attempt the request within a period of time that would be specific to a conference control client or conference control server.

Response codes such as "unauthorized", "forbidden", "operationNotAllowed" and "modifyFailedProtected" indicate the client does not have the appropriate permissions, the conference is locked, there is an error in the permissions, or there is a system error in the client or conference control server, thus re-attempting the request would likely not succeed.

The response code of "deleteParentFailed" is not applicable to the "modify" operation and SHOULD be treated as "serverInternalError", the handling of which is specific to the conference control client.

Any other response code indicates an error in the client or conference control server (e.g., "forbidden", "badRequest") and the handling is specific to the conference control client.

8.3.2.5. Delete Response

If the CCMP response to the "delete" operation contains a response code of "success", the response MUST contain the XCON-URI for the conference object that was deleted.

The response code of "deleteParentFailed" indicates that the conference object could not be deleted because it is the Parent of another conference object that is in use. In this case, the response also includes the XCON-URI for the conference object.

If a response code of "requestTimeout" is received, a conference control client MAY re-attempt the request within a period of time that would be specific to a conference control client or conference control server.

Response codes such as "unauthorized", "forbidden" and "operationNotAllowed" indicate the client does not have the appropriate permissions, the conference is locked, there is an error in the permissions, or there is a system error in the client or conference control server, thus re-attempting the request would likely not succeed.

The response code of "modifyFailedProtected" is not applicable to the "delete" operation and SHOULD be treated as "serverInternalError", the handling of which is specific to the conference control client.

Any other response code indicates an error in the client or conference control server (e.g., "forbidden", "badRequest") and the handling is specific to the conference control client.

9. Protocol Parameters

This section describes in detail the parameters that are used for the CCMP protocol.

9.1. Operation Parameter

The "operation" attribute is a mandatory token included in all CCMP request and response messages. This document defines five possible values for this parameter: "options", "retrieve", "create", "modify" and "delete". The details for the specific processing based on the operation is provided in Section 9.1.

9.2. Request ID Parameter

The "requestID" attribute is a mandatory token included in all CCMP request and response messages. The "requestID" is used to correlate the requests with the appropriate response.

9.3. ConfObjID Parameter

The "confObjID" attribute is an optional URI included in the CCMP request and response messages. This attribute is required in the case of an "operation" of "change" and "delete" in the CCMP request and response messages. This attribute is the XCON-URI which is the target for the specific operation.

The only case when this attribute would not be included in the CCMP request for an operation of "create" is when there is no "confObjID" attribute in the CCMP request and the CCMP request contained a confUserID. This latter case is the mechanism whereby a conference control client can request the creation of a new conference user, as detailed in Section 9.4.

In the cases where the "conference-info" parameter Section 9.7 is also included in the requests and responses, the "confObjID" MUST match the XCON-URI in the "entity" attribute.

9.4. ConfUserID Parameter

The "confUserID" attribute is an optional URI included in the CCMP request and response messages. This is an XCON-USERID for the conference control client initiating the request.

This attribute is required in the case of an "operation" of "create", "change" and "delete" in the CCMP request message. In these cases, it is used to determine if the conference control client has the authority to perform the operation. Note that the details for authorization and related policy are specified in a separate document [TBD]. For any CCMP request with a "create" operation the conference control server MUST create a new conference user and return the associated confUserID in the response. In the case where the confUserID in the request has already been allocated, this request may be the creation of a confUserID for the conference control client to take on an additional role.

This attribute is required in the CCMP response message in the case of an "operation" of "create", which also contained a "confUserID" attribute in the CCMP request with no "confObjID".

9.5. ResponseCode Parameter

The "responseCode" attribute is a mandatory parameter in all CCMP response messages. The values for each of the "responseCode" values are detailed in Section 8.3.1 with the associated processing described in Section 8.3.2.

9.6. Blueprints Parameter

The "blueprints" attribute is a optional parameter in the CCMP request and response messages. In the case of a CCMP request with an operation of "options", the CCMP response includes the "blueprints" supported by the conference control server. The "blueprints" attribute is comprised of a list of blueprints supported by the specific conference server and includes a conference system specific "blueprintName" and a "confObjID" in the form of an XCON-URI for each of the blueprints.

The "blueprints" attribute is required for a CCMP request with an operation of "retrieve".

9.7. Conference-info Parameter

The "conference-info" element contains the data for the conference object that is the target for the CCMP request operations for "create", "change" and "delete" operations. It is returned in a CCMP response if the CCMP response contains a responseCode of "modified" or if the original CCMP request for the "create" operation did not contain a "conference-info" element. The latter case would occur if a conference control clients intends to create a conference object based on a default provided by a conferening system.

The details on the information that may be included in the "conference-info" element MUST follow the rules as specified in the XCON Data Model document [8]. The conference control client and conference control server MUST follow those rules in generating the "conference-info" in any of the CCMP request and response messages.

Note that the "conference-info" element is not explicitly shown in the XML schema Section 12 due to XML schema constraints.

10. Examples

The examples below omits the standard SOAP header and wrappers, i.e., the examples below contain simply the <body> of the requests and responses.

10.1. Creating a New Conference

The first example creates a new conference.

```
<ccmpRequest xmlns="urn:ietf-params:xml:ns:xcon:ccmp">
<requestID> 99 </requestID>
<operation>create</operation>

<conference-info
  xmlns="urn:ietf:params:xml:ns:conference-info"
  version="1">
  <conference-description>
   <parent>http://example.com/conf200</parent>
   <subject>Agenda: This month's goals</subject>
   <conf-uris>
     <entry>
       <uri>sips:conf223@example.com</uri>
       <purpose>participation</purpose>
     </entry>
   </conf-uris>
   <service-uris>
     <entry>
      <uri>http://sharep/salesgroup/</uri>
      <purpose>web-page</purpose>
     </entry>
     <entry>
      <uri>http://example.com/conf233</uri>
      <purpose>control</purpose>
     </entry>
    </service-uris>
   </conference-description>
</conference-info>

</ccmpRequest>
```

Figure 2: Create Request Example

The response to this request is shown below; it returns the object identifier as a URL and the final conference description, which may modify the description offered by the user.

```
   <ccmpResponse xmlns="urn:ietf-params:xml:ns:xcon:ccmp">
     <requestID> 99 </requestID>
     <operation>create</operation>
     <responseCode> modified </responseCode>
     <confObjID> xcon:confxyz987@example.com </confObjID>
     <confUserID> userA-confxyz987 </confUserID>

     <conference-info
       xmlns="urn:ietf:params:xml:ns:conference-info"
       version="1">
       <entity> xcon:confxyz987@example.com </entity>
       <conference-description>
        <parent>http://example.com/conf200</parent>
        <subject>Agenda: This month's goals</subject>
        <conf-uris>
          <entry>
            <uri>sips:conf223@example.com</uri>
            <purpose>participation</purpose>
          </entry>
        </conf-uris>
        <service-uris>
          <entry>
           <uri>http://sharep/salesgroup/</uri>
           <purpose>web-page</purpose>
          </entry>
          <entry>
           <uri>http://example.com/conf233</uri>
           <purpose>control</purpose>
          </entry>
         </service-uris>

         <allowed-users-list>
          <target uri="sip:alice@example.com" method="dial-out"/>
          <target uri="sip:bob@example.com" method="dial-out"/>
          <target uri="sip:userA@example.com" method="dial-in"/>
         </allowed-users-list>

          <!-- Addt'l modified conference description including users alice,
               bob and userA... -->

       </conference-description>
     </conference-info>

     </ccmpResponse>
```

Figure 3: Create Response Example

10.2. Creating a New Conference User

The request below creates a new conference user, independent of a specific conference object.

```
<ccmpRequest xmlns="urn:ietf-params:xml:ns:xcon:ccmp">
    <requestID> 101 </requestID>
    <operation>create</operation>

    <conference-info
        xmlns="urn:ietf:params:xml:ns:conference-info"
        version="1">

      <conference-description>
        <user entity="sip:cliff@example.com">
         <role>observer</role>
         <type><dial-in/></type>
        </user>
      </conference-description>
    </conference-info>

 </ccmpRequest>
```

Figure 4: Create User Example

The response to this request is shown below; it returns the conference user identifier.

```
<ccmpResponse xmlns="urn:ietf-params:xml:ns:xcon:ccmp">
  <requestID> 101 </requestID>
  <operation>create</operation>
  <responseCode> success </responseCode>
  <confUserID> userC-confxyz987 </confUserID>
</ccmpResponse>
```

Figure 5: Create Response Example

10.3. Adding a User to a Conference

The request below adds a user to the conference identified by the XCON-URI.

```
<ccmpRequest xmlns="urn:ietf-params:xml:ns:xcon:ccmp">
    <requestID> 100 </requestID>
    <operation>change</operation>
    <confObjID> xcon:confxyz987@example.com </confObjID>

    <conference-info
        xmlns="urn:ietf:params:xml:ns:conference-info"
        version="1">
      <entity> xcon:confxyz987@example.com </entity>
      <conference-description>

        <user entity="sip:bob@example.com">
         <role>participant</role>
         <type><dial-out/></type>
        </user>
      </conference-description>
    </conference-info>

 </ccmpRequest>
```

Figure 6: Add User Example

The response to this request is shown below; it returns the conference user identifier.

```
   <ccmpResponse xmlns="urn:ietf-params:xml:ns:xcon:ccmp">
     <requestID> 100 </requestID>
     <operation>create</operation>
     <responseCode> success </responseCode>
     <confObjID> xcon:confxyz987@example.com </confObjID>
     <confUserID> userA-confxyz987 </confUserID>

     <!-- Note that additional conference-info may also be returned depending
upon
      Bob's privileges. In this case, the response code would be "modified". --
>

     <conference-info
           xmlns="urn:ietf:params:xml:ns:conference-info"
           version="1">
         <entity> xcon:confxyz987@example.com </entity>
         <conference-description>
           <user entity="sip:bob@example.com">
            <role>participant</role>
            <type><dial-out/></type>
           </user>
         </conference-description>
       </conference-info>

   </ccmpResponse>
```

Figure 7: Add User Response Example

11. Transaction Model

The transaction model for CCMP complies fully with SOAP version 1.2 as defined by W3C in [5] and [6].

12. XML Schema

This section provides the XML schema definition of the "application/ccmp+xml" format.

```
   <?xml version="1.0" encoding="utf-8" ?>
   <xs:schema
       targetNamespace="urn:ietf:params:xml:ns:xcon:ccmp"
```

```
       xmlns:tns="urn:ietf:params:xml:ns:xcon:ccmp"
```

```
       xmlns:xs="http://www.w3.org/2001/XMLSchema"

       <!-- CCMP-REQUEST-TYPE definition-->
       <xs:complexType name="ccmp-request-type">
         <xs:sequence minOccurs="1" maxOccurs="unbounded"

           <xs:attribute name="operation"
            type="tns:operationType" use="required"/>

           <xs:attribute name="requestId"
            type="xs:string" use="required"/>

           <!--The XCON-URI for the target conference object.
               Required for "change" and "delete".
               Optional for "create" -->
           <xs:attribute name="confObjID" type="xs:anyURI" use="optional"/>

           <!--The XCON-USERID for the conference Control client.
               Required for "change" and "delete".
               Optional for "options", "retrieve" and "create". -->
           <xs:attribute name="confUserID" type="xs:token" use="optional"/>

           <!-- The blueprint for a "retrieve" request -->
           <xs:attribute name="blueprint"
                         type="tns:blueprintListType"
                         use="optional"/>

           <!-- For conference-info elements in the request
                and extensibility. -->
           <xs:anyAttribute namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>

         </xs:sequence>

       </xs:complexType>


       <!-- CCMP-RESPONSE-TYPE definition -->
       <xs:complexType name="ccmp-response-type">

         <xs:sequence minOccurs="1" maxOccurs="unbounded">

           <xs:attribute name="operation"
                         type="tns:operationType" use="required"/>

           <xs:attribute name="requestID" type="xs:token" use="required"/>

           <!--The XCON-URI for the target conference object.
```

```
            Included in the response for all but "options".-->
         <xs:attribute name="confObjID" type="xs:anyURI" use="optional"/>

         <!--The XCON-USERID for the conference Control client.
             Required for "create" for User. Optional for all others. -->
         <xs:attribute name="confUserID" type="xs:token" use="optional"/>

         <xs:attribute name="responseCode"
          type="tns:response-code-type" use="required"/>

         <!-- The list of blueprints for an "options" request -->
         <xs:element  name="blueprints"
                      type="tns:blueprintListType"
                      use="optional"/>

         <!-- This is where any conference-info elements and extensibility.
-->
         <xs:anyAttribute namespace="##other" processContents="lax"
             minOccurs="0" maxOccurs="unbounded"/>

       </xs:sequence>

    </xs:complexType>

    <!-- OPERATION TYPE for all requests and responses -->

    <xs:simpleType name="operationTypes">
       <xs:restriction base="xs:token">
              <xs:enumeration value="options"/>
              <xs:enumeration value="retrieve"/>
              <xs:enumeration value="create"/>
              <xs:enumeration value="change"/>
              <xs:enumeration value="delete"/>
       </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="operationType">
           <xs:union memberTypes='tns:operationTypes xs:token'/>
    </xs:simpleType>


    <!-- BLUEPRINT TYPE for "options" response and "retrieve" request -->

    <xs:complexType name="blueprintType">
       <xs: attribute name="blueprintName" type="xs:token" use="required" />
       <xs: attribute name="confObjID" type="xs:anyURI" use="required"/>
    </xs:complexType>

     <xs:simpleType name="blueprintListType">
```

```
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
              <xs:element name="blueprint"
                          type="ccmp:blueprintType"/>
          </xs:sequence>
       </xs:simpleType>

      <!-- RESPONSE-CODE-TYPE definition -->
       <xs:simpleType name="response-code">
         <xs:restriction base="xs:token">
                 <xs:enumeration value="success"/>
                 <xs:enumeration value="pending"/>
                 <xs:enumeration value="modified"/>
                 <xs:enumeration value="badRequest"/>
                 <xs:enumeration value="unauthorized"/>
                 <xs:enumeration value="forbidden"/>
                 <xs:enumeration value="objectNotFound"/>
                 <xs:enumeration value="operationNotAllowed"/>
                 <xs:enumeration value="deleteFailedParent"/>
                 <xs:enumeration value="modifyFailedProtected"/>
                 <xs:enumeration value="requestTimeout"/>
                 <xs:enumeration value="serverInternalError"/>
                 <xs:enumeration value="notImplemented"/>
             </xs:restriction>
       </xs:simpleType>

       <xs:simpleType name="response-code-type">
            <xs:union memberTypes='tns:response-code xs:token' />
       </xs:simpleType>

      <!-- CONF-CTL-REQUEST-TYPE element -->
      <xs:element name="ccmpRequest" type="tns:ccmp-request-type"/>

       <!-- CONF-CTL-RESPONSE-TYPE element-->
      <xs:element name="ccmpResponse" type="tns:ccmp-response-type"/>


   </xs:schema>
```

Figure 8

13. WSDL Definition

The following provides the WSDL definition for conference control and manipulation, using the the XML schema defined in Section 12 as a basis.

```
<?xml version="1.0" encoding="UTF-8"?>
   <definitions name="CCMP"
     xmlns="http://schemas.xmlsoap.org/wsdl/"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:ccmp="urn:ietf:params:xml:ns:ccmp"
     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
     xmlns:tns="urn:ietf:params:xml:ns:xcon:ccmp"
     targetNamespace="urn:ietf:params:xml:ns:xcon:ccmp">

     <xs:import
        namespace="urn:ietf:params:xml:ns:xcon:ccmp"
        schemaLocation="ccmp.xsd"/>

     <message name="CCMPRequestMessage">
       <part name="body" element="ccmp:request"/>
     </message>
     <message name="CCMPReponseMessage">
       <part name="body" element="ccmp:response"/>
     </message>

     <wsdl:portType name="CCMPPortType">
       <wsdl:operation name="confOperation" parameterOrder="body">
         <wsdl:input message="tns:CCMPRequestMessage"/>
         <wsdl:output message="tns:CCMPResponseMessage"/>
       </wsdl:operation>
     </wsdl:portType>

     <wsdl:binding name="ccpSoapBinding" type="tns:CCMPPortType">
         <wsdlsoap:binding style="rpc"
           transport="http://schemas.xmlsoap.org/soap/http"/>
         <wsdl:operation name="confOperation">
            <wsdlsoap:operation soapAction=""/>
            <wsdl:input>
               <wsdlsoap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               use="encoded"/>
            </wsdl:input>
            <wsdl:output>
               <wsdlsoap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               use="encoded"/>
            </wsdl:output>
         </wsdl:operation>
     </wsdl:binding>
```

```
    <wsdl:service name="CCMP">
      <wsdl:port binding="tns:ccpSoapBinding" name="CCMPPortType">
       <wsdlsoap:address location="http://www.example.com"/>
      </wsdl:port>
    </wsdl:service>

  </definitions>
```

Figure 9

14. IANA Considerations

This document registers a new XML namespace, a new XML schema, and the MIME type for the schema. This document also defines registries for the CCMP operation types and response codes.

14.1. URN Sub-Namespace Registration

This section registers a new XML namespace, ""urn:ietf:params:xml:ns:xcon:ccmp"".

URI: "urn:ietf:params:xml:ns:xcon:ccmp"
Registrant Contact: IETF, XCON working group, (xcon@ietf.org), Mary Barnes (mary.barnes@nortel.com).
XML:

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

14.2. XML Schema Registration

This section registers an XML schema as per the guidelines in [3].

URI: urn:ietf:params:xml:schema:xcon:ccmp
Registrant Contact: IETF, XCON working group, (xcon@ietf.org), Mary Barnes (mary.barnes@nortel.com).
Schema: The XML for this schema can be found as the entirety of Section 12 of this document.

14.3. MIME Media Type Registration for 'application/ccmp+xml'

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

To: ietf-types@iana.org
Subject: Registration of MIME media type application/ccmp+xml
MIME media type name: application
MIME subtype name: ccmp+xml
Required parameters: (none)
Optional parameters: charset
Indicates the character encoding of enclosed XML. Default is UTF-8.
Encoding considerations: Uses XML, which can employ 8-bit characters, depending on the character encoding used. See RFC 3023 [9], section 3.2.
Security considerations: This content type is designed to carry protocol data related conference control. Some of the data could be considered private and thus should be protected.
Interoperability considerations: This content type provides a basis for a protocol
Published specification: RFC XXXX [[NOTE TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number for this specification.]]
Applications which use this media type: Centralized Conferencing control clients and servers.
Additional Information: Magic Number(s): (none)
File extension(s): .xml
Macintosh File Type Code(s): (none)
Person & email address to contact for further information: Mary Barnes <mary.barnes@nortel.com>
Intended usage: LIMITED USE
Author/Change controller: The IETF
Other information: This media type is a specialization of application/xml [9], and many of the considerations described there also apply to application/ccmp+xml.

14.4. CCMP Protocol Registry

This document requests that the IANA create a new registry for the CCMP protocol including an initial registry for operation types and response codes.

14.4.1. CCMP Operations

The "operation" are included in CCMP messages as described in Section 9.1 and defined in the 'operationType' in the XML schema in Section 12. The following summarizes the requested registry:

Related Registry: CCMP Operation Registry
Defining RFC: RFC XXXX [NOTE TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number for this specification.]
Registration/Assignment Procedures: New operations are allocated on a first-come/first-serve basis with specification required.
Registrant Contact: IETF, XCON working group, (xcon@ietf.org), Mary Barnes (mary.barnes@nortel.com).

This section pre-registers the following five initial operations:

options: Used by a conference control client to query a conferncing system for its capabiliites.
retrieve: Used by a conference control client to retrieve a specific blueprint.
create: Used by a conference control client to create a new conference object or new conference user.
modify: Used by a conference control client to modify a conference object(s).
delete: Used by a client to delete a conference object(s).

14.4.2. CCMP Response Codes

The following summarizes the requested registry for CCMP Response codes:

Related Registry: CCMP Response Code Registry
Defining RFC: RFC XXXX [NOTE TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number for this specification.]
Registration/Assignment Procedures: New response codes are allocated on a first-come/first-serve basis with specification required.
Registrant Contact: IETF, XCON working group, (xcon@ietf.org), Mary Barnes (mary.barnes@nortel.com).

This section pre-registers the following thirteen initial response codes as described above in Section 8.3:

success: This code indicates that the request was successfully processed.
pending: This code indicates that the notification is to follow.
modified: This code indicates that the object was created, but may differ from the request.
badRequest: This code indicates that the request was badly formed in some fashion.
unauthorized: This code indicates that the user was not authorized for the specific operation on the conference object.
forbidden: This code indicates that the specific operation is not valid for the target conference object.
objectNotFound: This code indicates that the specific conference object was not found.
operationNotAllowed: This code indicates that the specific operation is not allowed for the target conference object (e.g.., due to policies, etc.)
deleteFailedParent: This code indicates that the conferencing system cannot delete the specific conference object because it is a parent for another conference object.
modifyFailedProtected: This code indicates that the target conference object cannot be changed (e.g., due to policies, roles, privileges, etc.).
requestTimeout: This code indicates that the request could not be processed within a reasonable time, with the time specific to a conferencing system implementation.
serverInternalError: This code indicates that the conferencing system experienced some sort of internal error.
notImplemented: This code indicates that the specific operation is not implemented on that conferencing system.

15. Security Considerations

Access to conference control functionality needs to be tightly controlled to keep attackers from disrupting conferences, adding themselves to conferences or engaging in theft of services. Implementors need to deploy standard HTTP and SOAP authentication and authorization mechanisms. Since conference information may contain secrets such as participant lists and dial-in codes, all conference control information SHOULD be carried over TLS (HTTPS).

16. Acknowledgments

The authors appreciate the feedback provided by Simon Pietro Romano, Dave Morgan and Pierre Tane.

17. References

17.1. Normative References

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

[2] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[3] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[4] Chinnici, R., Moreau, J., Ryman, A., and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language", W3C CR CR-wsdl20-20051215, December 2005.

[5] Gudgin, M., Hadley, M., Moreau, J., Mendelsohn, N., and H. Nielsen, "SOAP Version 1.2 Part 1: Messaging Framework", World Wide Web Consortium FirstEdition REC-soap12-part1-20030624, June 2003, <http://www.w3.org/TR/2003/REC-soap12-part1-20030624>.

[6] Mendelsohn, N., Moreau, J., Hadley, M., Nielsen, H., and M. Gudgin, "SOAP Version 1.2 Part 2: Adjuncts", World Wide Web Consortium FirstEdition REC-soap12-part2-20030624, June 2003, <http://www.w3.org/TR/2003/REC-soap12-part2-20030624>.

[7] Barnes, M., Boulton, C., and O. Levin, "A Framework for Centralized Conferencing", draft-ietf-xcon-framework-10 (work in progress), November 2007.

[8] Novo, O., Camarillo, G., Morgan, D., and R. Even, "Conference Information Data Model for Centralized Conferencing (XCON)", draft-ietf-xcon-common-data-model-09 (work in progress), February 2008.

17.2. Informative References

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

[10] 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.

[11] Lennox, J., Wu, X., and H. Schulzrinne, "Call Processing

Language (CPL): A Language for User Control of Internet Telephony Services", RFC 3880, October 2004.

[12] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966, December 2004.

[13] Camarillo, G., Srinivasan, S., Even, R., and J. Urpalainen, "Conference Event Package Data Format Extension for Centralized Conferencing (XCON)", draft-ietf-xcon-event-package-00 (work in progress), February 2008.

[14] Royer, D., "iCalendar in XML Format (xCal-Basic)", draft-royer-calsch-xcal-03 (work in progress), October 2005.

Authors' Addresses

Mary Barnes
Nortel
2201 Lakeside Blvd
Richardson, TX

Email: mary.barnes@nortel.com

Chris Boulton
Avaya
Building 3
Wern Fawr Lane
St Mellons
Cardiff, South Wales CF3 5EA

Email: cboulton@avaya.com

Henning Schulzrinne
Columbia University
Department of Computer Science
450 Computer Science Building
New York, NY 10027

Email: hgs+xcon@cs.columbia.edu

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).