XCON
Internet-Draft
Expires: June 12, 2006

O. Novo
Ericsson
C. Jennings
Cisco Systems
A. Roach
Estacado Systems
G. Camarillo
Ericsson

December 9, 2005

Conference State Change Protocol (CSCP)
draft-jennings-xcon-cscp-02.txt

Status of this Memo

Copyright Notice

Abstract

Conference State Control Protocol (CSCP) is a means to modify the state in a conference service. It extends the Binary Floor Control Protocol and adds commands to get, set, add, and delete fields in the

conference state.

Table of Contents

1. Conventions 3
2. Overview 3
3. New Primitives 3
4. New Attributes 4
4.1. ELEMENT-ID 4
4.2. NAME 5
4.3. VALUE 5
5. Behavior 6
5.1. Retrieving Attribute Values 6
5.2. Setting Attribute Values 7
5.3. Adding Elements 8
5.4. Deleting an Element 9
5.5. Deleting an Attribute 9
5.6. Errors 10
6. Server Behaviour 11
7. Example 11
8. To Do Items 13
9. IANA Considerations 13
9.1. Attribute Registration 13
9.2. Primitive Registration 13
9.3. Error Code Registration 14
10. Security Considerations 14
11. Acknowledgments 14
12. References 15
12.1. Normative References 15
12.2. Informative References 15
Authors' Addresses 16
Intellectual Property and Copyright Statements 17

1. Conventions

This document extends the Binary Floor Control Protocol (BFCP) [1] and makes no sense if you have not read that document. This document uses the terminology from the conference framework[4].

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 [2].

2. Overview

CSCP is a client server protocol used to change the state of a conference object. Note that the CSCP is just a protocol to manipulate the conference object. It is not a general XML protocol manipulation mechanism.

A client sends the server a request representing a sequence of commands. Each command can set, get, add, or delete a single field in the conference object. Changes to the conference object are performed on a hierarchal set of elements and unique attributes within those elements. A series of changes can be pipelined in a single BFCP message [1]. The server executes each action in series. If one of them fails, the server returns an error for the action that failed and does not execute any of the actions after that. Each individual action is atomic but the pipelined series is not.

The item that a command applies to is specified by a unique ID and, where appropriate, attribute name. The ID is an unsigned 32 bit integer called the ELEMENT-ID. How the server discovers the ELEMENT-ID of a given element is outside of CSCP. Typically this is done by using the conferencing framework notification service [5]. Each field in the data received in the notification contains a unique field ID attribute that can be used in BFCP requests.

The values for fields are transferred as UTF-8 [3] encoded strings.

3. New Primitives

This document updates Table 1 in BFCP [1] with the following primitives:

Value	Primitive	Direction
14	GetRequest	P -> S ; Ch -> S
15	GetInfo	P <- S ; Ch <- S
16	SetRequest	P -> S ; Ch -> S
17	SetAck	P <- S ; Ch <- S
18	AddElementRequest	P -> S ; Ch -> S
19	AddElementInfo	P <- S ; Ch <- S
20	DelElementRequest	P -> S ; Ch -> S
21	DelElementAck	P <- S ; Ch <- S
22	DelAttributeRequest	P -> S ; Ch -> S
23	DelAttributeAck	P <- S ; Ch <- S

S: Floor Control Server
P: Floor Participant
Ch: Floor Chair

Table 1: CSCP primitives

4. New Attributes

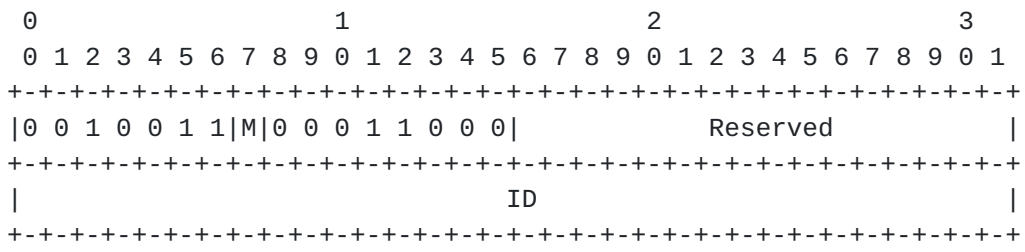
This document updates Table 2 in BFCP [1] and Table 1 in [6] with the following new attributes:

Type	Attribute	Format
19	ELEMENT-ID	Unsigned32
20	NAME	OctetString
21	VALUE	OctetString

Table 2: CSCP attributes

4.1. ELEMENT-ID

The following is the format of a ELEMENT-ID attribute:

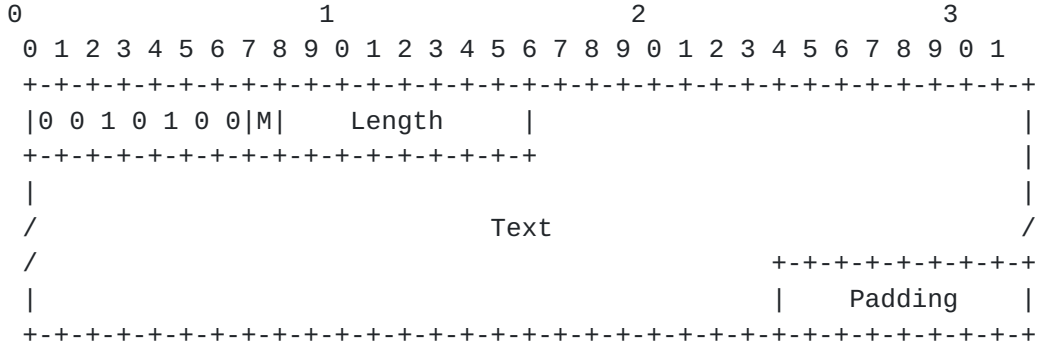


The reserved field MUST be set to 0 by the sender, and MUST be ignored by the receiver.

ID: This field contains a 32-bit value that uniquely identifies an element within a conference. The ID values 0 and 0xFFFFFFFF are reserved and MUST NOT be used.

4.2. NAME

The following is the format of a NAME attribute:

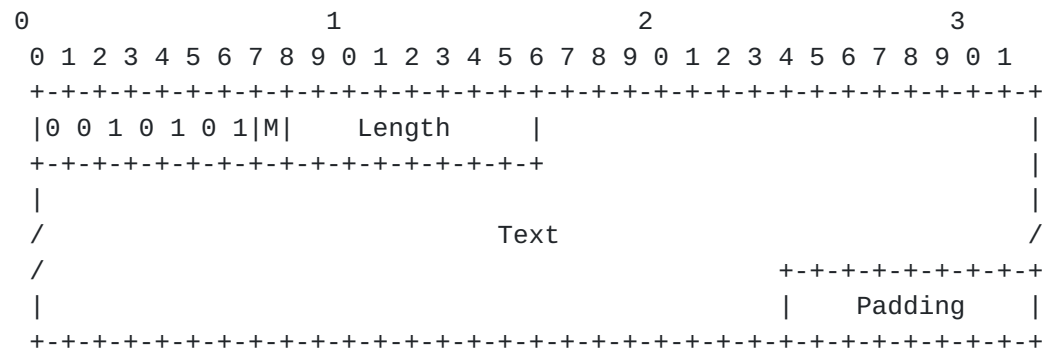


Text: this field contains UTF-8 [3] encoded text.

Padding: one, two, or three bytes of padding added so that the size of the VALUE attribute is 32-bit aligned. The Padding bits SHOULD be set to zero by the sender and MUST be ignored by the receiver. If the attribute is already 32-bit aligned, no padding is included.

4.3. VALUE

The following is the format of a VALUE attribute:



Text: this field contains the string encoding of the value. For fields that are an integer, this is a base 10 encoded ASCII integer. For binary values it is sent as the ASCII characters 0 and 1.

[Open Issue: should it be a binary type?)]

[Open Issue: Is attribute retrieval actually necessary or even useful? This protocol is predicated on the client having access to the conference state via some other mechanism and is intended primarily for manipulation of such state.]

Padding: one, two, or three bytes of padding added so that the size of the VALUE attribute is 32-bit aligned. The Padding bits SHOULD be set to zero by the sender and MUST be ignored by the receiver. If the attribute is already 32-bit aligned, no padding is included.

5. Behavior

5.1. Retrieving Attribute Values

Clients request the values of various attribute fields by sending the GetRequest message to the server. A single GetRequest MUST NOT contain the same {ELEMENT-ID,NAME} combination more than once; otherwise, it is not possible to tell which of the multiple operations has caused an error. If the request is not well-formed, the server sends the Error message code 3 (Unknown Primitive), as described in [1].

If the client inserts more than one ELEMENT-ID, the server will treat the request as an atomic package. That is, it will either send all the attributes information the client wishes to retrieve or sends an error message.

If the NAME field is omitted for a given ELEMENT-ID, then the request applies to the CDATA associated with the indicated element. The following is the format of the message.

```
GetRequest    =   (COMMON-HEADER)
                  1*( (ELEMENT-ID)
                      [NAME])
```

The request contains all the ELEMENT-ID and NAME elements for the fields the client wishes to retrieve. The values of the requested fields are returned in a GetInfo message.

If the client is not authorized to perform the operation being requested, the Error message code 5 (Unauthorized Operation), as described in [1] is sent. If the client requests some attributes unknown from the server, the server sends the Error message 16 (Attribute could not be deleted).

```
GetInfo       =       (COMMON-HEADER)
                   1*( (ELEMENT-ID)
                       (NAME)
                       (VALUE))
```

5.2. Setting Attribute Values

The SetRequest asks for the values of one or more fields to be changed. A single SetRequest MUST NOT contain the same combination of {ELEMENT-ID,NAME} more than once. If the request is not well-formed, the server sends the Error message code 3 (Unknown Primitive), as described in [1].

If the NAME field is omitted for a given ELEMENT-ID, then the request applies to the CDATA associated with the indicated element.

If the client inserts more than one ELEMENT-ID, the server will treat the request as an atomic package. That is, it will either send all the attributes information the client wishes to retrieve or sends an error message.

If the SetRequest refers to an attribute which does not yet exist in the indicated element, then that attribute is added and set to the indicated value.

```
SetRequest =   (COMMON-HEADER)
               1*( (ELEMENT-ID)
                   [NAME]
                   (VALUE))
```

If a SetRequest is successful, the server responds with a SetAck. This SetAck contains the ELEMENT-ID and NAME fields for all fields which were successfully set. If any of the fields cannot be successfully set, an Error Code 13 (One or more attributes could not be succesfully set) returns the value of the first ELEMENT-ID that failed. If the client is not authorized to perform the operation being requested, the Error message code 5 (Unauthorized Operation), as described in [1] is sent.

```
SetAck  =       (COMMON-HEADER)
            1*( (ELEMENT-ID)
                (NAME))
```

5.3. Adding Elements

The AddElementRequest message contains a pair of ELEMENT-ID/NAME attributes. The ELEMENT-ID field refers to an existing element; the new element will be created as a child to the indicated element. The NAME field indicates the name of the element that is to be added. If the client is not authorized to perform the operation being requested, the Error message code 5 (Unauthorized Operation), as described in [1] is sent.

```
AddElementRequest = (COMMON-HEADER)
                    (ELEMENT-ID)
                    (NAME)
```

The server sends an AddElementInfo containing the newly created ELEMENT-ID. It is not possible to add multiple elements of the same name in a single request because there is no way to correlate errors with the addition that caused the error.

```
AddElementInfo =    (COMMON-HEADER)
                    (ELEMENT-ID)
```

AddElementInfo returns the new ELEMENT-ID for any elements that were created. Otherwise, if there is an error in creating the field, an Error message with code 13 (Element can not be created) is returned;

5.4. Deleting an Element

DelElementRequest requests that the server delete the specified element and all of its attributes and children.

```
DelElementRequest    = (COMMON-HEADER)
                       1*(ELEMENT-ID)
```

If the DelElementRequest is successful, the server responds with a DelElementAck, which includes the list of deleted elements. If the client is not authorized to perform the operation being requested, the Error message code 5 (Unauthorized Operation), as described in [1] is sent.

```
DelElementAck        = (COMMON-HEADER)
                       1*(ELEMENT-ID)
```

DelElementAck returns the ELEMENT-ID for any elements that were deleted. Otherwise, if there is an error deleting any fields, an Error message with code 15 (Element can not be deleted) with the ELEMENT-ID of the first field that could not be deleted is returned. The server will treat the request as an atomic package. That is, it will either sends all the attributes information the client wishes to retrieve or sends an error message.

5.5. Deleting an Attribute

DelAttributeRequest requests that the server delete the specified attribute. Upon success, the server responds with a DelAttributeAck with the list of deleted fields.

```
DelAttributeRequest    = (COMMON-HEADER)
                       1*( (ELEMENT-ID)
                           [NAME])
```

If the client is not authorized to perform the operation being requested, the Error message code 5 (Unauthorized Operation), as described in [1] is sent.

If the NAME field is omitted for a given ELEMENT-ID, then the request applies to the CDATA associated with the indicated element.

```
DelAttributeAck        = (COMMON-HEADER)
                       1*( (ELEMENT-ID)
                           (NAME))
```

DelAttributeAck returns the ELEMENT-ID for any elements that were deleted. Otherwise, if there is an error deleting any fields, an Error message with code 16 (Attribute can not be deleted) with the ELEMENT-ID and NAME of the first field that could not be deleted is returned. The server will treat the request as an atomic package. That is, it will either sends all the attributes information the client wishes to retrieve or sends an error message.

5.6. Errors

The format of the Error messages defined in [1] is also modified to have optional ELEMENT-ID and NAME attributes. When these attributes are present in an Error, it indicates the element (and, where appropriate, attribute) in the request that caused the error.

Note that we use new ELEMENT-ID and NAME attributes in the Error messages instead of defining a Error Specific Details for this error codes.

```
Error              =    (COMMON-HEADER)
                        (ERROR-CODE)
                        [ERROR-INFO]
                        [ELEMENT-ID]
                        [NAME]
                       *[EXTENSION-ATTRIBUTE]
```

The following is the definition of Error code meaning:

Value	Meaning
13	One or more attributes could not be succesfully set
14	Element could not be created
15	Element could not be deleted
16	Attribute could not be deleted
17	Unknown Attribute

Table 3: Error Code meaning

6. Server Behaviour

A conference server sending a notification to a client using the SIP event package [5] MUST included the ELEMENT-ID in every element of the XML document. The ELEMENT-ID values 0 and 0xFFFFFFFF are reserved and MUST NOT be used. It is RECOMMENDED that once an ELEMENT-ID value is used in an element within a conference, the same ELEMENT-ID value is not used in the same conference again in a different element.

7. Example

The following is an example of a partial conference information document extracted from the SIP event package [5]. Each element in this example has a new attribute "index" that contains the value to be used in the ELEMENT-ID attribute of a CSCP message acting on that element.

```
<?xml version="1.0" encoding="UTF-8"?>
 <conference-info index="1"
  xmlns="urn:ietf:params:xml:ns:conference-info"
  entity="sips:conf233@example.com"
  state="full" version="1">
 <!--
   CONFERENCE INFO
 -->
  <conference-description index="2">
   <subject index="3">Agenda: This month's goals</subject>
    <service-uris index="4">
     <entry index="5">
      <uri index="6">http://sharepoint/salesgroup/</uri>
      <purpose index="7">web-page</purpose>
     </entry>
    </service-uris>
   </conference-description>
 <!--
    CONFERENCE STATE
 -->
  <conference-state index="8">
   <user-count index="9">33</user-count>
  </conference-state>
 <!--
   USERS
 -->
  <users index="10">
   <user index="11" entity="sip:bob@example.com" state="full">
```

```
         <display-text index="12">Bob Hoskins</display-text>
      <!--
        ENDPOINTS
      -->
         <endpoint index="13"entity="sip:bob@pc33.example.com">
          <display-text index="14">Bob's Laptop</display-text>
          <status index="15">disconnected</status>
          <disconnection-method index="16">departed</disconnection-method>
          <disconnection-info index="17">
           <when index="18">2005-03-04T20:00:00Z</when>
           <reason index="19">bad voice quality</reason>
           <by index="20">sip:mike@example.com</by>
          </disconnection-info>
      <!--
        MEDIA
      -->
          <media index="21" id="1">
           <display-text index="22">main audio</display-text>
           <type index="23">audio</type>
           <label index="24">34567</label>
           <src-id index="25">432424</src-id>
           <status index="26">sendrecv</status>
          </media>
         </endpoint>
        </user>
      <!--
        USER
      -->
        <user index="27" entity="sip:alice@example.com" state="full">
         <display-text index="28">Alice</display-text>
      <!--
        ENDPOINTS
      -->
         <endpoint index="29" entity="sip:
4kfk4j392jsu@example.com;grid=433kj4j3u">
          <status index="30">connected</status>
          <joining-method index="31">dialed-out</joining-method>
          <joining-info index="32">
           <when index="33">2005-03-04T20:00:00Z</when>
           <by index="34">sip:mike@example.com</by>
          </joining-info>
      <!--
        MEDIA
      -->
          <media index="35" id="1">
           <display-text index="36">main audio</display-text>
           <type index="37">audio</type>
           <label index="38">34567</label>
```

<src-id index="39">534232</src-id>

```
          <status index="40">sendrecv</status>
        </media>
      </endpoint>
    </user>
  </users>
</conference-info>
```

So, changing the entity attribute of the <user> element from "sip:bob@pc33.example.com" to "Cullen.Fluffy@example.com" would be requested by sending a SetRequest with ELEMENT-ID = 13, NAME = "entity", and VALUE = "Cullen.Fluffy@example.com".

8. To Do Items

This needs to be lined up so that it can be correlated with the "SIP Event Package for Conference State" [5].

9. IANA Considerations

The following sections instruct the IANA to perform a set of actions.

9.1. Attribute Registration

The IANA is instructed to register the following new values under the Attribute subregistry under the BFCP Parameters registry.

Type	Attribute	Reference
19	ELEMENT-ID	[RFC XXXX]
20	NAME	[RFC XXXX]
21	VALUE	[RFC XXXX]

Table 4: New values of the BFCP Attribute subregistry

9.2. Primitive Registration

The IANA is instructed to register the following new values under the Primitive subregistry under the BFCP Parameters registry.

Value	Primitive	Reference
14	GetRequest	[RFC XXXX]
15	GetInfo	[RFC XXXX]
16	SetRequest	[RFC XXXX]
17	SetAck	[RFC XXXX]
18	AddElementRequest	[RFC XXXX]
19	AddElementInfo	[RFC XXXX]
20	DelElementRequest	[RFC XXXX]
21	DelElementAck	[RFC XXXX]
22	DelAttributeRequest	[RFC XXXX]
23	DelAttributeAck	[RFC XXXX]

Table 5: New values of the BFCP Primitives subregistry

9.3. Error Code Registration

The IANA is instructed to register the following new values under the Error Code subregistry under the BFCP Parameters registry.

Value	Meaning	Reference
13	One or more attributes could not be succesfully set	[RFC XXXX]
14	Element could not be created	[RFC XXXX]
15	Element could not be deleted	[RFC XXXX]
16	Attribute could not be deleted	[RFC XXXX]
17	Unknown Attribute	[RFC XXXX]

Table 6: New Values of the Error Code subregistry

10. Security Considerations

The security considerations of [1] and [6] apply. This memo does not introduce any known additional security risk.

11. Acknowledgments

Thanks to Roni Even for his comments.

12. References

12.1. Normative References

[1] Camarillo, G., "The Binary Floor Control Protocol (BFCP)", draft-ietf-xcon-bfcp-06 (work in progress), December 2005.

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

[3] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

12.2. Informative References

[4] Barnes, M. and C. Boulton, "A Framework and Data Model for Centralized Conferencing", draft-barnes-xcon-framework-02 (work in progress), February 2005.

[5] Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Conference State", draft-ietf-sipping-conference-package-12 (work in progress), July 2005.

[6] Camarillo, G., "Connection Establishment in the Binary Floor Control Protocol (BFCP)", draft-ietf-xcon-bfcp-connection-00 (work in progress).

Authors' Addresses

Oscar Novo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Oscar.Novo@ericsson.com

Cullen Jennings
Cisco Systems
170 West Tasman Drive
Mailstop SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 421 9990
Email: fluffy@cisco.com

Adam Roach
Estacado Systems
Dallas, TX
US

Email: adam@estacado.net

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Gonzalo.Camarillo@ericsson.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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