The Binary Floor Control Protocol (BFCP)
draft-camarillo-xcon-bfcp-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

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 September 30, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document specifies the Binary Floor Control Protocol (BFCP). BFCP is used between conference participants and floor control servers, and between floor chairs and floor control servers.
# Table of Contents

1. Introduction ........................................... 4
2. Terminology ........................................... 4
3. Overview of Operation ................................. 5
4. Packet Format ........................................ 5
   4.1 FIXED-HEADER Format .................................. 5
   4.2 Attribute Format .................................... 6
      4.2.1 FLOOR-ID ...................................... 7
      4.2.2 USER-ID ....................................... 7
      4.2.3 REQUEST-ID .................................... 7
      4.2.4 HUMAN-READABLE-INFO ............................ 7
      4.2.5 TOKEN .......................................... 8
      4.2.6 REQUEST-STATUS ................................ 8
      4.2.7 ERROR-CODE ................................... 9
      4.2.8 USER-DISPLAY-NAME .............................. 11
      4.2.9 USER-URI ...................................... 11
      4.2.10 PRIORITY ...................................... 11
5. Message Format .......................................... 11
   5.1 FloorRequest ....................................... 11
   5.2 FloorRelease ........................................ 12
   5.3 FloorRequestStatus .................................. 12
   5.4 FloorStatus ......................................... 12
   5.5 ChairAction ........................................ 13
   5.6 Ping ............................................... 13
   5.7 PingAck ............................................ 13
   5.8 Error ............................................... 14
6. bfcp and bfcps URI Definitions .......................... 14
7. Contacting a bfcp or bfcps URI .......................... 15
   7.1 Selecting a Transport Protocol ....................... 15
   7.2 Determining Port and IP Address ..................... 16
   7.3 Lower-Layer Security ................................ 17
   7.4 Conference ID, User ID, and Token Values ............ 17
8. Client Operations ........................................ 18
   8.1 Requesting a Floor .................................. 18
      8.1.1 Receiving Responses ............................. 19
   8.2 Cancelling a Floor Request ........................... 19
      8.2.1 Receiving Responses ............................. 20
   8.3 Releasing Floors .................................... 20
      8.3.1 Receiving Responses ............................. 21
   8.4 Checking the Liveness of a Server .................... 21
      8.4.1 Receiving Responses ............................. 21
   8.5 Responding to a Ping Message ........................ 22
9. Chair Operations ........................................ 22
   9.1 Obtaining Information about Floor Requests .......... 22
   9.2 Instructing the Floor Control Server ................ 22
10. Server Operations ...................................... 23
    10.1 Reception of a FloorRequest Message ............... 23
1. Introduction

Conference applications may need to manage the access to a set of shared resources such as the right to send media over a particular media stream. Floor control enables such applications to provide users with safe access to these resources.

The Requirements for Floor Control Protocol [11] list a set of requirements that need to be met by floor control protocols. The Binary Floor Control Protocol (BFCP), which is specified in this document, meets these requirements.

BFCP provides a means for a floor control server to grant or deny requests to access a given resource from users. Nevertheless, BFCP does not allow floor control servers to keep a particular user or set of users from actually accessing the resource. Enforcing policy decisions is outside the scope of BFCP. In any case, one possible way to implement policy enforcement is to have the floor control server provide the conference policy server (e.g., using CPCP [12]) with policies that the conference policy server will apply to the conference.

Floor control servers identify users using a 32-bit user ID. The way a conference server provides a user with its 32-bit user ID is out of the scope of this document.

BFCP assumes that conference servers, after authenticating a user, provides this user with a token, in addition to the user's user ID. The user's client needs to include this token in its requests to prove that they belong to the same user as the conference server authenticated previously. The way a conference server provides a user with such a token in a secure fashion falls outside the scope of this document.

Section 2 defines the terminology used throughout this document, Section 3 provides a non-normative overview of BFCP operation, and subsequent sections provide the normative specification of BFCP.

2. Terminology

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.

Floor: A permission to temporarily access or manipulate a specific shared resource or set of resources.
Floor chair: A user (or an entity) who manages one floor (grants, denies, or revokes a floor). The floor chair does not have to be a member in a conference.

Floor control: A mechanism that enables applications or users to gain safe and mutually exclusive or non-exclusive input access to the shared object or resource.

Floor control server: A logical entity that maintains the state of the floor(s) including which floors exists, who the floor chairs are, who holds a floor, etc. Requests to manipulate a floor are directed at the floor control server.

3. Overview of Operation

TBD: the Introduction and this section need more work.

4. Packet Format

BFCP packets consist of an 8-byte fixed header followed by attributes. All the protocol values MUST be sent in network byte order.

4.1 FIXED-HEADER Format

The following is the FIXED-HEADER format.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Ver |Reserved |  Primitive    |        Payload Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Conference ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Ver: the 3-bit version field MUST be set to 1 to indicate this version of BFCP.

Reserved: at this point, the 5 bits in the reserved field SHOULD be set to zero by the sender of the message and MUST be ignored by the receiver.

Primitive: this 8-bit field identifies the main purpose of the message. The following primitive values are defined:

<table>
<thead>
<tr>
<th>Value</th>
<th>Primitive</th>
<th>Direction</th>
</tr>
</thead>
</table>

0 FloorRequest C -> S
1 FloorRelease C -> S
2 FloorRequestStatus S -> C
3 FloorStatus S -> C ; S -> Ch
4 ChairAction Ch -> S
5 Ping C <-> S ; Ch <-> S
6 PingAck C <-> S ; Ch <-> S
7 Error S -> C ; S -> Ch

S: Server  C: Client  Ch: Chair

Payload Length: this 16-bit field contains length of the message in 4-byte units excluding the fixed header.

Conference ID: this 32-bit identifies the conference the message belongs to.

4.2 Attribute Format

BFCP attributes are encoded in TLV (Type-Length-Value) format. TLVs are 32-bit aligned.

```
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----------------------------------------------+
|    Type     |M|    Length     |                               |
+-----------------------------------------------|
|                                                               |
/                       Attribute Contents                      /
|                                                               |
+----------------------------------------------------------------+
```

Type: this 7-bit field contains the code for the attribute.

M: the 'M' bit, known as the Mandatory bit, indicates whether support of the attribute is required. If an unrecognized attribute with the 'M' bit set is received, the message is rejected.

Length: this 8-bit field contains the length of the attribute in bytes, excluding any padding defined for specific attributes. The Type, 'M' bit, and Length fields are included.

Attribute Contents: the contents of the different TLVs are defined in the following sections.
4.2.1 FLOOR-ID

The following is the format of the contents of the FLOOR-ID attribute, whose attribute type is 1.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type = 1   |M|    Length     |           Floor ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Floor ID: this field contains a 16-bit value that uniquely identifies a floor within a conference.

4.2.2 USER-ID

The following is the format of the contents of the USER-ID attribute, whose attribute type is 2.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type = 2   |M|    Length     |            User ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

User ID: this field contains a 16-bit value that uniquely identifies a user within a conference.

4.2.3 REQUEST-ID

The following is the format of the contents of the REQUEST-ID attribute, whose attribute type is 3.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type = 3   |M|    Length     |          Request ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Request ID: this field contains a 16-bit value that allows users to match a given message with its responses.

4.2.4 HUMAN-READABLE-INFO

The following is the format of the contents of the
HUMAN-READABLE-INFO attribute, whose attribute type is 4.

    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Type = 4   |M|    Length     |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
   |                                                               |
   /                             Text                              /
   /                                               +-+-+-+-+-+-+-+-+
   |                                               |    Padding    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Text: this field contains UTF-8 [10] encoded text.

In some situations, the contents of the Text field may be generated by an automata. If such automata has information about the preferred language of the receiver of a particular HUMAN-READABLE-INFO TLV, it may use this language to generate the Text field.

Padding: one, two, or three bytes of padding added so that the contents of the HUMAN-READABLE-INFO TLV is 32-bit aligned. The Padding bits SHOULD be set to zero by the sender and MUST be ignored by the receiver. If the TLV is already 32-bit aligned, no padding is needed.

4.2.5 TOKEN

The following is the format of the contents of the TOKEN attribute, whose attribute type is 5.

    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Type = 5   |M|    Length     |             Token             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Token: this field contains a 16-bit token used to authenticate the user.

4.2.6 REQUEST-STATUS

The following is the format of the contents of the REQUEST-STATUS attribute, whose attribute type is 6.
Request Status: this 16-bit field contains the status of the request, as described in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pending</td>
</tr>
<tr>
<td>1</td>
<td>Accepted</td>
</tr>
<tr>
<td>2</td>
<td>Granted</td>
</tr>
<tr>
<td>3</td>
<td>Denied</td>
</tr>
<tr>
<td>4</td>
<td>Cancelled</td>
</tr>
<tr>
<td>5</td>
<td>Released</td>
</tr>
<tr>
<td>6</td>
<td>Revoked</td>
</tr>
</tbody>
</table>

Queue Position: this 16-bit field contains, when applicable, the position of the floor request in the floor request queue at the server. If the Request Status value is different from Accepted, the floor control server does not implement a floor request queue, or the floor control server does not want to provide the client with this information, all the bits of this field SHOULD be set to zero.

A floor request is in Pending state if the floor control server needs to contact a floor chair in order to accept the floor request, but has not done it yet. Once the floor control chair accepts the floor request, the floor request is moved to the Accepted state.

Padding: two bytes of padding added so that the contents of the REQUEST-STATUS TLV is 32-bit aligned. The Padding bits SHOULD be set to zero by the sender and MUST be ignored by the receiver.

### 4.2.7 ERROR-CODE

The following is the format of the contents of the ERROR-CODE attribute, whose attribute type is 7.
Error Code: this 16-bit field contains an error code from the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Conference does not Exist</td>
</tr>
<tr>
<td>1</td>
<td>Authentication Failed</td>
</tr>
<tr>
<td>2</td>
<td>Unknown Mandatory TLV</td>
</tr>
<tr>
<td>3</td>
<td>Floor Request does not Exist</td>
</tr>
<tr>
<td>4</td>
<td>Unauthorized Operation</td>
</tr>
<tr>
<td>5</td>
<td>User does not Exist</td>
</tr>
<tr>
<td>6</td>
<td>Invalid Request-ID</td>
</tr>
</tbody>
</table>

Error Specific Details: Present only for certain Error Codes. In this document, only for Error Code 2 (Unknown Mandatory TLV). For Error Code 2, this field contains the Types of the TLVs (which were present in the message that triggered the Error message) that were unknown to the receiver, encoded as follows.

Padding: one, two, or three bytes of padding added so that the contents of the ERROR-CODE TLV is 32-bit aligned. If the TLV is already 32-bit aligned, no padding is needed.

The Padding bits SHOULD be set to zero by the sender and MUST be ignored by the receiver. Note all the Error Codes defined in this document but Error Code 2, result in a TLV which is already 32-bit aligned.
aligned (i.e., no need of padding). Error Code 2 results in a TLV that may need 2 bytes of padding.

4.2.8 USER-DISPLAY-NAME

The USER-DISPLAY-NAME attribute type is 8 and its format is the same as the HUMAN-READABLE-INFO attribute. The Text field in the USER-DISPLAY-NAME attribute contains the name of the user.

4.2.9 USER-URI

The USER-URI attribute type is 9 and its format is the same as the HUMAN-READABLE-INFO attribute. The Text field in the USER-URI attribute contains the URI of the user.

4.2.10 PRIORITY

The following is the format of the contents of the PRIORITY attribute, whose attribute type is 10.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type = 7   |M|    Length     |   Priority    |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Priority: the higher the 8-bit value, the more priority is requested for a given floor request.

Reserved: at this point, the 8 bits in the reserved field SHOULD be set to zero by the sender of the message and MUST be ignored by the receiver.

5. Message Format

This section contains the ABNF [2] of the BFCP messages.

5.1 FloorRequest

Clients request a floor by sending a FloorRequest message to the floor control server. The following is the format of the FloorRequest message:

```
FloorRequest =   (FIXED-HEADER)
                 (REQUEST-ID)
                 1*(FLOOR-ID)
```
5.2 FloorRelease

Clients release a floor by sending a FloorRelease message to the floor control server. Clients also use the FloorRelease message to cancel pending floor requests. The following is the format of the FloorRelease message:

FloorRelease = (FIXED-HEADER)
(Request-ID)
1*(FLOOR-ID)
(USER-ID)
(TOKEN)

5.3 FloorRequestStatus

The floor control server informs clients about the status of their floor requests by sending them FloorRequestStatus messages. The following is the format of the FloorRequestStatus message:

FloorRequestStatus = (FIXED-HEADER)
(Request-ID)
1*(FLOOR-ID)
(USER-ID)
(REQUEST-STATUS)

5.4 FloorStatus

The floor control server informs clients about the holder of a floor by sending them FloorStatus messages. The following is the format of the FloorStatus message:

FloorStatus = (FIXED-HEADER)
(FLOOR-ID)
*(REQUEST-ID)
(USER-ID)
5.5 ChairAction

Chairs send instructions to floor control servers by sending ChairAction messages. The following is the format of the ChairAction message:

ChairAction = (FIXED-HEADER)
(REQUEST-ID)
(USER-ID)
(TOKEN)
(FLOOR-ID)
(REQUEST-ID)
(USER-ID)
(REQUEST-STATUS)
[HUMAN-READABLE-INFO]

5.6 Ping

Clients check the liveness of servers, and servers of clients, by sending a Ping message. The following is the format of the Ping message:

Ping = (FIXED-HEADER)
(REQUEST-ID)
(USER-ID)
(TOKEN)

5.7 PingAck

Clients and servers confirm that they are alive on reception of a Ping message by sending a PingAck message. The following is the format of the PingAck message:

PingAck = (FIXED-HEADER)
(REQUEST-ID)
(USER-ID)
5.8 Error

The floor control server informs clients about errors processing requests by sending them Error messages. The following is the format of the Error message:

```
Error              =   (FIXED-HEADER)
                     (REQUEST-ID)
                     (USER-ID)
                     (ERROR-CODE)
                     (HUMAN-READABLE-INFO)
```

6. bfcp and bfcps URI Definitions

The following is the ABNF [2] for the "bfcp" and "bfcps" URIs:

```
bfcpURI           =  "bfcp://" server "/" conf-id [ uri-parameters ]
bfcpsURI          =  "bfcps://" server "/" conf-id [ uri-parameters ]
server            =  [ user "@" ] hostport
user              =  user-id [ ":" token ]
user-id           =  1*DIGIT
token             =  1*DIGIT
conf-id           =  1*DIGIT
uri-parameters    =  *( "?" uri-parameter )
uri-parameter     =  transport-param / other-param
transport-param   =  "transport="
                     ( "tcp" / other-transport )
other-transport   =  pvalue
other-param       =  pname [ "=" pvalue ]
pname             =  1*paramchar
pvalue             =  1*paramchar
paramchar         =  param-unreserved / unreserved / escaped
param-unreserved  =  "[" / "]" / "/" / ":" / ";" / "+" / "/$"
unreserved        =  alphanum / mark
alphanum           =  ALPHA / DIGIT
mark               =  ";" / ";" / "/;" / ";:" / ";=" / ";+" / ";*" / ";" / ";($" / ";")
escaped           =  "%" HEXDIG HEXDIG
```
We import several definitions from RFC 2396 [4] and RFC 2732 [6], but we make them compatible with RFC 2234 [2]:

\[
\begin{align*}
\text{hostport} & = \text{host} [ ":" \text{port} ] \\
\text{host} & = \text{hostname} / \text{IPv4address} / \text{IPv6reference} \\
\text{hostname} & = *( \text{domainlabel "." } \text{toplable} [ ":" ] \\
\text{domainlabel} & = \text{alnum} \\
& / \text{alnum} *( \text{alnum} / ":" ) \text{alnum} \\
\text{toplable} & = \text{ALPHA} / \text{ALPHA} *( \text{alnum} / ":" ) \text{alnum} \\
\text{IPv4address} & = 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT \\
\text{IPv6reference} & = "[" \text{IPv6address "} "]" \\
\text{IPv6address} & = \text{hexpart} [ ":" \text{IPv4address} ] \\
\text{hexpart} & = \text{hexseq} / \text{hexseq ":" } [ \text{hexseq} ] / ":" [ \text{hexseq} ] \\
\text{hexseq} & = \text{hex4} *( ":" \text{hex4} ) \\
\text{hex4} & = 1*4HEXDIG \\
\text{port} & = 1*DIGIT
\end{align*}
\]

The following are examples of bfcp and bfcps URIs:

```
bfcp://1234:472@server34.example.com/4321
bfcp://5678:243@server45.example.com:20000/9876?transport=tcp
bfcps://server57.example.com/5643
```

### 7. Contacting a bfcp or bfcps URI

A client contacting a bfcp or bfcps follows the rules described in the following sections.

#### 7.1 Selecting a Transport Protocol

XXX: Once we resolve Issue 3 in the tracker, we will edit this section.

If the URI specifies a transport protocol in the transport parameter, that transport protocol SHOULD be used.

Otherwise, if no transport protocol is specified, but the host part of the URI is a numeric IP address, the client SHOULD use TCP. Similarly, if no transport protocol is specified, the host part is not numeric, but an explicit port is provided, the client SHOULD use TCP.

Otherwise, if no transport protocol or port is specified, and the
host part of the URI is not a numeric IP address, the client SHOULD perform a NAPTR [9] query for the domain in the URI. The services relevant for the task of transport protocol selection are those with NAPTR service fields with values "BFCP+D2X" and "BFCPS+D2X", where X is a letter that corresponds to a transport protocol supported by the domain. This specification defines D2T for TCP.

These NAPTR records provide a mapping from a domain to the SRV [7] record for contacting a server with the specific transport protocol in the NAPTR services field. The resource record will contain an empty regular expression and a replacement value, which is the SRV record for that particular transport protocol. If the server supports multiple transport protocols, there will be multiple NAPTR records, each with a different service value.

A client resolving a BFCPS URI MUST discard any services that do not contain "BFCPS" as the protocol in the service field. On the other hand, a client resolving a BFCP URI SHOULD retain records with "BFCPS" as the protocol, if the client supports TLS.

The client MUST discard any service fields that identify a resolution service whose value is not "D2X", for values of X that indicate transport protocols supported by the client. The NAPTR processing as described in RFC 3403 [9] will result in the discovery of the most preferred transport protocol of the server that is supported by the client, as well as an SRV record for the server.

If no NAPTR records are found, the client constructs SRV queries for those transport protocols it supports, and does a query for each. Queries are done using the service identifier "_bfcp" for bfcp URIs and "_bfcps" for bfcps URIs. If the client supports TLS and wishes to use it, it SHOULD also query using the service identifier "_bfcps", even when handling floor URIs. A particular transport is supported if the query is successful. The client MAY use any transport protocol it desires which is supported by the server.

If no SRV records are found, the client SHOULD use TCP to contact the server.

7.2 Determining Port and IP Address

Once the transport protocol has been determined, the next step is to determine the IP address and port.

If the host part of the URI is a numeric IP address, the client uses that address. If the URI also contains a port, the client uses that port. If no port is specified, the client uses the default port for the particular transport protocol.
If the host part of the URI was not a numeric IP address, but a port is present in the URI, the client performs an A or AAAA record lookup of the domain name. The result will be a list of IP addresses, each of which can be contacted at the specific port from the URI and transport protocol determined previously. The client SHOULD try the first record. If an attempt should fail, the next SHOULD be tried, and if that should fail, the next SHOULD be tried, and so on.

If the host part of the URI was not a numeric IP address, and no port was present in the URI, the client performs an SRV query on the record returned from the NAPTR processing of Section 7.1, if such processing was performed. If it was not, because a transport was specified explicitly, the client performs an SRV query for that specific transport, using the service identifiers as described in Section 7.1. If the NAPTR processing was not done because no NAPTR records were found, but an SRV query for a supported transport protocol was successful, those SRV records are selected. Regardless of how the SRV records were determined, the procedures of RFC 2782 [7], as described in the section titled "Usage rules" are followed.

If no SRV records were found, the client performs an A or AAAA record lookup of the domain name. The result will be a list of IP addresses, each of which can be contacted using the transport protocol determined previously, at the default port for that transport.

7.3 Lower-Layer Security

SFTP relies on lower-layer security mechanisms to provide integrity and replay protection, and confidentiality. SFTP servers MUST support TLS [3], and clients SHOULD support TLS. Clients and servers MAY support other security mechanisms.

Servers and clients that implement TLS MUST support XXX: cypher and hash algorithm.

If a client is contacting a BFCPS URI, the client MUST use TLS towards the server. If a client is contacting a BFCP URI, and the client wishes to use TLS, the client MAY use TLS towards the server. If the client is contacting a BFCP URI, and the client does not wish to use or does not support TLS, it SHOULD use a different security mechanism which SHOULD provide integrity and replay protection, and confidentiality.

7.4 Conference ID, User ID, and Token Values

Once the client has determined the transport protocol to use, the IP address and port number of the server, and the security mechanism to use, the client SHOULD establish the transport and security
connections needed to contact the server. At this point, the client is ready to send BFCP messages to the server.

In order to do this, the client needs to fill a set of fields. In particular, the client needs to fill the Conference ID field in the FIXED-HEADER, and the User ID and Token fields in their respective TLVs.

The conf-id of the BFCP or BFCPS URI contains the integer representation of the 32-bit Conference ID to be used in the FIXED-HEADER. If the conf-id contains an integer too large to be represented using 32 bits, the client MUST only use the least significant 32 bits of its representation.

The values to be used in the USER-ID and TOKEN parameters may be obtained from the URI. If the URI does not contain these values, they are obtained by the client using a mechanism which is outside the scope of this document. Examples of such a mechanism are using the Conference Event Package [13] and using the SIP [8] Call-Info header field.

8. Client Operations

This section specifies how clients can perform different operations, such as requesting a floor, using the protocol elements described in earlier sections. Chair operations, such as instructing the floor server to grant or revoke a floor, are described in Section 9.

8.1 Requesting a Floor

A client that wishes to request one or more floors does so by sending a FloorRequest message to the floor control server. The ABNF in Section 5.1 describes the TLVs that a FloorRequest message can contain. In addition, the ABNF specifies which of these TLVs are mandatory, and which ones are optional.

The client MUST set the Conference ID in the FIXED-HEADER of the FloorRequest to the Conference ID for the conference that the client obtained previously.

The client MUST set the Request ID value in the REQUEST-ID TLV to a random number. The Request ID value is used by the client to match this floor request with the responses received from the floor control server. The Request ID value is also used by the server to match this floor request with eventual FloorRelease messages from the client cancelling the floor request. The client MUST NOT reuse the same Request ID value for new messages different than FloorRelease until the floor request is granted or denied or an Error message for this
FloorRequest is received.

If the client inserts more than one FLOOR-ID TLVs in the FloorRequest message, the server will treat all the floor requests as an atomic package. That is, the floor control server will either grant or deny all the floors in the FloorRequest message.

The USER-ID and the TOKEN TLVs in the FloorRequest message are used by the server to authenticate and authorize the request.

The USER-DISPLAY-NAME and the USER-URI TLVs in the FloorRequest message, if present, provide extra information about the user requesting the floor.

The client can request the server to handle the floor request with a certain priority using a PRIORITY TLV.

The client MAY use a HUMAN-READABLE-INFO TLV to state the reason why the floor or floors are being requested. The Text in the HUMAN-READABLE-INFO TLV is intended for human consumption.

### 8.1.1 Receiving Responses

A message from the server is considered a response to the FloorRequest message by the client if the message from the server has the same Conference ID, Request ID, and User ID as the FloorRequest message.

If the response is a FloorRequestStatus message, the client obtains information about the status of the FloorRequest in a REQUEST-STATUS TLV. If the Request Status value is Granted, the client has been granted all the floors that were requested in the FloorRequest message. If the Request Status value is Denied, the client has been denied all the floors that were requested in the FloorRequest message. The HUMAN-READABLE-INFO TLV, if present, provides extra information which the client MAY display to the user.

If the response is an Error message, the server could not process the FloorRequest message for some reason, which is described in the Error message.

### 8.2 Cancelling a Floor Request

A client that wishes to cancel a pending floor request does so by sending a FloorRelease message to the floor control server. The ABNF in Section 5.2 describes the TLVs that a FloorRelease message can contain. In addition, the ABNF specifies which of these TLVs are mandatory, and which ones are optional.
The client MUST set the Conference ID in the FIXED-HEADER of the FloorRelease to the Conference ID for the conference that the client obtained previously.

Note that the FloorRelease message is also used to release a floor or floors that were granted to the client. Using the same message for both situations helps resolve the race condition that occurs when the FloorRelease message and the FloorGrant message cross each other on the wire.

The client MUST copy the REQUEST-ID, FLOOR-ID, USER-ID, and TOKEN TLVs from the FloorRequest message that the FloorRelease message is cancelling into the FloorRelease message.

8.2.1 Receiving Responses

A message from the server is considered a response to the FloorRelease message by the client if the message from the server has the same Conference ID, Request ID, and User ID as the FloorRelease message.

If the response is a FloorRequestStatus message, the client obtains information about the status of the FloorRequest in a REQUEST-STATUS TLV. If the Request Status value is different from Cancelled, the FloorRelease message and the FloorRequestStatus crossed on the wire. The client does not need to resend the FloorRelease message, because the server will send a new FloorRequestStatus message with a Request Status value of Cancelled as soon as it receives the original FloorRelease message. The HUMAN-READABLE-INFO TLV, if present, provides extra information which the client MAY display to the user.

If the response is an Error message, the server could not process the FloorRelease message for some reason, which is described in the Error message.

8.3 Releasing Floors

A client that wishes to release one or more floors, which were granted to it before, does so by sending a FloorRelease message. The ABNF in Section 5.2 describes the TLVs that a FloorRelease message can contain. In addition, the ABNF specifies which of these TLVs are mandatory, and which ones are optional.

The client MUST set the Conference ID in the FIXED-HEADER of the FloorRelease to the Conference ID for the conference that the client obtained previously.

The client MUST copy the REQUEST-ID, FLOOR-ID, USER-ID, and TOKEN
TLVs from the FloorRequest message that resulted in the floor being granted into the FloorRelease message.

### 8.3.1 Receiving Responses

A message from the server is considered a response to the FloorRelease message by the client if the message from the server has the same Conference ID, Request ID, and User ID as the FloorRelease message.

If the response is a FloorRequestStatus message, the client obtains information about the status of the FloorRequest in a REQUEST-STATUS TLV. The server will send a FloorRequestStatus with a Request Status value of Released as soon as it receives the FloorRelease message. The HUMAN-READABLE-INFO TLV, if present, provides extra information which the client MAY display to the user.

If the response is an Error message, the server could not process the FloorRelease message for some reason, which is described in the Error message.

### 8.4 Checking the Liveness of a Server

A client that wishes to check the liveness of a server does so by sending a Ping message to the floor control server. The ABNF in Section 5.6 describes the TLVs that a Ping message can contain. In addition, the ABNF specifies which of these TLVs are mandatory, and which ones are optional.

The client MUST set the Conference ID in the FIXED-HEADER of the Ping to the Conference ID for the conference that the client obtained previously.

The client MUST set the Request ID value of the REQUEST-ID TLV to a random number. The Request ID value is used by the client to match this ping request with the response received from the floor control server. The client MUST NOT reuse the same Request ID value for new messages until the client receives a PingAck or an Error message for this request.

The server uses the values of the USER-ID and TOKEN TLVs in the Ping message to authenticate and authorize the message.

### 8.4.1 Receiving Responses

A message from the server is considered a response to the Ping message by the client if the message from the server has the same Conference ID, Request ID, and User ID as the Ping message.
If the response is a PingAck message, the server is alive and the User ID and Token values sent by the client were acceptable.

If the response is an Error message, the server could not process the Ping message for some reason, which is described in the Error message.

8.5 Responding to a Ping Message

The processing of a Ping message by a client involves generating a PingAck message. The PingAck message is constructed by copying the Conference ID, Request ID, and User ID values from the Ping message into the PingAck message. The PingAck message MUST also contain a TOKEN TLV.

9. Chair Operations

This section specifies how clients acting as chairs can perform different operations, such as instructing the floor server to grant or revoke a floor, using the protocol elements described in earlier sections.

9.1 Obtaining Information about Floor Requests

A chair can obtain information about the floor requests that the floor control server receives in different ways, which include using out-of-band mechanisms. Chairs using BFCP to obtain such information receive it in FloorStatus messages from the floor control server.

9.2 Instructing the Floor Control Server

Chairs that wish to send instructions to a floor control server do so by sending a ChairAction message. The ABNF in Section 5.5 describes the TLVs that a ChairAction message can contain. In addition, the ABNF specifies which of these TLVs are mandatory, and which ones are optional.

The client MUST set the Conference ID in the FIXED-HEADER of the ChairAction to the Conference ID for the conference that the client obtained previously.

The client MUST set the Request ID value of the REQUEST-ID TLV to a random number. The Request ID value is used by the chair to match this ChairAction message with the responses received from the floor control server. The chair MUST NOT reuse the same Request ID value for new messages until the floor server has performed the instructions sent in the ChairAction message or until an Error response is received.
The server uses the values of the USER-ID and TOKEN TLVs to authenticate and authorize the request.

The chair MUST identify the floor the instructions in the message apply to using a FLOOR-ID TLV. Additionally, the chair MUST identify the floor request the instructions in the message apply to using a REQUEST-ID and a USER-ID TLVs.

The chair MUST provide the new status of the floor request using a REQUEST-STATUS TLV. If the new status of the floor request is Accepted, the chair MAY use the Queue Position field to provide a queue position for the floor request. If the chair does not wish to provide a queue position, all the bits of the Queue Position field SHOULD be set to zero. The chair SHOULD use the Status Revoked to revoke a floor that was granted (i.e., Granted status) and to reject floor requests in any other status (e.g., Pending and Accepted).

The chair MAY use a HUMAN-READABLE-INFO TLV to state the reason why the floor or floors are being accepted, granted, or revoked. The Text in the HUMAN-READABLE-INFO TLV is intended for human consumption.

10. Server Operations

This section specifies how servers can perform different operations, such as granting a floor, using the protocol elements described in earlier sections. Section 10.6

On reception of a message from a client, the floor control server MUST check whether or not the values of the Conference ID, User ID, and Token identify a valid user. If they do not, the floor server SHOULD send an Error message, as described in Section 10.6, with Error code 1 (Authentication Failed).

On reception of a message from a client, the floor control server MUST check whether or not it understands all the mandatory ( 'M' bit set) TLVs in the message. If the server does not understand all of them, the floor server SHOULD send an Error message, as described in Section 10.6, with Error code 2 (Authentication Failed). The Error message SHOULD list the TLVs that were not understood.

10.1 Reception of a FloorRequest Message

The processing of a FloorRequest message by a server involves generating one or more FloorRequestStatus messages. The floor control server SHOULD generate a FloorRequestStatus message as soon as possible. If the floor server cannot accept, grant or deny the floor request right away, it SHOULD use a Request Status value of Pending.
At any time, when the status of the floor request changes, the floor control server SHOULD send a FloorRequest message with the appropriate Request Status.

On reception of a FloorRelease message, the server SHOULD send a FloorRequestStatus message with a Request Status value of Released, if the floor had been previously granted, or of Cancelled, if it had not.

The server constructs the FloorRequestStatus messages by copying the Conference ID, Request ID, and User ID values from the FloorRequest message into the response message. The server MAY add a HUMAN-READABLE-INFO TLV to provide extra information to the user about its decision.

The floor control server MAY use the PRIORITY TLV, if present in the FloorRequest, to give higher or lower priority to the floor request.

### 10.2 Checking the Liveness of a Client or Chair

A server that wishes to check the liveness of a client does so by sending a Ping message to the client or chair. Such a Ping message makes the client return a PingAck message, which contains the client's or chair's User ID and Token. So, the server can use Ping messages to re-authenticate the client (e.g., a new token is sent to the client or chair out-of-band and the Ping message makes the client show the server that the client or chair received it).

The ABNF in Section 5.6 describes the TLVs that a Ping message can contain. In addition, the ABNF specifies which of these TLVs are mandatory, and which ones are optional.

The server MUST set the Conference ID in the FIXED-HEADER of the Ping to the Conference ID for the conference that the client or chair is part of.

The server MUST set the Request ID value of the REQUEST-ID TLV to a random number. The Request ID value is used by the server to match this ping request with the response received from the client or chair. The server MUST NOT reuse the same Request ID value for new messages until the server receives a PingAck message for this request.

The server MUST set the User ID value of the USER-ID TLV to the User ID of the destination client or chair.
10.2.1 Receiving a PingAck Message

A PingAck message from the client is considered a response to the Ping message by the server if the PingAck message has the same Conference ID, Request ID, and User ID as the Ping message.

The PingAck message sent by the client in response to the Ping message sent by the server will contain the client's User ID and Token. The server authenticates this message as any other message, which may involve returning an Error message if the authentication is not successful.

10.3 Responding to a Ping Message

The processing of a Ping message by a server involves generating a PingAck message. The PingAck message is constructed by copying the Conference ID, Request ID, and User ID values from the Ping message into the PingAck message.

10.4 Informing about the Status of a Floor

Floor control servers send information to clients and chairs about the status of a floor by sending FloorStatus messages.

The server MUST set the Conference ID in the FIXED-HEADER of the FloorStatus to the Conference ID for the conference that the client or chair is part of.

The chair MUST identify the floor the FloorStatus message applies to using a FLOOR-ID TLV.

The server MAY provide information about a number of floor requests in a single FloorStatus message. For each floor request, the server MUST include in the FloorStatus the request's REQUEST-ID, USER-ID, and REQUEST-STATUS, and MAY include its USER-DISPLAY-NAME, USER-URI, HUMAN-READABLE-INFO, and requested PRIORITY.

10.5 Receiving a ChairAction Message

On reception of a ChairAction message, the floor control server checks whether the chair that send the message is authorized to perform the operation being requested. If the chair is authorized, the floor control server performs the operation. If the new Status is Accepted and all the bits of the Queue Position field are zero, the server assigns a queue position (e.g., the last in the queue) to the floor request based on local policy (in case the server implements a queue).
If the floor control server cannot find a floor request that matches the request the operation in the ChairAction message applies to, the floor control server generates an Error message, as described in Section 10.6, with an Error code with a value of 3 (Floor Request does not Exist).

If the chair is not authorized to perform the operation being requested, the floor control server generates an Error message, as described in Section 10.6, with an Error code with a value of 4 (Unauthorized Operation).

10.6 Error Message Generation

Error messages are always sent in response to a previous message from the client. Servers construct Error messages by copying the Conference ID, Request ID, and User ID values from the client's message into the Error message.

The ABNF in Section 5.8 describes the TLVs that a Error message can contain. In addition, the ABNF specifies which of these TLVs are mandatory, and which ones are optional.

11. Security Considerations

TBD.

12. IANA Considerations

TBD

12.1 bfcp and bfcps URI Schemes Registration

This section registers the bfcp and bfcps URI schemes following the template given in RFC 2717 [5].

URI scheme names: "bfcp" and "bfcps"

URI scheme syntax: specified in Section 6 of RFC XXXX (substitute XXXX with the number of this RFC).

Character encoding considerations: The "bfcp" and "bfcps" URIs support the UTF-8 encoding scheme.

Intended use: common within floor control protocols.

Applications and/or protocols which use these URI scheme: BFCP, which is specified in this document.
Interoperability considerations: none known.

Security considerations: the "bfcps" URI scheme indicates a requirement to use TLS to contact the floor control server.

Relevant publications: RFC XXXX (substitute XXXX with the number of this RFC).

Contact information: the IETF XCON Working group. In case the WG does no exist anymore, any person appointed by the IETF Transport Area Director(s).

Registered-by: Gonzalo Camarillo <Gonzalo.Camarillo@ericsson.com>

Change control: extensions, new parameters and new values to these URIs have to be documented in an RFC. The IETF XCON Working Group or, in case the WG does no exist anymore, any person appointed by the IETF Transport Area Director(s), will provide expert review and advise to the change control process.

13. Acknowledgments

The XCON WG chairs, Adam Roach and Alan Johnston, provided useful ideas for this document.

14. References

14.1 Normative References


14.2 Informational References


Authors' Addresses

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com
Intellectual Property Statement

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 IETF's procedures with respect to rights in IETF 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.

Disclaimer of Validity

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 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.

Copyright Statement

Copyright (C) The Internet Society (2004). 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.

Acknowledgment

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