XCON Working Group
Internet-Draft
Expires: November 5, 2005

G. Camarillo
Ericsson
J. Ott
Helsinki University of Technology
K. Drage
Lucent Technologies

May 4, 2005

The Binary Floor Control Protocol (BFCP)
draft-ietf-xcon-bfcp-04.txt

Status of this Memo

Copyright Notice

Abstract

Floor control is a means to manage joint or exclusive access to shared resources in a (multiparty) conferencing environment. Thereby, floor control complements other functions -- such as conference and media session setup, conference policy manipulation, and media control -- that are realized by other protocols.

This document specifies the Binary Floor Control Protocol (BFCP). BFCP is used between floor participants and floor control servers, and between floor chairs (i.e., moderators) and floor control servers.

Table of Contents

1. Introduction

Within a conference, some applications 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 coordinated (shared or exclusive) access to these resources.

The Requirements for Floor Control Protocol [10] 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.

In addition, BFCP has been designed so that it can be used in low-bandwidth environments. The binary encoding used by BFCP achieves a small message size (when message signatures are not used) that keeps the time it takes to transmit delay-sensitive BFCP messages at minimum. Delay-sensitive BFCP messages include FloorRequest, FloorRelease, FloorRequestInfo, and ChairAction. It is expected that future extensions to these messages do not increase the size of these messages in a significant way.

The remainder of this document is organized as follows: Section 2 defines the terminology used throughout this document, Section 3 discusses the scope of BFCP (i.e., which tasks fall within the scope of BFCP and which ones are performed using different mechanisms), Section 4 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 [2] and indicate requirement levels for compliant implementations.

Media Participant: An entity that has access to the media resources of a conference (e.g., it can receive a media stream). In floor-controlled conferences, a given media participant is typically co-located with a floor participant, but does not need to. Third-party floor requests consist of having a floor participant request a floor for a media participant when they are not colocated. The protocol between a floor participant and a media participant (that are not colocated) is outside the scope of this document.

Client: a floor participant or a floor chair that communicate with a floor control server using BFCP.

Floor: A permission to temporarily access or manipulate a specific shared resource or set of resources.

Floor Chair: A logical entity that manages one floor (grants, denies, or revokes a floor). An entity that assumes the logical role of a floor chair for a given transaction may assume a different role (e.g., floor participant) for a different transaction. The roles of floor chair and floor participant are defined on a transaction-by-transaction basis. BFCP transactions are defined in Section 8.

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. The floor control server of a conference may perform other logical roles (e.g., floor participant) in another conference.

Floor Participant: A logical entity that requests floors, and possibly information about them, from a floor control server. An entity that assumes the logical role of a floor participant for a given transaction may assume a different role (e.g., a floor chair) for a different transaction. The roles of floor participant and floor chair are defined on a transaction-by-transaction basis. BFCP transactions are defined in Section 8. In floor-controlled conferences, a given floor participant is typically co-located with a media participant, but does not need to. Third-party floor requests consist of having a floor participant request a floor for a media participant when they are not co-located.

Participant: An entity that acts as a floor participant, as a media participant, or as both.

3. Scope

As stated earlier, BFCP is a protocol to coordinate access to shared resources in a conference following the requirements defined in [10]. Floor control complements other functions defined in the XCON conferencing framework [12] and is compatible with the SIPPING conferencing framework [11]. The floor control protocol BFCP defined in this document only specifies a means to arbitrate access to floors. The rules and constraints for floor arbitration and the results of floor assignments are outside the scope of this document and defined by other protocols [12].

Figure 1 shows the tasks that BFCP can perform.

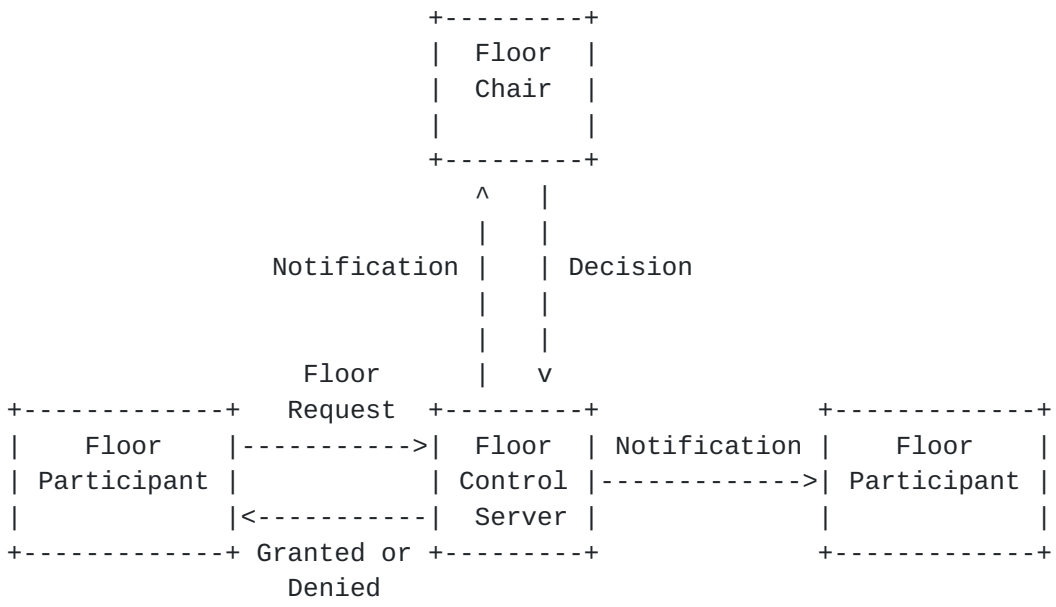


Figure 1: Functionality provided by BFCP

BFCP provides a means:

- for floor participants to send floor requests to floor control servers.
- for floor control servers to grant or deny requests to access a given resource from floor participants.
- for floor chairs to send floor control servers decisions regarding floor requests.
- for floor control servers to keep floor participants and floor chairs informed about the status of a given floor or a given floor request.

Even though tasks that do not belong to the previous list are outside the scope of BFCP, some of these out-of-scope tasks relate to floor control and are essential to create floors and to establish BFCP connections between different entities. In the following subsections, we discuss some of these tasks and mechanisms to perform them.

3.1 Floor Creation

The association of a given floor with a resource or a set of resources (e.g., media streams) is out of the scope of BFCP as described in [12]. Floor creation and termination are also outside the scope of BFCP; these aspects are handled using the conference

control protocol for manipulating the conference object. Consequently, the floor control server needs to stay up to date on changes to the conference object (e.g., when a new floor is created).

3.2 Obtaining Information to Contact a Floor Control Server

A client needs a set of data in order to establish a BFCP connection to a floor control server. These data include the transport address of the server, the conference identifier, and the user identifier.

Clients can obtain this information in different ways. One is to use an offer/answer [9] exchange, which is described in [13]. Other mechanisms are also described in the XCON framework (and other related documents).

3.3 Generating a Shared Secret

Authentication in BFCP is based on a shared secret between the client and the floor control server. So, there is a need for a mechanism to generate such a shared secret. However, such mechanism is outside the scope of BFCP.

Shared secrets can also be generated and exchanged using out-of-band means. For example, when the floor participant or the floor chair obtains the information needed to contact the BFCP floor control server over a secure channel (e.g., an offer/answer [9] exchange using SIP [8] protected using S/MIME), they can get the shared secret using the same channel.

3.4 Obtaining Floor-Resource Associations

Floors are associated with resources. For example, a floor that controls who talks at a given time has a particular audio stream as its associated resource. Associations between floors and resources are part of the conference object.

Floor participants and floor chairs need to know which resources are associated with which floors. They can obtain this information using different mechanisms, such as an offer/answer [9] exchange. How to use an offer/answer exchange to obtain these associations is described in [13].

> Note that floor participants perform offer/answer exchanges with the SIP Focus of the conference. So, the SIP Focus needs to obtain information about associations between floors and resources in order to be able to provide this information to a floor participant in an offer/answer exchange.

Other mechanisms for obtaining this information, including discussion of how the information is made available to a (SIP) Focus, are described in the XCON framework (and other related documents).

3.5 Privileges of Floor Control

A participant whose floor request is granted has the right to use (in a certain way) the resource or resources associated with the floor that was requested. For example, the participant may have the right to send media over a particular audio stream.

Nevertheless, holding a floor does not imply that others will not be able to use its associated resources at the same time, even if they do not have the right to do so. Determination of which media participants can actually use the resources in the conference is discussed in the XCON Framework.

4. Overview of Operation

This section provides a non-normative description of BFCP operations. Section 4.1 describes the interface between floor participants and floor control servers and Section 4.2 describes the interface between floor chairs and floor control servers

BFCP messages, which use a TLV (Type-Length-Value) binary encoding, consist of a common header followed by a set of attributes. The common header contains, among other information, a 32-bit conference identifier. Floor participants, media participants, and floor chairs are identified by a 16-bit user identifier, which is carried in an attribute.

Participant authentication in BFCP is based on shared secrets. The floor control server of a conference shares a secret with each of the participants in the conference and can request them to sign their messages using that shared secret.

BFCP supports nested attributes (i.e., attributes that contain attributes). These are referred to as grouped attributes.

There are two types of transactions in BFCP: client-initiated transactions and server-initiated transactions. Client-initiated transactions consist of a message from a client to the floor control server and a response from the floor control server to the client. Both messages can be related because they carry the same TRANSACTION-ID attribute. Server-initiated transactions consist of a single message, which has no TRANSACTION-ID attribute, from the floor control server to a client.

4.1 Floor Participant to Floor Control Server Interface

Floor participants request a floor by sending a FloorRequest message to the floor control server. BFCP supports third-party floor requests. That is, the floor participant sending the floor request need not be co-located with the media participant that will get the floor once the floor request is granted. FloorRequest messages carry the identity of the requester in a USER-ID attribute, and the identity of the beneficiary of the floor, in third party floor requests, in a BENEFICIARY-ID attribute.

> Third party floor requests can be sent, for example, by floor participants that have a BFCP connection to the floor control server but that are not media participants (i.e., they do not handle any media).

FloorRequest messages identify the floor or floors being requested by carrying their 16-bit floor identifiers in FLOOR-ID attributes. If a FloorRequest message carries more than one floor identifier, the floor control server treats all the floor requests as an atomic package. That is, the floor control server either grants or denies all the floors in the FloorRequest message.

Floor control servers respond to FloorRequest messages with FloorRequestInfo messages, which provide information about the status of the floor request. The first FloorRequestInfo message is the response to the FloorRequest message from the client, and therefore carries the same TRANSACTION-ID attribute as the FloorRequest.

Additionally, the first FloorRequestInfo message carries the Floor Request ID in a FLOOR-REQUEST-INFORMATION attribute. Subsequent FloorRequestInfo messages related to the same floor request will carry the same Floor Request ID. This way, the floor participant can associate them with the appropriate floor request.

Messages from the floor participant related to a particular floor request also use the same Floor Request ID as the first FloorRequestInfo Message from the floor control server.

Figure 2 shows how a floor participant requests a floor, obtains it, and, at a later time, releases it. This figure illustrates the use, among other attributes, of the TRANSACTION-ID and the FLOOR-REQUEST-ID attributes.

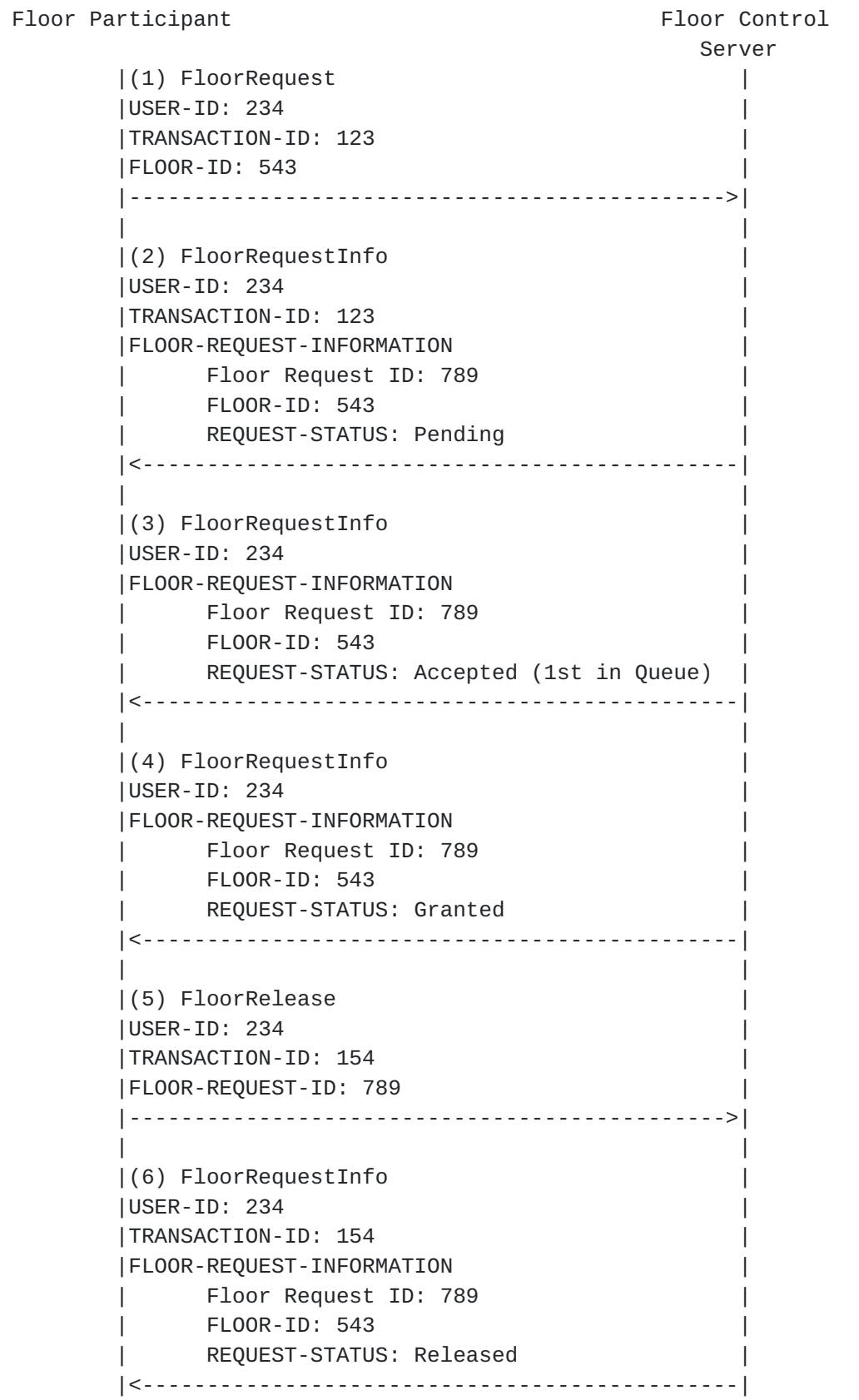
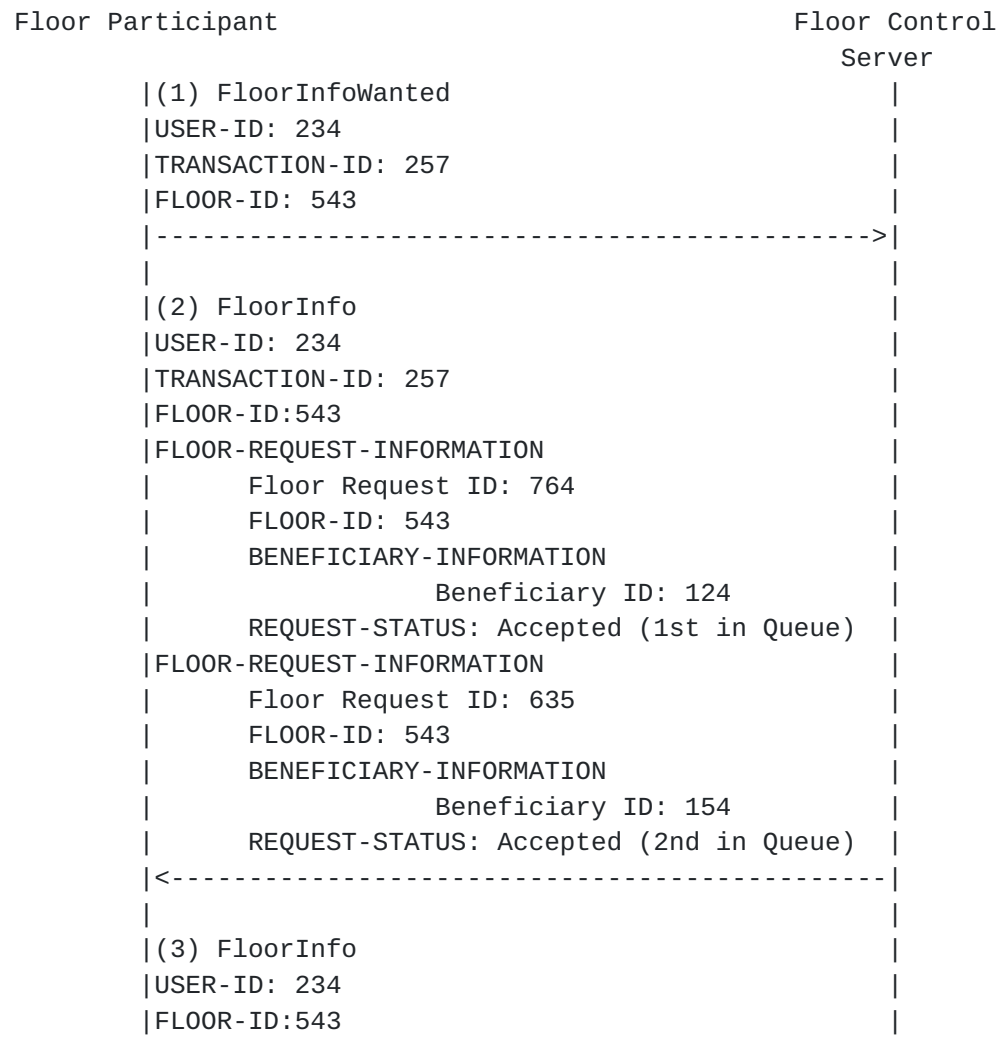

Floor Participant
Floor Control Server
(1) FloorRequest
USER-ID: 234
TRANSACTION-ID: 123
FLOOR-ID: 543
(2) FloorRequestInfo
USER-ID: 234
TRANSACTION-ID: 123
FLOOR-REQUEST-INFORMATION
Floor Request ID: 789
FLOOR-ID: 543
REQUEST-STATUS: Pending
(3) FloorRequestInfo
USER-ID: 234
FLOOR-REQUEST-INFORMATION
Floor Request ID: 789
FLOOR-ID: 543
REQUEST-STATUS: Accepted (1st in Queue)
(4) FloorRequestInfo
USER-ID: 234
FLOOR-REQUEST-INFORMATION
Floor Request ID: 789
FLOOR-ID: 543
REQUEST-STATUS: Granted
(5) FloorRelease
USER-ID: 234
TRANSACTION-ID: 154
FLOOR-REQUEST-ID: 789
(6) FloorRequestInfo
USER-ID: 234
TRANSACTION-ID: 154
FLOOR-REQUEST-INFORMATION
Floor Request ID: 789
FLOOR-ID: 543
REQUEST-STATUS: Released

Figure 2: Requesting and releasing a floor

Figure 3 shows how a floor participant requests to be informed on the status of a floor. The first FloorInfo message from the floor control server is the response to the FloorInfoWanted message, and as such, carries the same TRANSACTION-ID attribute as the FloorInfoWanted message.

Subsequent FloorInfo messages consist of server-initiated transactions, and therefore carry no TRANSACTION-ID attribute. FloorInfo message (2) indicates that there are currently two floor requests for the floor whose Floor ID is 543. FloorInfo message (3) indicates that the floor requests with Floor Request ID 764 has been granted, while the floor request with Floor Request ID 635 is the first in the queue. FloorInfo message (4) indicates that the floor request with Floor Request ID 635 has been granted.



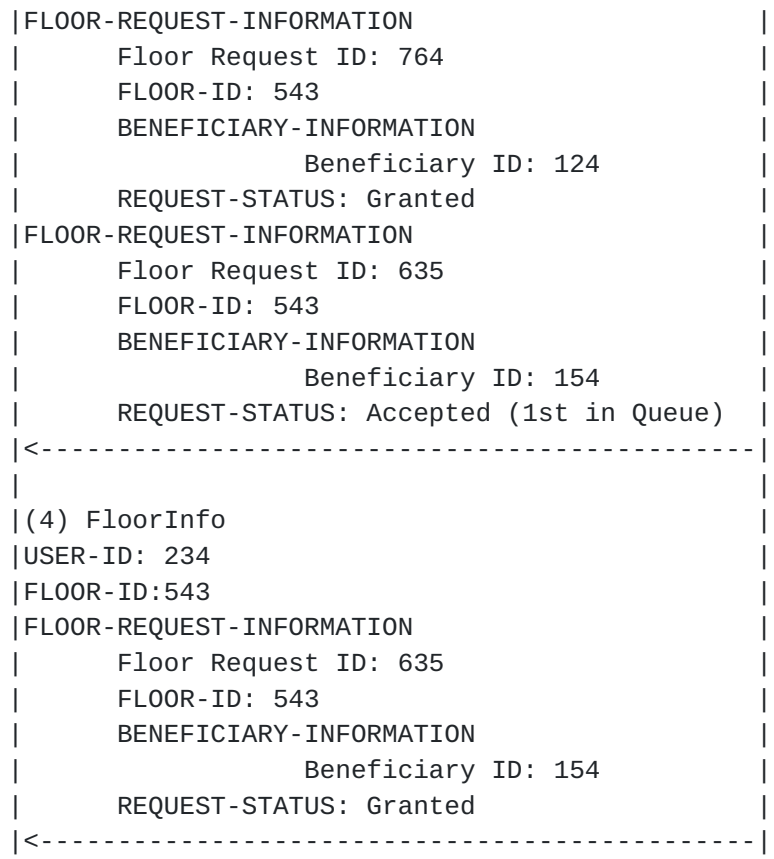


Figure 3: Obtaining status information about a floor

FloorInfo messages contain information about the floor requests they carry. For example, FloorInfo message (4) indicates that the floor request with Floor Request ID 635 has as the beneficiary (i.e., the participant that holds the floor when a particular floor request is granted) the participant whose User ID is 154. The floor request applies only to the floor whose Floor ID is 543. That is, this is not a multi-floor floor request.

A multi-floor floor request applies to more than one floor (e.g., a participant wants to be able to speak and write on the whiteboard at the same time). The floor control server treats a multi-floor floor request as an atomic package. That is, the floor control server either grants the request for all floors or denies the request for all the floors.

4.2 Floor Chair to Floor Control Server Interface

Figure 4 shows a floor chair instructing a floor control server to grant a floor. Note, however, that although the floor control server needs to take into consideration the instructions received in

ChairAction messages (e.g., granting a floor), it does not necessarily need to perform them exactly as requested by the floor chair. The operation that the floor control server performs depends on the ChairAction message and on the internal state of the floor control server.

For example, a floor chair may send a ChairAction message granting a floor which was requested as part of an atomic floor request operation that involved several floors. Even if the chair responsible for one of the floors instructs the floor control server to grant the floor, the floor control server will not grant it until the chairs responsible for the other floors agree to grant them as well. In another example, a floor chair may instruct the floor control server to grant a floor to a participant. The floor control server needs to revoke the floor from its current holder before granting it to the new participant.

So, the floor control server is ultimately responsible to keep a coherent floor state using instructions from floor chairs as input to this state.

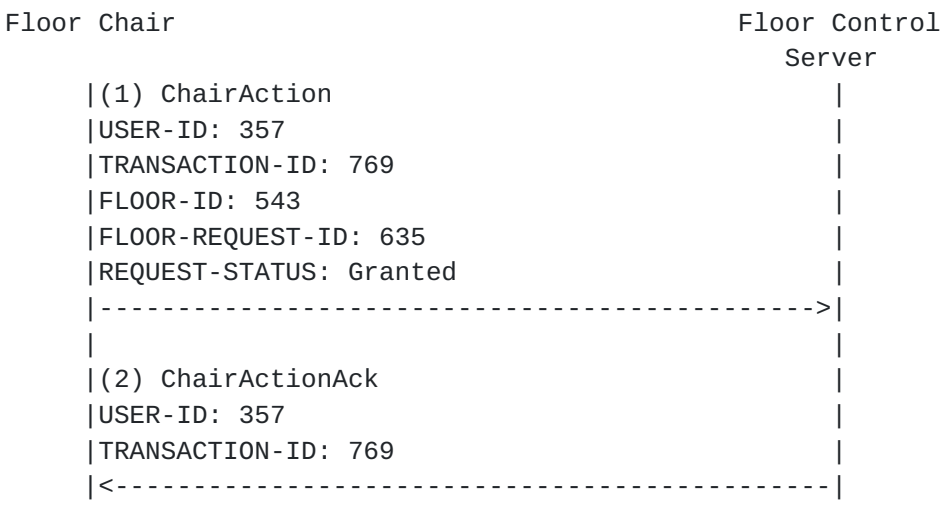


Figure 4: Chair instructing the floor control server

5. 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.

5.1 FIXED-HEADER Format

The following is the FIXED-HEADER format.

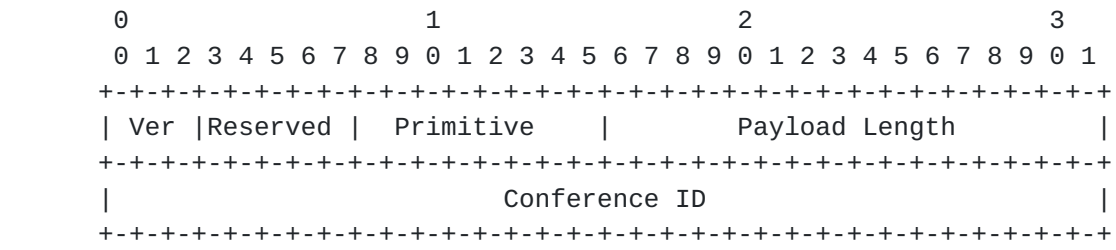


Figure 5: FIXED-HEADER format

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:

Value	Primitive	Direction
1	FloorRequest	P -> S
2	FloorRelease	P -> S
3	FloorRequestInfoWanted	P -> S ; Ch -> S
4	FloorRequestInfo	P <- S ; Ch <- S
5	UserInfoWanted	P -> S ; Ch -> S
6	UserInfo	P <- S ; Ch <- S
7	FloorInfoWanted	P -> S ; Ch -> S
8	FloorInfo	P <- S ; Ch <- S
9	ChairAction	Ch -> S
10	ChairActionAck	Ch <- S
11	Hello	P -> S ; Ch -> S
12	HelloAck	P <- S ; Ch <- S
13	Error	P <- S ; Ch <- S

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

Table 1: BFCP primitives

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

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

5.2 Attribute Format

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

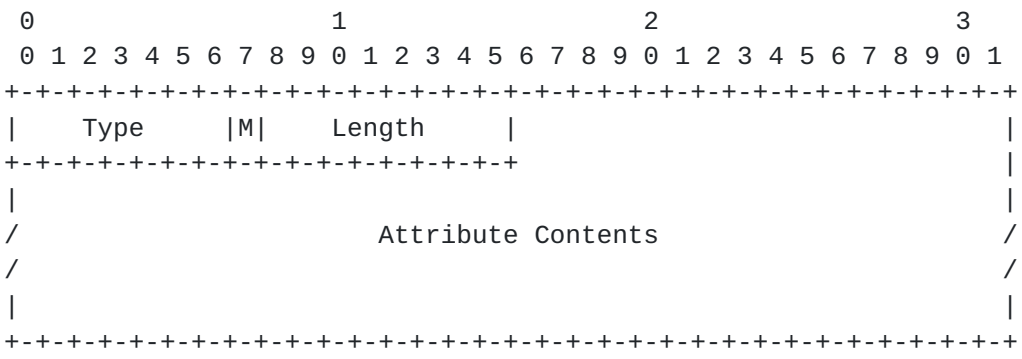


Figure 6: Attribute format

Type: this 7-bit field contains the type of the attribute. Each attribute, identified by its type, has a particular format. The attribute formats defined are:

Unsigned16: the contents of the attribute consist of a 16-bit unsigned integer.

OctetString16: the contents of the attribute consist of 16 bits of arbitrary data.

OctetString: the contents of the attribute consist of arbitrary data of variable length.

Grouped: the contents of the attribute consist of a sequence of attributes.

Note that extension attributes defined in the future may define new attribute formats.

The following attribute types are defined:

Type	Attribute	Format
1	BENEFICIARY-ID	Unsigned16
2	FLOOR-ID	Unsigned16
3	FLOOR-REQUEST-ID	Unsigned16
4	NONCE	Unsigned16
5	TRANSACTION-ID	Unsigned16
6	USER-ID	Unsigned16
7	PRIORITY	OctetString16
8	REQUEST-STATUS	OctetString16
9	DIGEST	OctetString
10	ERROR-CODE	OctetString
11	ERROR-INFO	OctetString
12	PARTICIPANT-PROVIDED-INFO	OctetString
13	STATUS-INFO	OctetString
14	SUPPORTED-ATTRIBUTES	OctetString
15	SUPPORTED-PRIMITIVES	OctetString
16	USER-DISPLAY-NAME	OctetString
17	USER-URI	OctetString
18	BENEFICIARY-INFORMATION	Grouped
19	FLOOR-REQUEST-INFORMATION	Grouped
20	REQUESTED-BY-INFORMATION	Grouped

Table 2: BFCP attributes

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. The Length in grouped attributes is the length of the grouped attribute itself (including Type, 'M' bit, and Length fields) plus the total length (including padding) of all the included attributes.

Attribute Contents: the contents of the different attributes are defined in the following sections.

5.2.1 BENEFICIARY-ID

The following is the format of the BENEFICIARY-ID attribute.

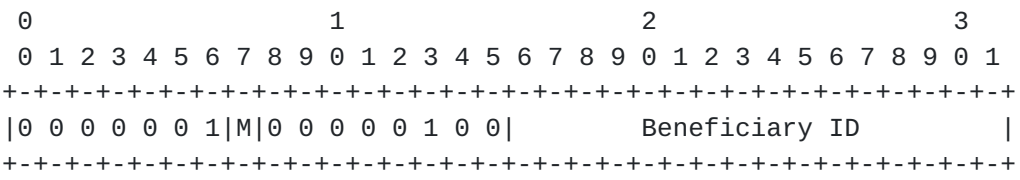


Figure 7: BENEFICIARY-ID format

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

Note that although the formats of the BENEFICIARY-ID attribute and of the USER-ID attribute (see Figure 12) are similar, their semantics are different. The BENEFICIARY-ID attribute is used in third-party floor requests and to request information about a particular participant.

5.2.2 FLOOR-ID

The following is the format of the FLOOR-ID attribute.

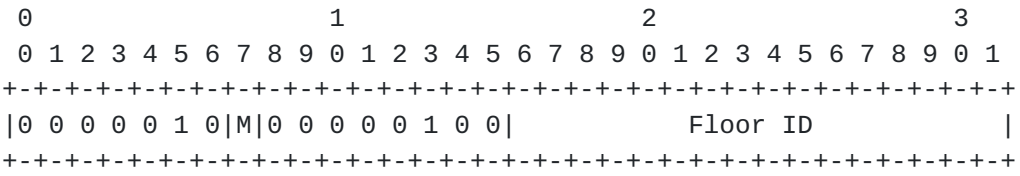


Figure 8: FLOOR-ID format

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

5.2.3 FLOOR-REQUEST-ID

The following is the format of the FLOOR-REQUEST-ID attribute.

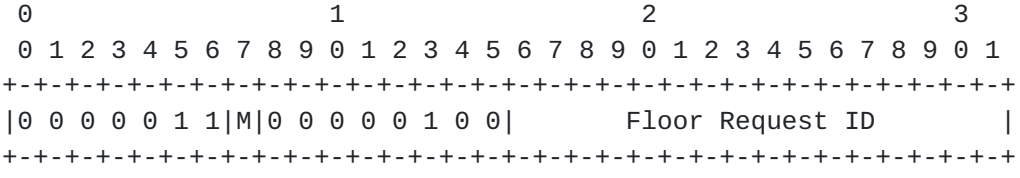


Figure 9: FLOOR-REQUEST-ID format

Floor Request ID: this field contains a 16-bit value that indentifies a floor request at the floor control server.

5.2.4 NONCE

The following is the format of the NONCE attribute.

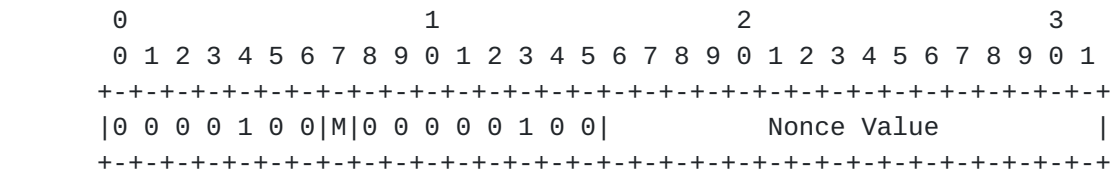


Figure 10: NONCE format

Nonce Value: this 16-bit field contains a nonce.

5.2.5 TRANSACTION-ID

The following is the format of the TRANSACTION-ID attribute.

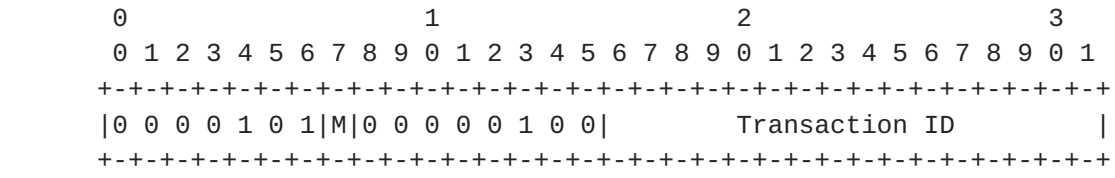


Figure 11: TRANSACTION-ID format

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

5.2.6 USER-ID

The following is the format of the USER-ID attribute.

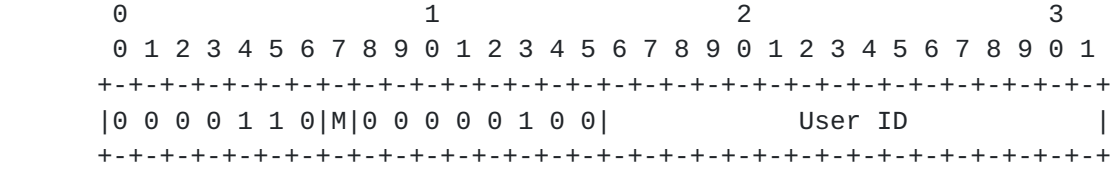


Figure 12: USER-ID format

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

The identify used by a participant in BFCP, which is carried in the

User ID field, is generally mapped to the identity used by the same participant in the session establishment protocol (e.g., in SIP). The way this mapping is performed is outside the scope of this specification.

5.2.7 PRIORITY

The following is the format of the PRIORITY attribute.

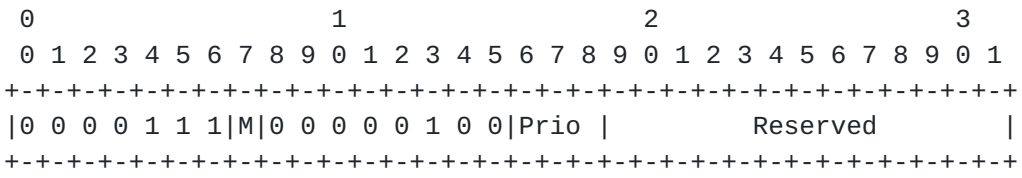


Figure 13: PRIORITY format

Prio: this field contains a 3-bit priority value as shown in Table 3. Senders SHOULD NOT use values higher than 4 in this field. Receivers MUST treat values higher than 4 as if the value received had been 4 (Highest). The default priority value when the PRIORITY attribute is missing is 2 (Normal).

Value	Priority
0	Lowest
1	Low
2	Normal
3	High
4	Highest

Table 3: Priority values

Reserved: at this point, the 13 bits in the reserved field SHOULD be set to zero by the sender of the message and MUST be ignored by the receiver.

5.2.8 REQUEST-STATUS

The following is the format of the REQUEST-STATUS attribute.

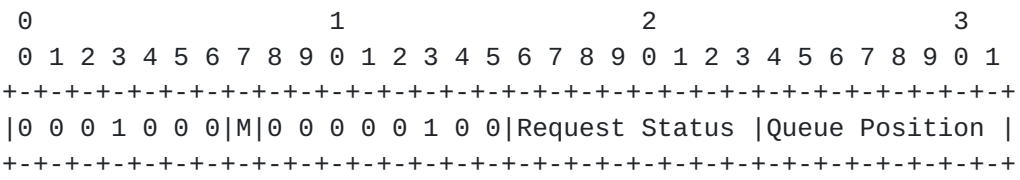


Figure 14: REQUEST-STATUS format

Request Status: this 8-bit field contains the status of the request, as described in the following table.

Value	Status
1	Pending
2	Accepted
3	Granted
4	Denied
5	Cancelled
6	Released
7	Revoked

Table 4: Request Status values

Queue Position: this 8-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.

5.2.9 DIGEST

The following is the format of the DIGEST attribute.

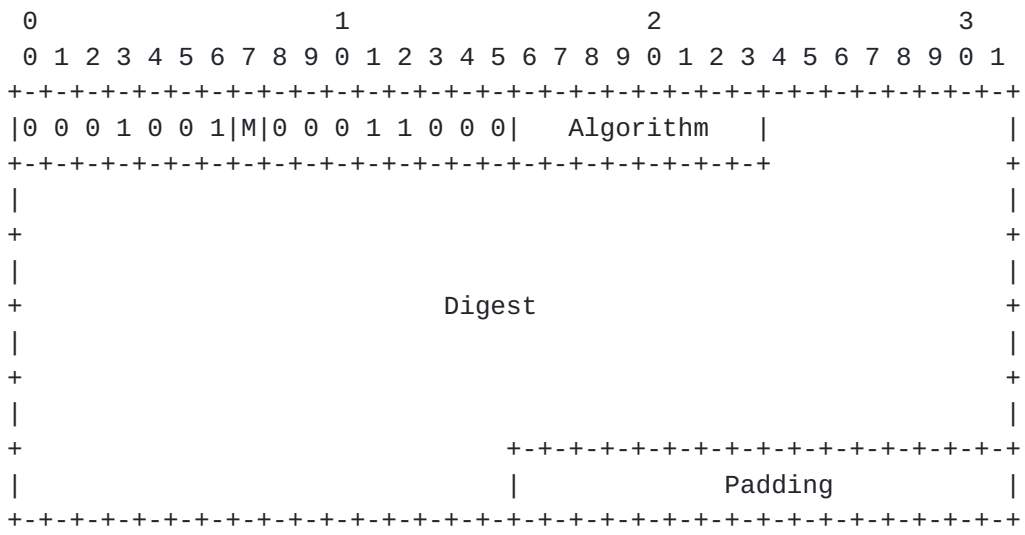


Figure 15: DIGEST format

Algorithm: this 8-bit field contains the identifier of the algorithm used to calculate the keyed digest. The following are the algorithm identifiers defined:

Identifier	Algorithm	Digest Length	Reference
0	HMAC-SHA1	20 bytes	RFC 2104 [1]

Table 5: Digest algorithms

The text used as input to the digest algorithm is the BFCP message, including the FIXED-HEADER, up to and including the attribute preceding the DIGEST attribute. Depending on the algorithm, this text may need to be padded with zeroes. When HMAC-SHA1 is used, the input text needs to be padded so as to be a multiple of 64 bytes.
The key used as input to the keyed digest is the secret shared between the server and the user identified by the USER-ID attribute in the message.

Digest: this field contains a keyed digest of the BFCP message. Its calculation is described in Section 9.

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

5.2.10 ERROR-CODE

The following is the format of the ERROR-CODE attribute.

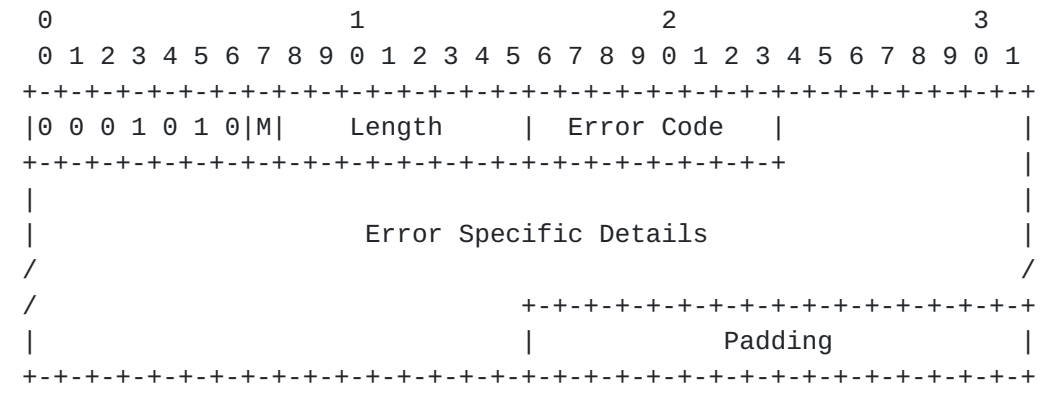


Figure 16: ERROR-CODE format

Error Code: this 8-bit field contains an error code from the following table.

Value	Meaning
1	Conference does not Exist
2	User does not Exist
3	DIGEST Attribute Required
4	Invalid Nonce
5	Authentication Failed
6	Unknown Primitive
7	Unknown Mandatory Attribute
8	Unauthorized Operation
9	Invalid Floor ID
10	Floor Request ID Does Not Exist
11	You have Already Reached the Maximum Number of Ongoing Floor Requests for this Floor

Table 6: Error Code meaning

Error Specific Details: Present only for certain Error Codes. In this document, only for Error Code 3 (DIGEST Attribute Needed) and Error Code 7 (Unknown Mandatory Attribute). See Section 5.2.10.1 and Section 5.2.10.2 for their respective definitions.

Padding: one, two, or three bytes of padding added so that the

contents of the ERROR-CODE attribute is 32-bit aligned. If the attribute 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.

5.2.10.1 Error Specific Details for Error Code 3

The following is the format of the Error Specific Details field for Error Code 3.

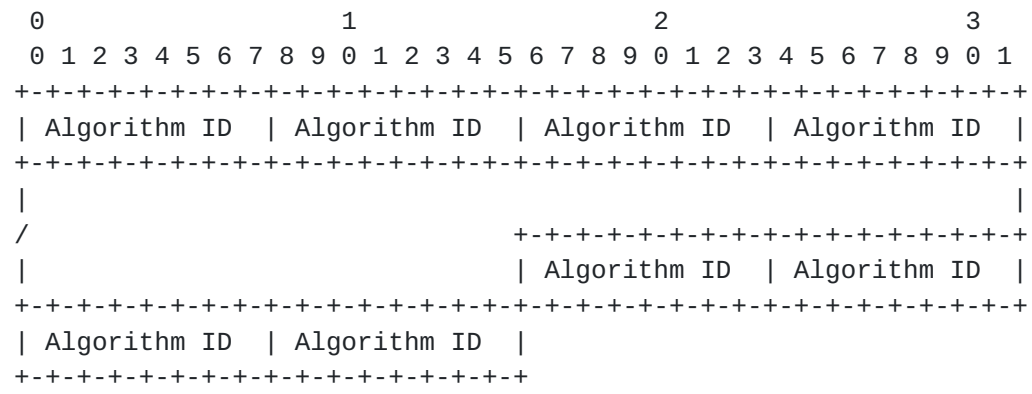


Figure 17: Digest algorithms format

Algorithm ID: these 8-bit fields contain the identifiers of the digest algorithms supported by the floor control server in order of preference (i.e., the first algorithm is the most preferred).

5.2.10.2 Error Specific Details for Error Code 7

The following is the format of the Error Specific Details field for Error Code 7.

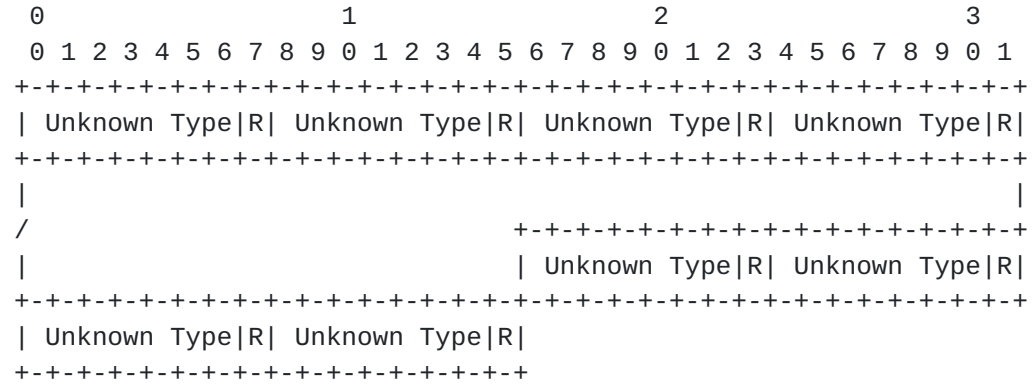


Figure 18: Unknown attributes format

Unknown Type: these 7-bit fields contain the Types of the attributes (which were present in the message that triggered the Error message) that were unknown to the receiver

R: at this point, this bit is reserved. It SHOULD be set to zero by the sender of the message and MUST be ignored by the receiver.

5.2.11 ERROR-INFO

The following is the format of the ERROR-INFO attribute.

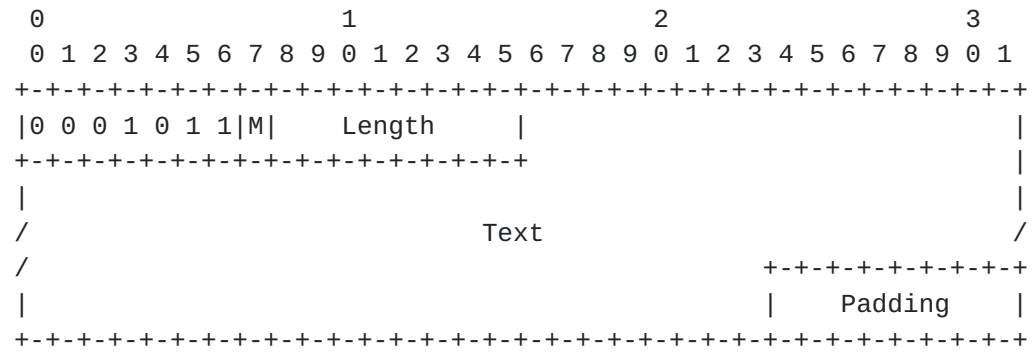


Figure 19: ERROR-INFO format

Text: this field contains UTF-8 [7] encoded text.

In some situations, the contents of the Text field may be generated by an automaton. If such automaton has information about the preferred language of the receiver of a particular ERROR-INFO attribute, 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 ERROR-INFO 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 needed.

5.2.12 PARTICIPANT-PROVIDED-INFO

The following is the format of the PARTICIPANT-PROVIDED-INFO attribute.

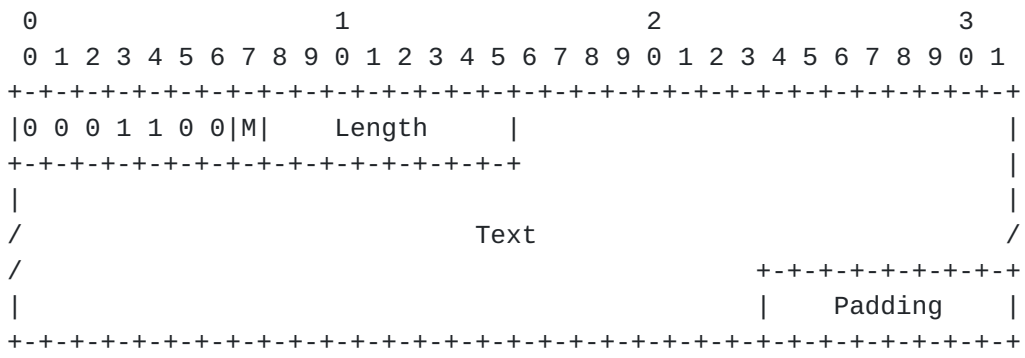


Figure 20: PARTICIPANT-PROVIDED-INFO format

Text: this field contains UTF-8 [7] encoded text.

Padding: one, two, or three bytes of padding added so that the contents of the PARTICIPANT-PROVIDED-INFO 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 needed.

5.2.13 STATUS-INFO

The following is the format of the STATUS-INFO attribute.

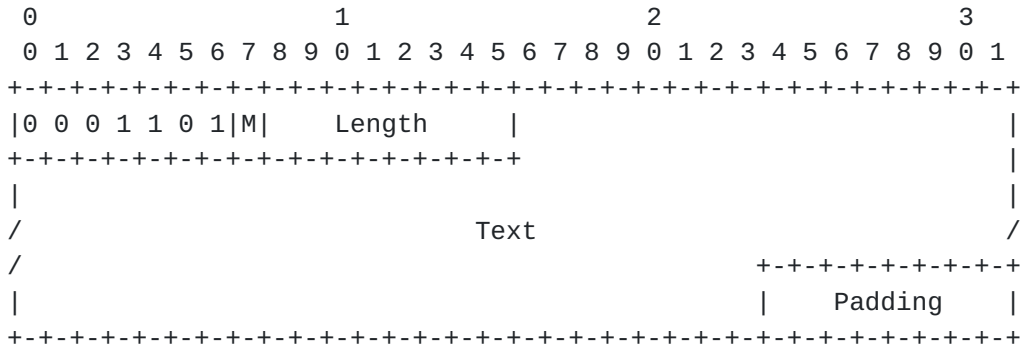


Figure 21: STATUS-INFO format

Text: this field contains UTF-8 [7] encoded text.

In some situations, the contents of the Text field may be generated by an automaton. If such automaton has information about the preferred language of the receiver of a particular STATUS-INFO attribute, 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 STATUS-INFO 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 needed.

5.2.14 SUPPORTED-ATTRIBUTES

The following is the format of the SUPPORTED-ATTRIBUTES attribute.

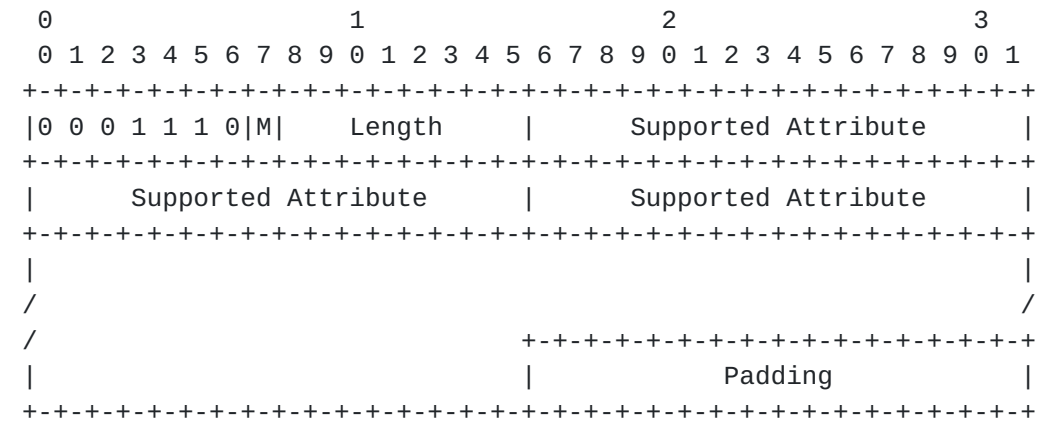


Figure 22: SUPPORTED-ATTRIBUTES format

Supported Attribute: these fields contain the Types of the attributes that are supported by the floor control server.

Padding: two bytes of padding added so that the contents of the SUPPORTED-ATTRIBUTES attribute is 32-bit aligned. If the attribute 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.

5.2.15 SUPPORTED-PRIMITIVES

The following is the format of the SUPPORTED-PRIMITIVES attribute.

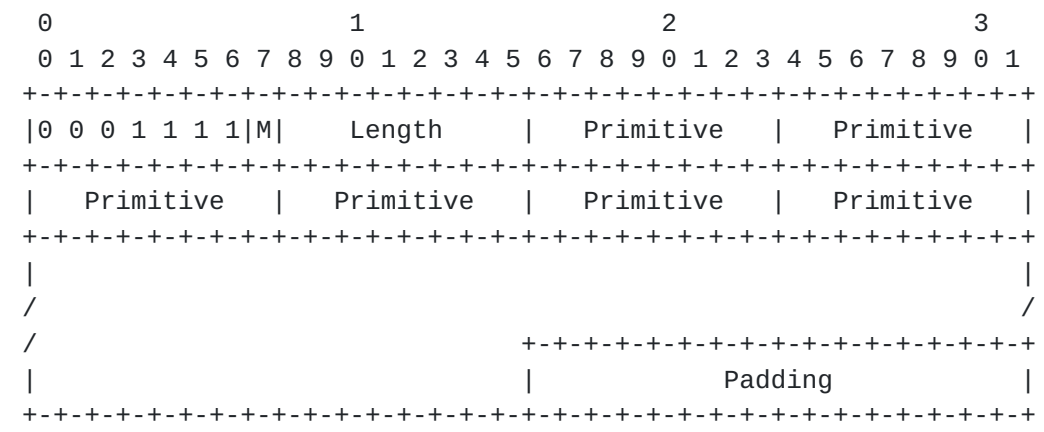


Figure 23: SUPPORTED-PRIMITIVES format

Primitive: these fields contain the types of the BFCP messages that are supported by the floor control server. See Table 1 for the list of BFCP primitives.

Padding: one, two, or three bytes of padding added so that the contents of the SUPPORTED-PRIMITIVES attribute is 32-bit aligned. If the attribute 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.

5.2.16 USER-DISPLAY-NAME

The following is the format of the USER-DISPLAY-NAME attribute.

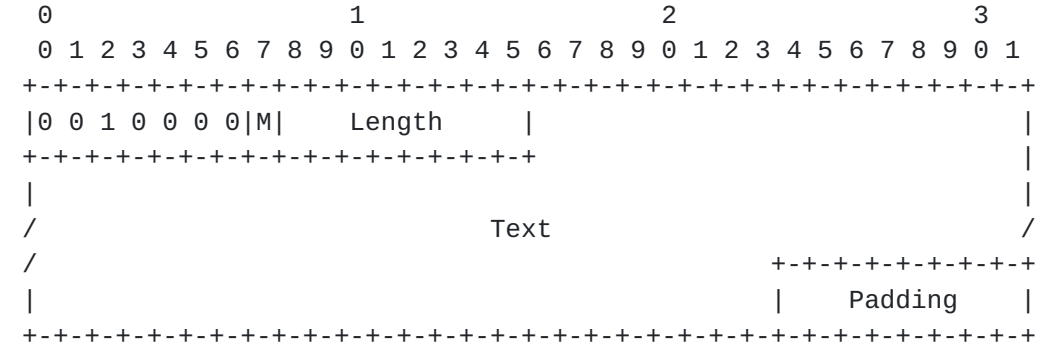


Figure 24: USER-DISPLAY-NAME format

Text: this field contains the UTF-8 encoded name of the user.

Padding: one, two, or three bytes of padding added so that the contents of the USER-DISPLAY-NAME 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 needed.

5.2.17 USER-URI

The following is the format of the USER-URI attribute.

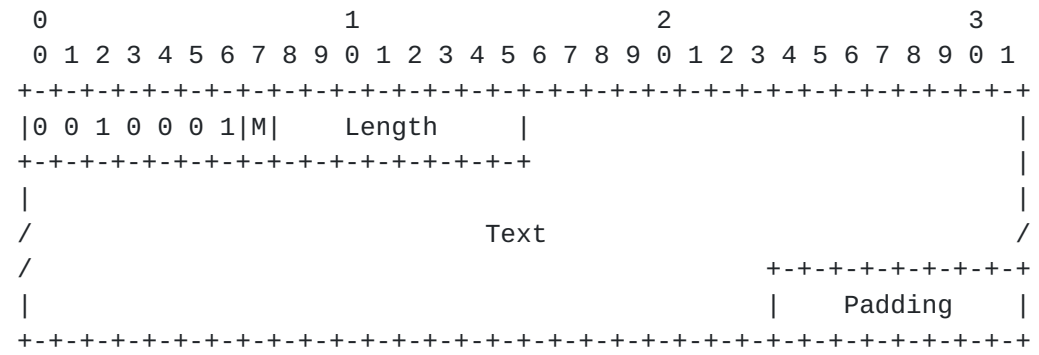


Figure 25: USER-URI format

Text: this field contains the UTF-8 encoded URI of the user.

Padding: one, two, or three bytes of padding added so that the contents of the USER-URI 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 needed.

5.2.18 BENEFICIARY-INFORMATION

The BENEFICIARY-INFORMATION attribute is a grouped attribute that consists of a header, which is referred to as BENEFICIARY-INFORMATION-HEADER, followed by a sequence of attributes. The following is the format of the BENEFICIARY-INFORMATION-HEADER:

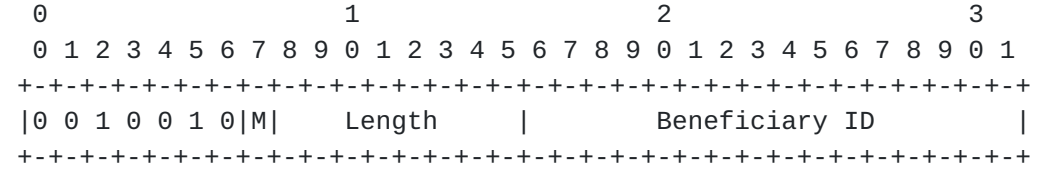


Figure 26: BENEFICIARY-INFORMATION-HEADER format

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

The following is the ABNF (Augmented Backus-Naur Form) [3] of the BENEFICIARY-INFORMATION grouped attribute. (EXTENSION-ATTRIBUTE refers to extension attributes that may be defined in the future.)

```
BENEFICIARY-INFORMATION =   (BENEFICIARY-INFORMATION-HEADER)
                            [USER-DISPLAY-NAME]
                            [USER-URI]
                           *[EXTENSION-ATTRIBUTE]
```

Figure 27: BENEFICIARY-INFORMATION format

5.2.19 FLOOR-REQUEST-INFORMATION

The FLOOR-REQUEST-INFORMATION attribute is a grouped attribute that consists of a header, which is referred to as FLOOR-REQUEST-INFORMATION-HEADER, followed by a sequence of attributes. The following is the format of the FLOOR-REQUEST-INFORMATION-HEADER:

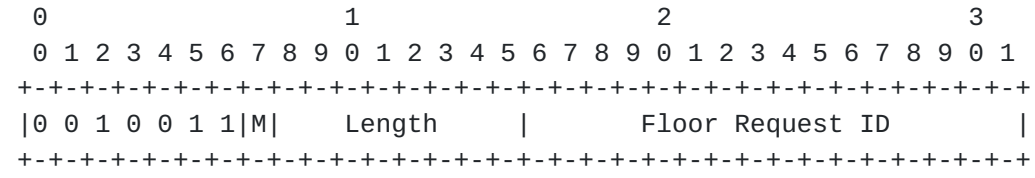


Figure 28: FLOOR-REQUEST-INFORMATION-HEADER format

Floor Request ID: this field contains a 16-bit value that indentifies a floor request at the floor control server.

The following is the ABNF of the FLOOR-REQUEST-INFORMATION grouped attribute. (EXTENSION-ATTRIBUTE refers to extension attributes that may be defined in the future.)

```
FLOOR-REQUEST-INFORMATION =   (FLOOR-REQUEST-INFORMATION-HEADER)
                              (REQUEST-STATUS)
                             1*(FLOOR-ID)
                              [BENEFICIARY-INFORMATION]
                              [REQUESTED-BY-INFORMATION]
                              [PRIORITY]
                              [PARTICIPANT-PROVIDED-INFO]
                              [STATUS-INFO]
                              *[EXTENSION-ATTRIBUTE]
```

Figure 29: FLOOR-REQUEST-INFORMATION format

5.2.20 REQUESTED-BY-INFORMATION

The REQUESTED-BY-INFORMATION attribute is a grouped attribute that consists of a header, which is referred to as REQUESTED-BY-INFORMATION-HEADER, followed by a sequence of attributes. The following is the format of the REQUESTED-BY-INFORMATION-HEADER:

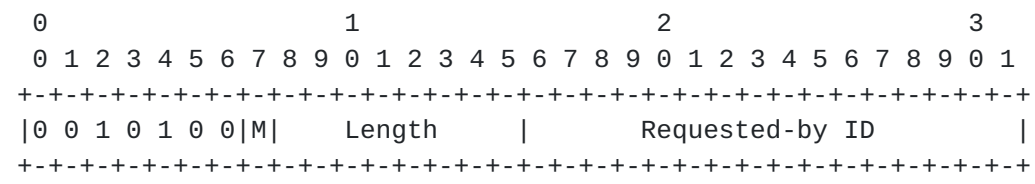


Figure 30: REQUESTED-BY-INFORMATION-HEADER format

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

The following is the ABNF of the REQUESTED-BY-INFORMATION grouped attribute. (EXTENSION-ATTRIBUTE refers to extension attributes that may be defined in the future.)

```
REQUESTED-BY-INFORMATION =   (REQUESTED-BY-INFORMATION-HEADER)
                             [USER-DISPLAY-NAME]
                             [USER-URI]
                            *[EXTENSION-ATTRIBUTE]
```

Figure 31: REQUESTED-BY-INFORMATION format

5.3 Message Format

This section contains the normative ABNF (Augmented Backus-Naur Form) [3] of the BFCP messages. Extension attributes that may be defined in the future are referred to as EXTENSION-ATTRIBUTE in the ABNF.

5.3.1 FloorRequest

Floor participants 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)
                 (USER-ID)
                 (TRANSACTION-ID)
                *(FLOOR-ID)
                 [BENEFICIARY-ID]
                 [PARTICIPANT-PROVIDED-INFO]
                 [PRIORITY]
                *[EXTENSION-ATTRIBUTE]
                 [NONCE]
                 [DIGEST]
```

Figure 32: FloorRequest format

5.3.2 FloorRelease

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

```
FloorRelease =   (FIXED-HEADER)
                 (USER-ID)
                 (TRANSACTION-ID)
                 (FLOOR-REQUEST-ID)
                *[EXTENSION-ATTRIBUTE]
                 [NONCE]
                 [DIGEST]
```

Figure 33: FloorRelease format

5.3.3 FloorRequestInfoWanted

Floor participants and floor chairs request information about a floor request by sending a FloorRequestInfoWanted message to the floor control server. The following is the format of the FloorRequestInfoWanted message:

```
FloorRequestInfoWanted =   (FIXED-HEADER)
                           (USER-ID)
                           (TRANSACTION-ID)
                           (FLOOR-REQUEST-ID)
                          *[EXTENSION-ATTRIBUTE]
                           [NONCE]
                           [DIGEST]
```

Figure 34: FloorRequestInfoWanted format

5.3.4 FloorRequestInfo

The floor control server informs floor participants and floor chairs about the status of their floor requests by sending them FloorRequestInfo messages. The following is the format of the FloorRequestInfo message:

```
FloorRequestInfo =     (FIXED-HEADER)
                       (USER-ID)
                       [TRANSACTION-ID]
                       (FLOOR-REQUEST-INFORMATION)
                       [NONCE]
                      *[EXTENSION-ATTRIBUTE]
```

Figure 35: FloorRequestInfo format

5.3.5 UserInfoWanted

Floor participants and floor chairs request information about a participant and the floor requests related to this participant by sending a UserInfoWanted message to the floor control server. The following is the format of the UserInfoWanted message:

```
UserInfoWanted =   (FIXED-HEADER)
                   (TRANSACTION-ID)
                   (USER-ID)
                   [BENEFICIARY-ID]
                  *[EXTENSION-ATTRIBUTE]
                   [NONCE]
                   [DIGEST]
```

Figure 36: UserInfoWanted format

5.3.6 UserInfo

The floor control server provide information about participants and their related floor requests to floor participants and floor chairs by sending them UserInfo messages. The following is the format of the UserInfo message:

```
UserInfo =     (FIXED-HEADER)
               (TRANSACTION-ID)
               (USER-ID)
               [BENEFICIARY-INFORMATION]
             1*(FLOOR-REQUEST-INFORMATION)
               [NONCE]
              *[EXTENSION-ATTRIBUTE]
```

Figure 37: UserInfo format

5.3.7 FloorInfoWanted

Floor participants and floor chairs request information about a floor or floors by sending a FloorInfoWanted message to the floor control server. The following is the format of the FloorRequest message:

```
FloorInfoWanted =   (FIXED-HEADER)
                    (USER-ID)
                    (TRANSACTION-ID)
                   *(FLOOR-ID)
                   *[EXTENSION-ATTRIBUTE]
                    [NONCE]
                    [DIGEST]
```

Figure 38: FloorInfoWanted format

5.3.8 FloorInfo

The floor control server informs floor participants and floor chairs about the status (e.g., the current holder) of a floor by sending them FloorInfo messages. The following is the format of the FloorInfo message:

```
FloorInfo        =       (FIXED-HEADER)
                         (USER-ID)
                         [TRANSACTION-ID]
                         (FLOOR-ID)
                        *[FLOOR-REQUEST-INFORMATION]
                         [NONCE]
                        *[EXTENSION-ATTRIBUTE]
```

Figure 39: FloorInfo format

5.3.9 ChairAction

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

```
ChairAction  =   (FIXED-HEADER)
                 (USER-ID)
                 (TRANSACTION-ID)
                1*(FLOOR-ID)
                 (FLOOR-REQUEST-ID)
                 (REQUEST-STATUS)
                 [STATUS-INFO]
                 *[EXTENSION-ATTRIBUTE]
                 [NONCE]
                 [DIGEST]
```

Figure 40: ChairAction format

5.3.10 ChairActionAck

Floor control servers confirm that they have accepted a ChairAction message by sending a ChairActionAck message. The following is the format of the ChairActionAck message:

```
ChairActionAck  =   (FIXED-HEADER)
                    (USER-ID)
                    (TRANSACTION-ID)
                    [NONCE]
                   *[EXTENSION-ATTRIBUTE]
```

Figure 41: ChairActionAck format

5.3.11 Hello

Floor participants and floor chairs check the liveness of floor control servers by sending a Hello message. The following is the format of the Hello message:

```
Hello         =  (FIXED-HEADER)
                 (TRANSACTION-ID)
                 (USER-ID)
                *[EXTENSION-ATTRIBUTE]
                 [NONCE]
                 [DIGEST]
```

Figure 42: Hello format

5.3.12 HelloAck

Floor control servers confirm that they are alive on reception of a Hello message by sending a HelloAck message. The following is the format of the HelloAck message:

```
HelloAck      =  (FIXED-HEADER)
                 (TRANSACTION-ID)
                 (USER-ID)
                 (SUPPORTED-PRIMITIVES)
                 (SUPPORTED-ATTRIBUTES)
                 [NONCE]
                *[EXTENSION-ATTRIBUTE]
```

Figure 43: HelloAck format

5.3.13 Error

Floor control servers inform floor participants and floor chairs about errors processing requests by sending them Error messages. The following is the format of the Error message:

```
Error             =   (FIXED-HEADER)
                      (TRANSACTION-ID)
                      (USER-ID)
                      (ERROR-CODE)
                      [NONCE]
                      [ERROR-INFO]
                     *[EXTENSION-ATTRIBUTE]
```

Figure 44: Error format

6. Transport

BFCP entities exchange BFCP messages using TCP connections. TCP provides an in-order reliable delivery of a stream of bytes. Consequently, message framing is implemented in the application layer. BFCP implements application-layer framing using TLV-encoded attributes.

A client MUST NOT use more than one TCP connection to communicate with a given floor control server within a conference. Nevertheless, if the same physical box handles different clients (e.g., a floor chair and a floor participant), which are identified by different User IDs, a separate connection per client is allowed.

If a BFCP entity (a client or a floor control server) receives data from TCP that cannot be parsed the entity MUST close the TCP connection using a RESET call (send a TCP RST bit) and the connection SHOULD be reestablished. Similarly, if a TCP connection cannot deliver a BFCP message and times out, the TCP connection SHOULD be reestablished.

The way connection reestablishment is handled depends on how the client obtains information to contact the floor control server (e.g., using an offer/answer exchange [13]). Once the TCP connection is reestablished, the client MAY resend those message it did not get a response for from the floor control server.

If a floor control server detects that the TCP connection towards one of the floor participants is lost, it is up to the local policy of the floor control server what to do with the pending floor requests of the floor participant. In any case, it is RECOMMENDED that the floor control server keeps the floor requests (i.e., does not cancel them) while the TCP connection is reestablished.

If a client wishes to end its BFCP connection with a floor control server, the client closes (i.e., a graceful close) the TCP connection

towards the floor control server. If a floor control server wishes to end its BFCP connection with a client (e.g., the Focus of the conference informs the floor control server that the client has been kicked out from the conference), the floor control server closes (i.e., a graceful close) the TCP connection towards the client.

7. Lower-Layer Security

BFCP relies on lower-layer security mechanisms to provide replay and integrity protection, and confidentiality. BFCP floor control servers MUST support TLS [4], and BFCP clients (which include both floor participants and floor chairs) SHOULD support TLS. Any BFCP entity MAY support other security mechanisms.

BFCP entities that implement TLS MUST support, at a minimum, the TLS TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite [6].

8. Protocol Transactions

In BFCP, there are two types of transactions: client-initiated transactions and server-initiated transactions (notifications). Client-initiated transactions consist of a request from a client to a floor control server and a response from the floor control server to the client. The request carries a TRANSACTION-ID attribute which the floor control server copies into the response. Clients use Transaction ID values to match responses with previously-issued requests.

Server-initiated transactions consist of a single message from a floor control server to a client. Since they do not trigger any response, server-initiated transactions do not have Transaction IDs associated with them.

8.1 Client Behavior

A client starting a client-initiated transaction MUST set the Conference ID in the FIXED-HEADER of the message to the Conference ID for the conference that the client obtained previously.

The client MUST set the Transaction ID value in the TRANSACTION-ID attribute to a number which MUST NOT be reused in another message from the client until a response from the server is received for the transaction. The client uses the Transaction ID value to match this message with the response from the floor control server.

8.2 Server Behavior

A floor control server sending a response within a client-initiated

transaction MUST copy the Conference ID, the TRANSACTION-ID attribute, and the USER-ID attribute from the request received from the client into the response. Server-initiated transactions MUST NOT contain a TRANSACTION-ID attribute.

9. Authentication and Authorization

BFCP clients SHOULD authenticate the floor control server before sending any BFCP message to it. Similarly, floor control servers SHOULD authenticate a client before accepting any BFCP message from it.

BFCP supports TLS-based mutual authentication between clients and floor control servers, as specified in Section 9.1. This is the RECOMMENDED authentication mechanism in BFCP.

Additionally, BFCP also provides a digest mechanism based on a shared secret to provide client authentication in situations where TLS is not used for some reason. This mechanism is described in Section 9.2.

9.1 TLS-based Mutual Authentication

BFCP supports TLS-based mutual authentication between clients and floor control servers. Authentication based on both, certificates signed by a certificate authority and self-signed certificates is supported.

If a client and a floor control server have certificates signed by a certificate authority known to both, they can use these certificates to authenticate each other at TLS establishment time. Otherwise, BFCP assumes that there is an integrity-protected channel between the client and the floor control server that can be used to exchange their self-signed certificates or, more commonly, the fingerprints of these certificates. These certificates are used at TLS establishment time.

> The implementation of such an integrity-protected channel using SIP and the offer/answer model is described in [13].

9.2 Digest-based Client Authentication

BFCP supports digest-based client authentication based on a shared secret between a client and the floor control server. It is assumed that an encrypted and integrity-protected channel exists between the client and the floor control server. This channel is used to generate a shared secret between them.

The implementation of such an encrypted and integrity-protected channel using SIP and the offer/answer model is described in [13].

Digest-based client authentication in BFCP is based on the DIGEST attribute. This attribute contains an algorithm identifier and a keyed digest of the of the BFCP message using that algorithm. The text used as input to the digest algorithm is the BFCP message, including the FIXED-HEADER, up to and including the attribute preceding the DIGEST attribute. Depending on the algorithm, this text may need to be padded with zeroes. Section 5.2.9 lists the algorithms specified in BFCP.

The key used as input to the keyed digest is the secret shared between the server and the user identified by the USER-ID attribute in the message.

Section 9.2.1 and Section 9.2.2 discuss how to achieve client authentication using the DIGEST attribute.

9.2.1 Client Behavior

To achieve client authentication, a client needs to prove to the floor control server that the client can produce a DIGEST attribute for a message using their shared secret and that the message is fresh (to avoid replay attacks). Clients prove the freshness of a message by including a NONCE attribute in the message. The NONCE attribute is the second to last attribute in the message (the last one is the DIGEST attribute).

Clients can obtain the digest algorithms supported by the floor control server in an Error response from the floor control server with Error Code 3 (DIGEST Attribute Required). A client SHOULD use the first digest algorithm in the list that it supports.

Additionally, as an optimization, the floor control server and the client can agree on the algorithm to be used using an out-of-band mechanism (e.g., using an offer/answer exchange as described in [13]). This way, the client does not need to generate an initial BFCP message only to have it rejected by the floor control server with an Error response containing a list with its supported algorithms.

If after sending a message with a DIGEST attribute, a client receives an Error response with Error Code 3 (DIGEST Attribute Required) with a list of digest algorithms, the client SHOULD re-send the message using the first digest algorithm in the list that it supports.

The nonce to be placed in the NONCE attribute by the client is

typically provided by the floor control server in an Error response -- typically with Error Code 3 (DIGEST Attribute Required) or 6 (Invalid Nonce). Additionally, as an optimization, the floor control server can provide a client with a NONCE to be used in the first message generated by the client using an out-of-band mechanism (e.g., using an offer/answer exchange as described in [13]). This way, the client does not need to generate an initial BFCP message only to have it rejected by the floor control server with an Error response containing a nonce.

A client that obtains a nonce out-of-band SHOULD add a NONCE attribute and a DIGEST attribute to the first message it sends to the floor control server. Furthermore, if any client generates a message without a DIGEST attribute and receives an Error response with Error Code 3 (DIGEST Attribute Required), the client SHOULD re-send the message with a DIGEST attribute and a NONCE attribute with the nonce received in the Error response.

If after sending a message with a DIGEST attribute, a client receives an Error response with Error Code 4 (Invalid Nonce), the client SHOULD re-send the message using the new nonce received in the Error response. If the Error Code is 5 (Authentication Failed) instead, the client MUST NOT send further messages to the floor control server until it has obtained a different (hopefully valid) shared secret than the one used in the original message.

If a client receives a nonce in a message from the floor control server, the client SHOULD add a NONCE attribute with this nonce and a DIGEST attribute to its next message to the floor control server.

9.2.2 Floor Control Server Behavior

If the floor control server receives a message without DIGEST attribute from an unauthenticated client, the floor control server responds with an Error message with Error Code 3 (DIGEST Attribute Required). The floor control message MUST include a list with the digest algorithms supported by the floor control server in order of preference (i.e., the first algorithm is the most preferred) and a NONCE attribute with a nonce value that is unguessable by attackers.

When a floor control server receives a BFCP message with a DIGEST attribute, it checks whether the Algorithm identifier in the DIGEST attribute corresponds to an algorithm that is supported by the floor control server. If it does not, the floor control server SHOULD return an Error message with Error Code 3 (DIGEST Attribute Required) with a list with the digest algorithms supported by the floor control server.

If the algorithm identifier is valid, the floor control server checks whether the NONCE attribute carries a nonce which was generated by the floor control server for this client and which still has not expired. If the nonce is not valid, authentication is considered to have failed, in which case the floor control server SHOULD return an Error message with Error Code 4 (Invalid Nonce) with a new nonce in a NONCE attribute.

If the nonce is valid, the floor control server calculates the keyed digest of the message using the algorithm identified by the DIGEST attribute. The key used as input to the keyed digest is the secret shared between the server and the user identified by the USER-ID attribute in the message. If the resulting value is the same as the one in the DIGEST attribute, authentication is considered successful.

If the resulting value is different than the one in the DIGEST attribute, authentication is considered to have failed, in which case the server SHOULD return an Error message, as described in Section 13.8, with Error Code 5 (Authentication Failed). Messages from a client that cannot be authenticated MUST NOT be processed further.

Floor control servers MAY include a NONCE attribute in their responses to provide the nonce to be used in the next message by the client. However, when TLS is used, floor control servers typically authenticate only the first message sent over the TLS connection.

After authenticating a BFCP message, the floor control server checks whether or not the client is authorized to perform the operation it is requesting. If the client is not authorized to perform the operation being requested, the floor control server generates an Error message, as described in Section 13.8, with an Error code with a value of 8 (Unauthorized Operation). Messages from a client that cannot be authorized MUST NOT be processed further.

10. Floor Participant Operations

This section specifies how floor participants can perform different operations, such as requesting a floor, using the protocol elements described in earlier sections. Section 11 specifies operations that are specific to floor chairs, such as instructing the floor control server to grant or revoke a floor, and Section 12 specifies operations that can be performed by any client (i.e., both floor participants and floor chairs).

10.1 Requesting a Floor

A floor participant that wishes to request one or more floors does so

by sending a FloorRequest message to the floor control server.

10.1.1 Sending a FloorRequest Message

The ABNF in Section 5.3.1 describes the attributes that a FloorRequest message can contain. In addition, the ABNF specifies normatively which of these attributes are mandatory, and which ones are optional.

The floor participant sets the Conference ID in the FIXED-HEADER and the TRANSACTION-ID attribute following the rules given in Section 8.1. Additionally, the floor participant follows the rules in Section 9 which relate to the authentication of the message.

The floor participant must insert a USER-ID attribute, which will be used by the floor control server to authenticate and authorize the request. If the sender of the FloorRequest message (identified by the USER-ID attribute) is not the participant that would eventually get the floor (i.e., a third party floor request), the sender SHOULD add a BENEFICIARY-ID attribute to the message identifying the beneficiary of the floor.

> Note that the name space for both the User ID and the Beneficiary ID is the same. That is, a given participant is identified by a single 16-bit value that can be used in several attributes: USER-ID, BENEFICIARY-ID, BENEFICIARY-INFORMATION, and REQUESTED-BY-INFORMATION.

The floor participant must insert at least one FLOOR-ID attribute in the FloorRequest message. If the client inserts more than one FLOOR-ID attributes, the floor control 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 floor participant may use a PARTICIPANT-PROVIDED-INFO attribute to state the reason why the floor or floors are being requested. The Text field in the PARTICIPANT-PROVIDED-INFO attribute is intended for human consumption.

The floor participant may request the server to handle the floor request with a certain priority using a PRIORITY attribute.

10.1.2 Receiving a Response

A message from the floor control server is considered to be a response to the FloorRequest message if the message from the floor control server has the same Conference ID, Transaction ID, and User

ID as the FloorRequest message, as described in Section 8.1.

The successful processing of a FloorRequest message at the floor control server involves generating one or several FloorRequestInfo messages. The floor participant obtains a Floor Request ID in the Floor Request ID field of a FLOOR-REQUEST-INFORMATION attribute in the first FloorRequestInfo message from the floor control server. Subsequent FloorRequestInfo messages from the floor control server regarding the same floor request will carry the same Floor Request ID in a FLOOR-REQUEST-INFORMATION attribute as the initial FloorRequestInfo message. This way, the floor participant can associate subsequent incoming FloorRequestInfo messages with the ongoing floor request.

The floor participant obtains information about the status of the floor request in the FLOOR-REQUEST-INFORMATION attribute of each of the FloorRequestInfo messages received from the floor control server. This attribute is a grouped attribute and, as such, it includes a number of attributes that provide information about the floor request.

The REQUEST-STATUS attribute. If the Request Status value is Granted, all the floors that were requested in the FloorRequest message have been granted. If the Request Status value is Denied, all the floors that were requested in the FloorRequest message have been denied. A floor request is considered to be ongoing while it is in the Pending, Accepted, or Granted states.

The STATUS-INFO attribute, if present, provides extra information which the floor participant MAY display to the user.

The BENEFICIARY-INFORMATION attribute identifies the beneficiary of the floor request in third-party floor requests. The REQUESTED-BY-INFORMATION attribute may be not be present in FloorRequestInfo messages received by the floor participant that requested the floor because this floor participant is already identified by the USER-ID attribute.

The PRIORITY attribute, when present, contains the priority that was requested by the generator of the FloorRequest message.

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

10.2 Cancelling a Floor Request and Releasing a Floor

A floor participant that wishes to cancel an ongoing floor request

does so by sending a FloorRelease message to the floor control server. The FloorRelease message is also used by floor participants that hold a floor and would like to release it.

10.2.1 Sending a FloorRelease Message

The ABNF in Section 5.3.2 describes the attributes that a FloorRelease message can contain. In addition, the ABNF specifies normatively which of these attributes are mandatory, and which ones are optional.

The floor participant sets the Conference ID in the FIXED-HEADER and the TRANSACTION-ID attribute following the rules given in Section 8.1. Additionally, the floor participant follows the rules in Section 9 which relate to the authentication of the message. The floor participant must insert a USER-ID attribute, which will be used by the floor control server to authenticate and authorize the request.

> Note that the FloorRelease message is used to release a floor or floors that were granted and to cancel ongoing floor requests (from the protocol perspective both are ongoing floor requests). Using the same message in both situations helps resolve the race condition that occurs when the FloorRelease message and the FloorGrant message cross each other on the wire.

The floor participant uses the FLOOR-REQUEST-ID that was received in the response to the FloorRequest message that the FloorRelease message is cancelling.

> Note that if the floor participant requested several floors as an atomic operation (i.e., in a single FloorRequest message), all the floors are released as an atomic operation as well (i.e., all are released at the same time).

10.2.2 Receiving a Response

A message from the floor control server is considered to be a response to the FloorRelease message if the message from the floor control server has the same Conference ID, Transaction ID, and User ID as the FloorRequest message, as described in Section 8.1.

If the response is a FloorRequestInfo message, the Request Status value in the REQUEST-STATUS attribute (within the FLOOR-REQUEST-INFORMATION grouped attribute) will be Cancelled or Released.

If the response is an Error message, the floor control server could not process the FloorRequest message for some reason, which is

described in the Error message.

It is possible that the FloorRelease message crosses on the wire with a FloorRequestInfo message from the server with a Request Status different from Cancelled or Released. In any case, such a FloorRequestInfo message will not be a response to the FloorRelease message, because its Transaction ID will not match that of the FloorRelease.

11. Chair Operations

This section specifies how floor chairs can instruct the floor control server to grant or revoke a floor using the protocol elements described in earlier sections.

Floor chairs that wish to send instructions to a floor control server do so by sending a ChairAction message.

11.1 Sending a ChairAction Message

The ABNF in Section 5.3.9 describes the attributes that a ChairAction message can contain. In addition, the ABNF specifies normatively which of these attributes are mandatory, and which ones are optional.

The floor chair sets the Conference ID in the FIXED-HEADER and the TRANSACTION-ID attribute following the rules given in Section 8.1. Additionally, the floor chair follows the rules in Section 9 which relate to the authentication of the message. The floor chair must insert a USER-ID attribute, which will be used by the floor control server to authenticate and authorize the request.

The ChairAction message contains instructions that apply to one or more floors within a particular floor request. The floor or floors are identified by FLOOR-ID attributes and the floor request is identified by a FLOOR-REQUEST-ID attribute, which are carried in the ChairAction message.

> For example, if a floor request consists of two floors that depend on different floor chairs, each floor chair will grant its floor within the floor request. Once both chairs have granted their floor, the floor control server will grant the floor request as a whole. On the other hand, if one of the floor chairs denies its floor, the floor control server will deny the floor request as a whole, regardless of the other floor chair's decision.

The floor chair provides the new status for one or more floors within the floor request using a REQUEST-STATUS attribute. If the new status of the floor request is Accepted, the floor chair MAY use the

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

> Note that a floor request may involve several floors and that a ChairAction message may only deal with a subset of these floors (e.g., if a single floor chair is not authorized to manage all the floors). In this case, the REQUEST-STATUS that the floor chair provides in the ChairAction message might not be the actual status that the floor request gets at the server. The floor control server will combine the instructions received from the different floor chairs to come up with the actual status of the floor request.

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

11.2 Receiving a Response

A message from the floor control server is considered to be a response to the ChairAction message if the message from the server has the same Conference ID, Transaction ID, and User ID as the ChairAction message, as described in Section 8.1.

A ChairActionAck message from the floor control server confirms that the floor control server has accepted the ChairAction message. An Error message indicates that the floor control server could not process the ChairAction message for some reason, which is described in the Error message.

12. General Client Operations

This section specifies operations that can be performed by any client. That is, they are not specific to floor participants or floor chairs. They can be performed by both.

12.1 Requesting Information about Floors

A client can obtain information about the status of a floor or floors in different ways, which include using BFCP and using out-of-band mechanisms. Clients using BFCP to obtain such information use the procedures described in this section.

Clients request information about the status of one or several floors by sending a FloorInfoWanted message to the floor control server.

12.1.1 Sending a FloorInfoWanted Message

The ABNF in Section 5.3.7 describes the attributes that a FloorInfoWanted message can contain. In addition, the ABNF specifies normatively which of these attributes are mandatory, and which ones are optional.

The client sets the Conference ID in the FIXED-HEADER and the TRANSACTION-ID attribute following the rules given in Section 8.1. Additionally, the client follows the rules in Section 9 which relate to the authentication and the protection of the integrity of the message. The client must insert a USER-ID attribute, which will be used by the floor control server to authenticate and authorize the request.

The client inserts in the message all the Floor IDs it wants to receive information about. The floor control server will send periodic information about all these floors. If the client does not want to receive information about a particular floor any longer, it sends a new FloorInfoWanted message removing the FLOOR-ID of this floor. If the client does not want to receive information about any floor any longer, it sends a FloorInfoWanted message with no FLOOR-ID attribute.

12.1.2 Receiving a Response

A message from the floor control server is considered to be a response to the FloorInfoWanted message if the message from the floor control server has the same Conference ID, Transaction ID, and User ID as the FloorRequest message, as described in Section 8.1.

On reception of the FloorInfoWanted message, the floor control server will respond with a FloorInfo message or with an Error message. If the response is a FloorInfo message, it will contain information about one of the floors the client requested information about. If the client did not include any FLOOR-ID attribute in its FloorInfoWanted message (i.e., the client does not want to receive information about any floor any longer), the FloorInfo message from the floor control server will not include any FLOOR-ID attribute either.

FloorInfo messages which carry information about a floor contain a FLOOR-ID attribute that identifies the floor. After this attribute, FloorInfo messages contain information about existing (one or more) floor request that relate to that floor. The information about each

particular floor request is encoded in a FLOOR-REQUEST-INFORMATION attribute. This grouped attribute carries a Floor Request ID that identifies the floor request followed by a set of attributes that provide information about the floor request.

After the first FloorInfo, the floor control server will continue sending FloorInfo messages periodically informing the client about changes on the floors the client requested information about.

12.2 Requesting Information about Floor Requests

A client can obtain information about the status of one or several floor requests in different ways, which include using BFCP and using out-of-band mechanisms. Clients using BFCP to obtain such information use the procedures described in this section.

Clients request information about the current status of a floor requests by sending a FloorRequestInfoWanted message to the floor control server.

Requesting information about a particular floor request is useful in a number of situations. For example, on reception of a FloorRequest message, a floor control server may choose to return FloorRequestInfo messages only when the floor request changes its state (e.g., from Accepted to Granted), but not when the floor request advances in its queue. In this situation, if the user requests it, the floor participant can use a FloorRequestInfoWanted message to poll the floor control server for the status of the floor request.

12.2.1 Sending a FloorRequestInfoWanted Message

The ABNF in Section 5.3.3 describes the attributes that a FloorRequestInfoWanted message can contain. In addition, the ABNF specifies normatively which of these attributes are mandatory, and which ones are optional.

The client sets the Conference ID in the FIXED-HEADER and the TRANSACTION-ID attribute following the rules given in Section 8.1. Additionally, the client follows the rules in Section 9 which relate to the authentication of the message. The client must insert a USER-ID attribute, which will be used by the floor control server to authenticate and authorize the request.

The client must insert a FLOOR-REQUEST-ID attribute that identifies the floor request at the floor control server.

12.2.2 Receiving a Response

A message from the floor control server is considered to be a response to the FloorRequestInfoWanted message if the message from the floor control server has the same Conference ID, Transaction ID,and User ID as the FloorRequestInfoWanted message, as described in Section 8.1.

If the response is a FloorRequestInfo message, the client obtains information about the status of the FloorRequest the client requested information about in a FLOOR-REQUEST-INFORMATION attribute.

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

12.3 Requesting Information about a User

A client can obtain information about a participant and the floor requests related to this participant in different ways, which include using BFCP and using out-of-band mechanisms. Clients using BFCP to obtain such information use the procedures described in this section.

Clients request information about a participant and the floor requests related to this participant by sending a UserInfoWanted message to the floor control server.

This functionality may be useful for floor chairs or floor participants interested in the display name and the URI of a particular floor participant. In addition, a floor participant may find it useful to request information about itself. For example, a floor participant, after experiencing connectivity problems (e.g., its TCP connection with the floor control server was down for a while and eventually was re-established), may need to request information about all the still existing floor requests associated to itself.

12.3.1 Sending a UserInfoWanted Message

The ABNF in Section 5.3.5 describes the attributes that a UserInfoWanted message can contain. In addition, the ABNF specifies normatively which of these attributes are mandatory, and which ones are optional.

The client sets the Conference ID in the FIXED-HEADER and the TRANSACTION-ID attribute following the rules given in Section 8.1. Additionally, the client follows the rules in Section 9 which relate to the authentication of the message. The client must insert a USER-ID attribute, which will be used by the floor control server to

authenticate and authorize the request.

If the floor participant the client is requesting information about is not the client issuing the UserInfoWanted message (which is identified by the USER-ID attribute in the message) the client MUST insert a BENEFICIARY-ID attribute.

12.3.2 Receiving a Response

A message from the floor control server is considered to be a response to the UserInfoWanted message if the message from the floor control server has the same Conference ID, Transaction ID, and User ID as the UserInfoWanted message, as described in Section 8.1.

If the response is a UserInfo message, the client obtains information about the floor participant in a BENEFICIARY-INFORMATION grouped attribute and about the status of the floor requests associated with the floor participant in FLOOR-REQUEST-INFORMATION attributes.

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

12.4 Obtaining the Capabilities of a Floor Control Server

A client that wishes to obtain the capabilities of a floor control server does so by sending a Hello message to the floor control server.

12.4.1 Sending a Hello Message

The ABNF in Section 5.3.11 describes the attributes that a Hello message can contain. In addition, the ABNF specifies normatively which of these attributes are mandatory, and which ones are optional.

The client sets the Conference ID in the FIXED-HEADER and the TRANSACTION-ID attribute following the rules given in Section 8.1. Additionally, the client follows the rules in Section 9 which relate to the authentication and the protection of the integrity of the message. The client must insert a USER-ID attribute, which will be used by the floor control server to authenticate and authorize the request.

12.4.2 Receiving Responses

A message from the floor control server is considered a response to the Hello message by the client if the message from the floor control server has the same Conference ID, Transaction ID, and User ID as the

Hello message, as described in Section 8.1.

If the response is a HelloAck message, the floor control server could process successfully the Hello message. The SUPPORTED-ATTRIBUTES attribute indicates which attributes are supported by the server.

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

13. Floor Control Server Operations

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

On reception of a message from a client, the floor control server MUST check whether or not the value of the Conference ID matched an existing conference. If it does not, the floor control server SHOULD send an Error message, as described in Section 13.8, with Error code 1 (Conference does not Exist).

On reception of a message from a client, the floor control server follows the rules in Section 9, which relate to the authentication of the message.

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) attributes in the message. If the floor control server does not understand all of them, the floor control server SHOULD send an Error message, as described in Section 13.8, with Error code 2 (Authentication Failed). The Error message SHOULD list the attributes that were not understood.

13.1 Reception of a FloorRequest Message

On reception of a FloorRequest message, the floor control server follows the rules in Section 9 which relate to client authentication and authorization. If while processing the FloorRequest message, the floor control server encounters an error, it SHOULD generate an Error response following the procedures described in Section 13.8

BFCP allows floor participants to have several ongoing floor requests for the same floor (e.g., the same floor participant can occupy more than one position in a queue at the same time). A floor control server that only supports a certain number of ongoing floor requests per floor participant (e.g., one) can use Error Code 11 (You have Already Reached the Maximum Number of

Ongoing Floor Requests for this Floor) to inform the floor participant.

13.1.1 Generating the First FloorRequestInfo Message

The successful processing of a FloorRequest message by a floor control server involves generating one or several FloorRequestInfo messages, the first of which SHOULD be generated as soon as possible. If the floor control server cannot accept, grant, or deny the floor request right away (e.g., a decision from a chair is needed), it SHOULD use a Request Status value of Pending in the REQUEST-STATUS attribute (within the FLOOR-REQUEST-INFORMATION grouped attribute) of the first FloorRequestInfo message it generates.

The policy a floor control server follows to grant or deny floors is outside the scope of this document. A given floor control server may perform these decisions automatically while another may contact a human acting as a chair everytime a decision needs to be made.

The floor control server MUST copy the Conference ID, the TRANSACTION-ID, and the USER-ID attributes from the FloorRequest into the FloorRequestInfo, as described in Section 8.2. Additionally, the floor control server MUST add a FLOOR-REQUEST-INFORMATION grouped attribute to the FloorRequestInfo. The attributes contained in this grouped attribute carry information about the floor request.

The floor control server MUST assign an identitifier that is unique within the conference to this floor request, and MUST insert it in the Floor Request ID field of the FLOOR-REQUEST-INFORMATION attribute. This indentifier will be used by the floor participant (or by a chair or chairs) to refer to this specific floor request in the future.

The floor control server MUST copy the FLOOR-ID attributes from the FloorRequest into the FLOOR-REQUEST-INFORMATION attribute. These FLOOR-ID attributes identify the floors being requested (i.e., the floors associated with this particular floor request).

The floor control server SHOULD copy (if present) the contents of the BENEFICIARY-ID attribute from the FloorRequest into a BENEFICIARY-INFORMATION attribute inside the FLOOR-REQUEST-INFORMATION grouped attribute. Additionally, the floor control server MAY provide the display name and the URI of the beneficiary in this BENEFICIARY-INFORMATION attribute.

The floor control server MAY provide information about the requester of the floor in a REQUESTED-BY-INFORMATION attribute inside the

FLOOR-REQUEST-INFORMATION grouped attribute.

The floor control server MAY copy (if present) the PRIORITY attribute from the FloorRequest into the FLOOR-REQUEST-INFORMATION grouped attribute. Note that this attribute carries the priority requested by the participant. The priority the floor control server assigns to the floor request depends on the priority requested by the participant and the rights the participant has according to the policy of the conference. For example, a participant that is only allowed to use the Normal priority may request Highest priority for a floor request. In that case, the floor control server would ignore the priority requested by the participant.

The floor control server MAY copy (if present) the PARTICIPANT-PROVIDED-INFO attribute from the FloorRequest into the FLOOR-REQUEST-INFO grouped attribute.

13.1.2 Generation of Subsequent FloorRequestInfo Messages

A floor request is considered to be ongoing as long as it is not in the Cancelled, Released, or Revoked states. If the REQUEST-STATUS attribute (inside the FLOOR-REQUEST-INFORMATION grouped attribute) of the first FloorRequestInfo message generated by the floor control server did not indicate any of these states, the floor control server will need to send subsequent FloorRequestInfo messages.

When the status of the floor request changes, the floor control server SHOULD send new FloorRequestInfo messages with the appropriate Request Status. The floor control server MUST add a FLOOR-REQUEST-INFORMATION attribute with a Floor Request ID equal to the one sent in the first FloorRequestInfo message to any new FloorRequestInfo related to the same floor request. (The Floor Request ID identifies the floor request the FloorRequestInfo applies to.)

The floor control server MUST NOT add any TRANSACTION-ID attribute to subsequent FloorRequestInfo messages.

> The rate at which the floor control server sends FloorRequestInfo messages is a matter of local policy. A floor control server may choose to send a new FloorRequestInfo message every time the floor request moves in the floor request queue while another may choose to only send a new FloorRequestInfo message when the floor request is Granted or Denied.

The floor control server may add a STATUS-INFO attribute to any of the FloorRequestInfo messages it generates to provide extra information about its decisions regarding the floor request (e.g., why it was denied).

Floor participants and floor chairs may request to be informed about the status of a floor following the procedures in Section 12.1. If the processing of a floor request changes the status of a floor (e.g., the floor request is granted and consequently the floor has a new holder), the floor control server needs to follow the procedures in Section 13.5 to inform the clients that have requested that information

The FIXED-HEADER and the rest of the attributes are the same as in the first FloorRequestInfo message.

The floor control server can discard the state information about a particular floor request when this reaches a status of Cancelled, Released, or Revoked.

13.2 Reception of a FloorRequestInfoWanted Message

On reception of a FloorRequestInfoWanted message, the floor control server follows the rules in Section 9 which relate to client authentication and authorization. If while processing the FloorRequestInfoWanted message, the floor control server encounters an error, it SHOULD generate an Error response following the procedures described in Section 13.8

The successful processing of a FloorRequestInfoWanted message by a floor control server involves generating a FloorRequestInfo message, which SHOULD be generated as soon as possible.

The floor control server MUST copy the Conference ID, the TRANSACTION-ID, and the USER-ID attributes from the FloorRequestInfoWanted message into the FloorRequestInfo message, as described in Section 8.2. Additionally, the floor control server MUST add a FLOOR-REQUEST-INFORMATION grouped attribute to the FloorRequestInfo. The attributes contained in this grouped attribute carry information about the floor request.

The floor control server MUST copy the contents of the FLOOR-REQUEST-ID attribute from the FloorRequestInfoWanted message into the Floor Request ID field of the FLOOR-REQUEST-INFORMATION attribute.

The floor control server MUST add FLOOR-ID attributes to the FLOOR-REQUEST-INFORMATION grouped attribute identifying the floors being requested (i.e., the floors associated with the floor request identified by the FLOOR-REQUEST-ID attribute).

The floor control server SHOULD add a BENEFICIARY-ID attribute to the FLOOR-REQUEST-INFORMATION grouped attribute identifying the

beneficiary of the floor request. Additionally, the floor control server MAY provide the display name and the URI of the beneficiary in this BENEFICIARY-INFORMATION attribute.

The floor control server MAY provide information about the requester of the floor in a REQUESTED-BY-INFORMATION attribute inside the FLOOR-REQUEST-INFORMATION grouped attribute.

The floor control server MAY provide the reason why the floor participant requested the floor in a PARTICIPANT-PROVIDED-INFO.

The floor control server MAY also add to the FLOOR-REQUEST-INFORMATION grouped attribute a PRIORITY attribute with the Priority value requested for the floor request and a STATUS-INFO attribute with extra information about the floor request.

The floor control server adds a REQUEST-STATUS attribute to the FLOOR-REQUEST-INFORMATION grouped attribute with the current status of the floor request.

13.3 Reception of a UserInfoWanted Message

On reception of a UserInfoWanted message, the floor control server follows the rules in Section 9 which relate to client authentication and authorization. If while processing the UserInfoWanted message, the floor control server encounters an error, it SHOULD generate an Error response following the procedures described in Section 13.8

The successful processing of a UserInfoWanted message by a floor control server involves generating a UserInfo message, which SHOULD be generated as soon as possible.

The floor control server MUST copy the Conference ID, the TRANSACTION-ID, and the USER-ID attributes from the UserInfoWanted message into the USerInfo message, as described in Section 8.2.

The sender of the UserInfoWanted message is requesting information about all the floor requests associated to a given participant (i.e., the floor requests where the participant is either the beneficiary or the requester). This participant is identified by a BENEFICIARY-ID attribute or, in the absence of a BENEFICIARY-ID attribute, by a USER-ID attribute in the UserInfoWanted message.

The floor control server MUST copy, if present, the contents of the BENEFICIARY-ID attribute from the UserInfoWanted message into a BENEFICIARY-INFORMATION attribute in the UserInfo message. Additionally, the floor control server MAY provide the display name and the URI of the participant the UserInfo message provides

information on in this BENEFICIARY-INFORMATION attribute.

The floor control server SHOULD add to the UserInfo message a FLOOR-REQUEST-INFORMATION grouped attribute for each floor request related to the participant the message provides information on (i.e., the floor requests where the participant is either the beneficiary or the requester). For each FLOOR-REQUEST-INFORMATION attribute, the floor control server follows the following steps.

The floor control server MUST identity the floor request the FLOOR-REQUEST-INFORMATION attribute applies to by filling the Floor Request ID field of the FLOOR-REQUEST-INFORMATION attribute.

The floor control server MUST add FLOOR-ID attributes to the FLOOR-REQUEST-INFORMATION grouped attribute identifying the floors being requested (i.e., the floors associated with the floor request identified by the FLOOR-REQUEST-ID attribute).

The floor control server SHOULD add a BENEFICIARY-ID attribute to the FLOOR-REQUEST-INFORMATION grouped attribute identifying the beneficiary of the floor request. Additionally, the floor control server MAY provide the display name and the URI of the beneficiary in this BENEFICIARY-INFORMATION attribute.

The floor control server MAY provide information about the requester of the floor in a REQUESTED-BY-INFORMATION attribute inside the FLOOR-REQUEST-INFORMATION grouped attribute.

The floor control server MAY provide the reason why the floor participant requested the floor in a PARTICIPANT-PROVIDED-INFO.

The floor control server MAY also add to the FLOOR-REQUEST-INFORMATION grouped attribute a PRIORITY attribute with the Priority value requested for the floor request and a STATUS-INFO attribute with extra information about the floor request.

The floor control server MUST add a REQUEST-STATUS attribute to the FLOOR-REQUEST-INFORMATION grouped attribute with the current status of the floor request.

13.4 Reception of a FloorRelease Message

On reception of a FloorRelease message, the floor control server follows the rules in Section 9 which relate to client authentication and authorization. If while processing the FloorRelease message, the floor control server encounters an error, it SHOULD generate an Error response following the procedures described in Section 13.8

The successful processing of a FloorRelease message by a floor control server involves generating a FloorRequestInfo message, which SHOULD be generated as soon as possible.

The floor control server MUST copy the Conference ID, the TRANSACTION-ID, and the USER-ID attributes from the FloorRelease message into the FloorRequestInfo message, as described in Section 8.2.

The floor control server MUST add a FLOOR-REQUEST-INFORMATION grouped attribute to the FloorRequestInfo. The attributes contained in this grouped attribute carry information about the floor request.

The FloorRelease message identifies the floor request it applies to using a FLOOR-REQUEST-ID. The floor control server MUST copy the contents of the FLOOR-REQUEST-ID attribute from the FloorRelease message into the Floor Request ID field of the FLOOR-REQUEST-INFORMATION attribute.

The floor control server MUST add FLOOR-ID attributes to the FLOOR-REQUEST-INFORMATION grouped attribute identifying the floors being requested (i.e., the floors associated with the floor request identified by the FLOOR-REQUEST-ID attribute).

The floor control server SHOULD add a BENEFICIARY-ID attribute to the FLOOR-REQUEST-INFORMATION grouped attribute identifying the beneficiary of the floor request. Additionally, the floor control server MAY provide the display name and the URI of the beneficiary in this BENEFICIARY-INFORMATION attribute.

The floor control server MAY provide information about the requester of the floor in a REQUESTED-BY-INFORMATION attribute inside the FLOOR-REQUEST-INFORMATION grouped attribute.

The floor control server MAY provide the reason why the floor participant requested the floor in a PARTICIPANT-PROVIDED-INFO.

The floor control server MAY also add to the FLOOR-REQUEST-INFORMATION grouped attribute a PRIORITY attribute with the Priority value requested for the floor request and a STATUS-INFO attribute with extra information about the floor request.

The floor control server MUST add a REQUEST-STATUS attribute to the FLOOR-REQUEST-INFORMATION grouped attribute. The Request Status value SHOULD be Released, if the floor (or floors) had been previously granted, or Cancelled, if the floor (or floors) had not been previously granted.

13.5 Reception of a FloorInfoWanted Message

On reception of a FloorInfoWanted message, the floor control server follows the rules in Section 9 which relate to client authentication. If while processing the FloorRelease message, the floor control server encounters an error, it SHOULD generate an Error response following the procedures described in Section 13.8

A floor control server receiving a FloorInfoWanted message from a client SHOULD keep this client informed about the status of the floors identified by FLOOR-ID attributes in the FloorInfoWanted message. Floor Control Servers keep clients informed by using FloorInfo messages.

An individual FloorInfo message carries information about a single floor. So, when a FloorInfoWanted message requests information about more than one floor, the floor control server needs to send separate FloorInfo messages for different floors.

The information FloorInfoWanted messages carry may depend on the user requesting the information. For example, a chair may be able to receive information about pending requests while a regular user may not be authorized to do so.

13.5.1 Generation of the First FloorInfo Message

The successful processing of a FloorInfoWanted message by a floor control server involves generating one or several FloorInfo messages, the first of which SHOULD be generated as soon as possible.

The floor control server MUST copy the Conference ID, the TRANSACTION-ID, and the USER-ID attributes from the FloorInfoWanted message into the FloorInfo message, as described in Section 8.2.

If the FloorInfoWanted message did not contain any FLOOR-ID attribute, the floor control server sends the FloorInfo message without adding any additional attribute and does not send any subsequent FloorInfo message to the floor participant.

If the FloorInfoWanted message contained one or more FLOOR-ID attributes, the floor control server chooses one among them and adds this FLOOR-ID attribute to the FloorInfo message. The floor control server SHOULD add a FLOOR-REQUEST-INFORMATION grouped attribute for each floor request associated to the floor. Each FLOOR-REQUEST-INFORMATION grouped attribute contains a number of attributes which provide information about the floor request. For each FLOOR-REQUEST-INFORMATION attribute, the floor control server follows the following steps.

The floor control server MUST identity the floor request the FLOOR-REQUEST-INFORMATION attribute applies to by filling the Floor Request ID field of the FLOOR-REQUEST-INFORMATION attribute.

The floor control server MUST add FLOOR-ID attributes to the FLOOR-REQUEST-INFORMATION grouped attribute identifying the floors being requested (i.e., the floors associated with the floor request identified by the FLOOR-REQUEST-ID attribute).

The floor control server SHOULD add a BENEFICIARY-ID attribute to the FLOOR-REQUEST-INFORMATION grouped attribute identifying the beneficiary of the floor request. Additionally, the floor control server MAY provide the display name and the URI of the beneficiary in this BENEFICIARY-INFORMATION attribute.

The floor control server MAY provide information about the requester of the floor in a REQUESTED-BY-INFORMATION attribute inside the FLOOR-REQUEST-INFORMATION grouped attribute.

The floor control server MAY provide the reason why the floor participant requested the floor in a PARTICIPANT-PROVIDED-INFO.

The floor control server MAY also add to the FLOOR-REQUEST-INFORMATION grouped attribute a PRIORITY attribute with the Priority value requested for the floor request and a STATUS-INFO attribute with extra information about the floor request.

The floor control server MUST add a REQUEST-STATUS attribute to the FLOOR-REQUEST-INFORMATION grouped attribute with the current status of the floor request.

13.5.2 **Generation of Subsequent** FloorInfo Messages

If the FloorInfoWanted message carried more than one FLOOR-ID attribute, the floor control server SHOULD generate a FloorInfo message for each of them (except for the FLOOR-ID attribute chosen for the first FloorInfo message) as soon as possible. These FloorInfo messages are generated following the same rules as for the first FloorInfo message (see Section 13.5.1, but without adding a TRANSACTION attribute.

After generating these messages, the floor control server sends FloorInfo messages periodically keeping the client informed about all the floors the client requested information about. These messages MUST NOT carry a TRANSACTION-ID attribute.

The rate at which the floor control server sends FloorInfo messages is a matter of local policy. A floor control server may choose to send a new FloorInfo message every time a new floor request arrives while another may choose to only send a new FloorInfo message when a new floor request is Granted.

13.6 Reception of a ChairAction Message

On reception of a ChairAction message, the floor control server follows the rules in Section 9 which relate to client authentication and authorization. If while processing the ChairAction message, the floor control server encounters an error, it SHOULD generate an Error response following the procedures described in Section 13.8

The successful processing of a ChairAction message by a floor control server involves generating a ChairActionAck message, which SHOULD be generated as soon as possible.

The floor control server MUST copy the Conference ID, the TRANSACTION-ID, and the USER-ID attributes from the ChairAction message into the ChairActionAck message, as described in Section 8.2.

The floor control server needs to take into consideration the operation requested in the ChairAction message (e.g., granting a floor), but does not necessarily need to perform it as requested by the floor chair. The operation that the floor control server performs depends on the ChairAction message and on the internal state of the floor control server.

For example, a floor chair may send a ChairAction message granting a floor which was requested as part of an atomic floor request operation that involved several floors. Even if the chair responsible for one of the floors instructs the floor control server to grant the floor, the floor control server will not grant it until the chairs responsible for the other floors agree to grant them as well.

So, the floor control server is ultimately responsible to keep a coherent floor state using instructions from floor chairs as input to this state.

If the new Status in the ChairAction message is Accepted and all the bits of the Queue Position field are zero, the floor chair is requesting the floor control server to assign a queue position (e.g., the last in the queue) to the floor request based on the local policy of the floor control server. (Of course, such a request only applies in case the floor control server implements a queue.)

13.7 Reception of a Hello Message

On reception of a Hello message, the floor control server follows the rules in Section 9 which relate to client authentication. If while processing the Hello message, the floor control server encounters an error, it SHOULD generate an Error response following the procedures described in Section 13.8

The successful processing of a Hello message by a floor control server involves generating a HelloAck message, which SHOULD be generated as soon as possible. The floor control server MUST copy the Conference ID, the TRANSACTION-ID, and the USER-ID attributes from the Hello into the HelloAck, as described in Section 8.2.

The floor control server MUST add a SUPPORTED-PRIMITIVES attribute to the HelloAck message listing all the primitives (i.e., BFCP messages) supported by the floor control server.

The floor control server MUST add a SUPPORTED-ATTRIBUTES attribute to the HelloAck message listing all the attributes supported by the floor control server.

13.8 Error Message Generation

Error messages are always sent in response to a previous message from the client as part of a client-initiated transaction. The ABNF in Section 5.3.13 describes the attributes that an Error message can contain. In addition, the ABNF specifies normatively which of these attributes are mandatory, and which ones are optional.

The floor control server MUST copy the Conference ID, the TRANSACTION-ID, and the USER-ID attributes from the message from the client into the Error message, as described in Section 8.2.

The floor control server MUST add an ERROR-CODE attribute to the Error message. The ERROR-CODE attribute contains an Error Code from Table 6. Additionally, the floor control server may add a ERROR-INFO attribute with extra information about the error.

14. Security Considerations

BFCP can use TLS or message signatures to provide client authentication. Floor control server authentication is based on TLS, which also provides replay and integrity protection, and confidentiality. It is RECOMMENDED that TLS with non-null encryption is always used and that the first message from an unauthenticated client over a given TLS connection is signed using the DIGEST attribute. Clients and floor control servers MAY use other security

mechanisms as long as they provide similar security properties.

The remainder of this Section analyzes some of the threats against BFCP and how they are addressed.

An attacker may attempt to impersonate a client (a floor participant or a floor chair) in order to generate forged floor requests or to grant or deny existing floor requests. Client impersonation is avoided by having clients sign their messages. A nonce is included in the signature to ensure the freshness of the message. If the client is using a TLS connection to communicate with the floor control server, it is enough that the client signs its first message over the TLS connection. The floor control server assumes that attackers cannot hickjack the TLS connection and, therefore, that subsequent messages over the TLS connection come from the client that was initially authenticated. If TLS-based client authentication is used, there is not need for the client to sign BFCP messages over the connection.

An attacker may attempt to impersonate a floor control server. A successful attacker would be able to make clients think that they hold a particular floor so that they would try to access a resource (e.g., sending media) without having legitimate rights to access it. Floor control server impersonation is avoided by having floor control servers present their server certificates or prove that they have a shared secret with the client at TLS connection establishment time.

Attackers may attempt to modify messages exchanged by a client and a floor control server. The integrity protection provided by TLS connections prevents this attack.

An attacker may attempt to fetch a valid message sent by a client to a floor control server and replay it at a later point. If the message was signed, the attacker may attempt to establish a new TLS connection with the floor control server and replay the message over the new connection. Using TLS confidentiality prevents this attack because the attacker cannot access the contents of the message in the first place. Additionally, TLS provides replay protection for a given connection. Therefore, it is strongly RECOMMENDED that TLS is used with a non-null encryption algorithm.

Attackers may attempt to pick messages from the network to get access to confidential information between the floor control server and a client (e.g., why a floor request was denied). TLS confidentiality prevents this attack.

15. IANA Considerations

This document instructs the IANA to create a new registry for BFCP parameters called "Binary Floor Control Protocol (BFCP) Parameters". This new registry has a number of subregistries, which are described in the following Sections

15.1 Attribute Subregistry

This Section establishes the Attribute subregistry under the BFCP Parameters registry. As per the terminology in RFC 2434 [5], the registration policy for BFCP attributes shall be "Specification Required". For the purposes of this subregistry, the BFCP attributes for which IANA registration is requested MUST be defined by a standards-track RFC. Such RFC MUST specify the attribute's type, name, format, and semantics.

For each BFCP attribute, the IANA registers its type, its name, and the reference to the RFC where the attribute is defined. The following table contains the initial values of this subregistry.

Type	Attribute	Reference
1	BENEFICIARY-ID	[RFC XXXX]
2	FLOOR-ID	[RFC XXXX]
3	FLOOR-REQUEST-ID	[RFC XXXX]
4	NONCE	[RFC XXXX]
5	TRANSACTION-ID	[RFC XXXX]
6	USER-ID	[RFC XXXX]
7	PRIORITY	[RFC XXXX]
8	REQUEST-STATUS	[RFC XXXX]
9	DIGEST	[RFC XXXX]
10	ERROR-CODE	[RFC XXXX]
11	ERROR-INFO	[RFC XXXX]
12	PARTICIPANT-PROVIDED-INFO	[RFC XXXX]
13	STATUS-INFO	[RFC XXXX]
14	SUPPORTED-ATTRIBUTES	[RFC XXXX]
15	SUPPORTED-PRIMITIVES	[RFC XXXX]
16	USER-DISPLAY-NAME	[RFC XXXX]
17	USER-URI	[RFC XXXX]
18	BENEFICIARY-INFORMATION	[RFC XXXX]
19	FLOOR-REQUEST-INFORMATION	[RFC XXXX]
20	REQUESTED-BY-INFORMATION	[RFC XXXX]

Table 7: Initial values of the BFCP Attribute subregistry

15.2 Primitive Subregistry

This Section establishes the Primitive subregistry under the BFCP Parameters registry. As per the terminology in RFC 2434 [5], the registration policy for BFCP primitives shall be "Specification Required". For the purposes of this subregistry, the BFCP primitives for which IANA registration is requested MUST be defined by a standards-track RFC. Such RFC MUST specify the primitive's value, name, format, and semantics.

For each BFCP primitive, the IANA registers its value, its name, and the reference to the RFC where the primitive is defined. The following table contains the initial values of this subregistry.

Value	Primitive	Reference
1	FloorRequest	[RFC XXXX]
2	FloorRelease	[RFC XXXX]
3	FloorRequestInfoWanted	[RFC XXXX]
4	FloorRequestInfo	[RFC XXXX]
5	UserInfoWanted	[RFC XXXX]
6	UserInfo	[RFC XXXX]
7	FloorInfoWanted	[RFC XXXX]
8	FloorInfo	[RFC XXXX]
9	ChairAction	[RFC XXXX]
10	ChairActionAck	[RFC XXXX]
11	Hello	[RFC XXXX]
12	HelloAck	[RFC XXXX]
13	Error	[RFC XXXX]

Table 8: Initial values of the BFCP primitive subregistry

15.3 Request Status Subregistry

This Section establishes the Request Status subregistry under the BFCP Parameters registry. As per the terminology in RFC 2434 [5], the registration policy for BFCP request status shall be "Specification Required". For the purposes of this subregistry, the BFCP request status for which IANA registration is requested MUST be defined by a standards-track RFC. Such RFC MUST specify the value and the semantics of the request status.

For each BFCP request status, the IANA registers its value, its meaning, and the reference to the RFC where the request status is defined. The following table contains the initial values of this

subregistry.

Value	Status	Reference
1	Pending	[RFC XXXX]
2	Accepted	[RFC XXXX]
3	Granted	[RFC XXXX]
4	Denied	[RFC XXXX]
5	Cancelled	[RFC XXXX]
6	Released	[RFC XXXX]
7	Revoked	[RFC XXXX]

Table 9: Initial values of the Request Status subregistry

15.4 Error Code Subregistry

This Section establishes the Error Code subregistry under the BFCP Parameters registry. As per the terminology in RFC 2434 [5], the registration policy for BFCP error codes shall be "Specification Required". For the purposes of this subregistry, the BFCP error codes for which IANA registration is requested MUST be defined by a standards-track RFC. Such RFC MUST specify the value and the semantics of the error code, and any Error Specific Details that apply to it.

For each BFCP primitive, the IANA registers its value, its meaning, and the reference to the RFC where the primitive is defined. The following table contains the initial values of this subregistry.

Value	Meaning	Reference
1	Conference does not Exist	[RFC XXXX]
2	User does not Exist	[RFC XXXX]
3	DIGEST Attribute Required	[RFC XXXX]
4	Invalid Nonce	[RFC XXXX]
5	Authentication Failed	[RFC XXXX]
6	Unknown Primitive	[RFC XXXX]
7	Unknown Mandatory Attribute	[RFC XXXX]
8	Unauthorized Operation	[RFC XXXX]

	9	Invalid Floor ID	[RFC XXXX]
	10	Floor Request ID Does Not Exist	[RFC XXXX]
	11	You have Already Reached the Maximum Number of Ongoing Floor Requests for this Floor	[RFC XXXX]

Table 10: Initial Values of the Error Code subregistry

15.5 Digest Algorithm Subregistry

This Section establishes the Digest Algorithm subregistry under the BFCP Parameters registry. As per the terminology in RFC 2434 [5], the registration policy for BFCP digest algorithms shall be "Specification Required". For the purposes of this subregistry, the BFCP error codes for which IANA registration is requested MUST be defined by a standards-track RFC. Such RFC MUST specify the value and the semantics of the error code, and any Error Specific Details that apply to it.

For each BFCP digest algorithm, the IANA registers its numeric identifier, its name, and the reference to the RFC where the algorithm is defined. The following table contains the initial values of this subregistry.

Identifier	Algorithm	Reference
0	HMAC-SHA1	RFC 2104

Table 11: Initial values of the Digest Algorithms subregistry

16. Acknowledgments

The XCON WG chairs, Adam Roach and Alan Johnston, provided useful ideas for this document. Additionally, Xiaotao Wu, Paul Kyzivat, Jonathan Rosenberg, Miguel A. Garcia-Martin, Mary Barnes, Ben Campbell, and Dave Morgan provided useful comments.

17. References

17.1 Normative References

[1] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

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

[3] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[4] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[5] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[6] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.

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

17.2 Informational References

[8] 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.

[9] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[10] Schulzrinne, H., "Requirements for Floor Control Protocol", draft-ietf-xcon-floor-control-req-03 (work in progress), January 2005.

[11] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", draft-ietf-sipping-conferencing-framework-04 (work in progress), February 2005.

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

[13] Camarillo, G., "Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams", draft-ietf-mmusic-sdp-bfcp-00 (work in progress), January 2005.

Authors' Addresses

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Gonzalo.Camarillo@ericsson.com

Joerg Ott
Helsinki University of Technology
Department for Electrical and Communications Engineering
Networking Laboratory
Helsinki
Finland

Email: jo@netlab.hut.fi

Keith Drage
Lucent Technologies
Windmill Hill Business Park
Swindon
Wiltshire SN5 6PP
UK

Email: drage@lucent.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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