# Connection Establishment in the Binary Floor Control Protocol (BFCP)
## draft-ietf-xcon-bfcp-connection-02.txt

Status of this Memo

Copyright Notice

Abstract

This document specifies how a Binary Floor Control Protocol (BFCP) client establishes a connection to a BFCP floor control server outside the context of an offer/answer exchange. This document also specifies a digest authentication mechanism for BFCP based on shared secrets.

Table of Contents

## 1. Introduction

As discussed in the BFCP (Binary Floor Control Protocol) specification [9], a given BFCP 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.

Once a client obtains this information, it needs to establish a BFCP connection to the floor control server. The way this connection is established depends on the context of the client and the floor control server. How to establish such a connection in the context of an SDP [8] offer/answer [4] exchange between a client and a floor control server is specified in [10]. This document specifies how a client establishes a connection to a floor control server outside the context of an SDP offer/answer exchange.

BFCP entities establishing a connection outside an SDP offer/answer exchange need different authentication mechanisms than entities using offer/answer exchanges. This is because offer/answer exchanges provide parties with an initial integrity-protected channel that clients and floor control servers can use to exchange the fingerprints of their self-signed certificates. Outside the offer/answer model, such a channel is not typically available. This document defines a digest mechanism for BFCP that is based on shared secrets.

## 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.

## 3. TCP Connection Establishment

As stated in Section 1, a given BFCP 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. It is outside the scope of this document to specify how a client obtains this information. This document assumes that the client obtains this information using an out-of-band method.

Once the client has the transport address (i.e., IP address and port)

of the floor control server, it initiates a TCP connection towards it. That is, the client performs an active TCP open.

If the client is provided with the floor control server's host name instead of with its IP address, the client MUST perform a DNS lookup in order to resolve the host name into an IP address. Clients eventually perform an A or AAAA DNS lookup (or both) on the host name.

In order to translate the host name to the corresponding set of IP addresses, IPv6-only or dual-stack clients MUST use the newer getaddrinfo() name lookup function, instead of gethostbyname() [7]. The new function implements the Source and Destination Address Selection algorithms specified in [12], and is expected to be supported by all IPv6 hosts.

The advantage of the additional complexity is that this technique will output an ordered list of IPv6/IPv4 destination addresses based on the relative merits of the corresponding source/destination pairs. This will guarantee optimal routing. However, the Source and Destination Selection algorithms of [6] are dependent on broad operating system support and uniform implementation of the application programming interfaces that implement this behavior.

> Developers should carefully consider the issues described by Roy et al. [11] with respect to address resolution delays and address selection rules. For example, implementations of getaddrinfo() may return address lists containing IPv6 global addresses at the top of the list and IPv4 addresses at the bottom, even when the host is only configured with an IPv6 local scope (e.g., link-local) and an IPv4 address. This will, of course, introduce a delay in completing the connection.

The BFCP specification [9] describes a number of situations when the TCP connection between a client and the floor control server needs to be reestablished. However, that specification does not describe the reestablishment process because this process depends on how the connection was established in the first place.

When the existing TCP connection is closed following the rules in [9], the client SHOULD reestablish the connection towards the floor control server. If a TCP connection cannot deliver a BFCP message from the client to the floor control server and times out, the client SHOULD reestablish the TCP connection.

## 4. TLS Usage

All BFCP entities implement TLS and SHOULD use it in all their connections. TLS provides integrity and replay protection, and optional confidentiality. The floor control server MUST always act as the TLS server.

A floor control server that receives a BFCP message over TCP (no TLS) can request the use of TLS by generating an Error message with an Error code with a value of 9 (Use TLS).

## 5. Authentication

BFCP supports certificate-based mutual authentication between clients and floor control servers, as specified in Section 5.1. Additionally, BFCP also provides a digest mechanism based on a shared secret to provide client authentication for clients without certificates. This digest mechanism is described in Section 5.2.

### 5.1. Certificate-based Mutual Authentication

At TLS connection establishment, the floor control server MUST present its certificate to the client. Clients with certificates SHOULD also present their certificates to the floor control server.

The certificates provided at the TLS-level MUST either be directly signed by one of the other party's trust anchors or be validated using a certification path that terminates at one of the other party's trust anchors [5].

### 5.2. Digest-based Client Authentication

Clients without certificates can authenticate themselves to the floor control server using a digest-based mechanism instead. BFCP supports digest-based client authentication based on a shared secret between a client and the floor control server. 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. Consequently, there is a need for a mechanism to generate such a shared secret. However, such mechanism is outside the scope of this document. This document assumes that shared secrets are generated and exchanged using out-of-band means. However, shared secrets MUST be at least as long as the length of the output of the digest algorithm used, as recommended in [1].

Digest-based client authentication in BFCP is based on the DIGEST attribute, which is defined in Section 5.3.2. This attribute, which always appears as the last attribute in a message, contains an algorithm identifier and a keyed digest of the BFCP message using

that algorithm. The text used as input to the digest algorithm is the BFCP message, including the common 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 5.3.2 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 in the common header of the message.

Section 5.2.1 and Section 5.2.2 discuss how to achieve client authentication using the DIGEST attribute.

### 5.2.1. 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.

Clients can obtain the digest algorithms supported by the floor control server in an Error response from the floor control server with Error Code 10 (DIGEST Attribute Required). A client SHOULD use 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 with Error Code 10 (DIGEST Attribute Required) or 6 (Invalid Nonce). If a client generates a message without a DIGEST attribute and receives an Error response with Error Code 10 (DIGEST Attribute Required), the client SHOULD resend 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 11 (Invalid Nonce), the client SHOULD resend the message using the new nonce received in the Error response. If the Error Code is 12 (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.

### 5.2.2. 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 10 (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. Floor control servers MUST NOT use the same nonce for the same shared secret more than once.

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 10 (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 11 (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 in the common header of 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 with Error Code 12 (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 MAY choose to only authenticate the first message sent over the TLS connection. This way, the client does not need to sign every message it sends (message signatures can be long when compared with BFCP messages). Reducing the size of BFCP messages can considerably reduce

transmission times over low-bandwidth links.

### 5.3. Attribute Definitions

The following new attribute types are defined:

| Type | Attribute | Format |
|---|---|---|
| 19 | NONCE | Unsigned16 |
| 20 | DIGEST | OctetString |

Table 1: BFCP attributes

Both are EXTENSION-ATTRIBUTES as specified in [9].

#### 5.3.1. NONCE

The NONCE attribute can appear in any message. The NONCE attribute MUST be the last attribute of messages that do not contain a DIGEST attribute and the second to last attribute of messages that contain a DIGEST attribute (the DIGEST attribute is always the last). The following is the format of the NONCE attribute.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |0 0 1 0 0 1 1|M|0 0 0 0 0 1 0 0|          Nonce Value          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: NONCE format

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

#### 5.3.2. DIGEST

The DIGEST attribute can only appear in messages sent by clients. The DIGEST attribute MUST be the last attribute of the message in which it appears. The following is the format of the DIGEST attribute.

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |0 0 1 0 1 0 0|M|    Length     |   Algorithm   |               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               |
     |                                                               |
     |                                                               |
     |                                                               |
     /                           Digest                              /
     /                                                               /
     |                                                               |
     |                                                               |
     |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                               |             Padding           |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: 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 2: Digest algorithms

The text used as input to the digest algorithm is the BFCP message, including the common header, up to and including the attribute preceding the DIGEST attribute. Depending on the algorithm, this text may need to be padded with zeroes.

The key used as input to the keyed digest is the secret shared between the server and the user identified by the User ID in the common header of the message.

Digest: this field contains a keyed digest of the BFCP message. Its calculation is described in Section 5.2.

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.

## 5.4. Error Code Definitions

This specification defines the following new BFCP Error Codes:

| Value | Meaning |
|---|---|
| 10 | DIGEST Attribute Required |
| 11 | Invalid Nonce |
| 12 | Authentication Failed |

Table 3: Error Code meaning

The following is the definition of Error Specific Details for Error Code 10 (DIGEST Attribute Needed)

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Algorithm ID  | Algorithm ID  | Algorithm ID  | Algorithm ID  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               | Algorithm ID  | Algorithm ID  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Algorithm ID  | Algorithm ID  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: 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).

## 5.5. 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 challenged by the floor control server. Clients and floor control servers MAY use other security mechanisms as long as they provide similar security properties (i.e., replay and integrity protection, confidentiality, and server authentication).

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 at TLS connection establishment time. Clients MUST NOT send any signed BFCP message to an unauthenticated floor control server in order to prevent man-in-the-middle attacks.

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. The use of nonces avoids this type of attack. As stated in Section 5.2.2, floor control servers do not use the same nonce for the same shared secret more than once.

Using TLS confidentiality also prevents that attack because the attacker cannot access the contents of the message in the first place. Additionally, TLS provides replay protection within a given connection. Therefore, it is 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.

## 5.6. IANA Considerations

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

### 5.6.1. Attribute Registration

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

| Type | Attribute | Reference |
|---|---|---|
| 19 | NONCE | [RFC XXXX] |
| 20 | DIGEST | [RFC XXXX] |

Table 4: New values of the BFCP Attribute subregistry

[Note to the RFC editor: please, replace RFCxxxx with the RFC number that will be assigned to this document.]

### 5.6.2. Error Code Registration

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

| Value | Meaning | Reference |
|---|---|---|
| 10 | DIGEST Attribute Required | [RFC XXXX] |
| 11 | Invalid Nonce | [RFC XXXX] |
| 12 | Authentication Failed | [RFC XXXX] |

Table 5: New Values of the Error Code subregistry

[Note to the RFC editor: please, replace RFCxxxx with the RFC number that will be assigned to this document.]

### 5.6.3. Digest Algorithm Subregistry

This Section establishes the Digest Algorithm subregistry under the BFCP Parameters registry. As per the terminology in RFC 2434 [3], the registration policy for BFCP digest algorithms shall be "Specification Required".

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

| Identifier | Algorithm | Reference |
|---|---|---|
| 0 | HMAC-SHA1 | RFC 2104 |

Table 6: Initial values of the Digest Algorithms subregistry

## 6. Acknowledgments

Sam Hartman and Karim El Malki provided useful comments on this document.

## 7. References

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

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

[5] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[6] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[7] Shin, M-K., Hong, Y-G., Hagino, J., Savola, P., and E. Castro, "Application Aspects of IPv6 Transition", RFC 4038, March 2005.

[8] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session

Description Protocol", RFC 4566, July 2006.

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

[10] Camarillo, G., "Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams", draft-ietf-mmusic-sdp-bfcp-03 (work in progress), December 2005.

## 7.2. Informative References

[11] Roy, S., "IPv6 Neighbor Discovery On-Link Assumption Considered Harmful", draft-ietf-v6ops-onlinkassumption-04 (work in progress), January 2006.

## Author's Address

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas  02420
Finland

Email: Gonzalo.Camarillo@ericsson.com

## Intellectual Property Statement

## Disclaimer of Validity

## Copyright Statement

## Acknowledgment

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