Signaling Protocol — Client Emit Documentation

This document describes how the iOS client must interact with the signaling server via Socket.IO.

Important:
- Events "offer", "answer", and "candidate" are NOT strictly typed.
- The server does not validate or transform the payload.
- Whatever one peer sends — the other peer receives AS IS.
- Ensure the payload format matches the WebRTC engine you use.

----
1. Connecting to the Server

Clients connect to the server at:
http://localhost:3000

Connection parameters (must be passed during socket.connect()):
- roomId : Integer (required)
- username : String (required)
- isCaller : Boolean (true/false)

Example:
```
.connectParams([
"roomId": 1,
"username": username,
"isCaller": true
])
```

The server automatically:
- Joins the socket into roomId
- Notifies other peers via "room_user_joined" event

----
2. Client → Server Events (emits)

All events are forwarded to other peers in the same room.
The server does NOT modify the payload.

2.1 offer

Used by the peer initiating the call.

Event name:
offer

Payload:
Any JSON object representing the WebRTC SDP offer.

Example:

```
{
"type": "offer",
"sdp": "SDP_OFFER_STRING"
}
```

Notes:
- The server forwards this payload directly to other peers via "offer".
- No validation or transformation occurs.

2.2 answer

Used by the peer responding to an offer.

Event name:
answer

Payload:
Any JSON object representing the WebRTC SDP answer.

Example:
```
{
"type": "answer",
"sdp": "SDP_ANSWER_STRING"
}
```

Notes:
- This is passed through unchanged.

2.3 candidate

Used by both peers to exchange ICE candidates.

Event name:
candidate

Payload:
Any JSON object representing a WebRTC ICE candidate.

Example:
```
{
"candidate": "candidate:...",
"sdpMid": "0",
"sdpMLineIndex": 0
}
```

Notes:
- The server forwards the candidate as received.
- WebRTC engines must be compatible with the format you emit.

----
3. Server → Client Events (listeners)

3.1 room_user_joined
Payload:
```
{
"username": "Bob",
"roomId": 1,
"isCaller": false
}
```

3.2 offer
Payload: The exact object emitted by the remote peer.

3.3 answer
Payload: The exact object emitted by the remote peer.

3.4 candidate
Payload: The exact object emitted by the remote peer.

3.5 room_user_left
Payload:
```
{
"username": "Bob"
}
```

----
4. Summary

- offer/answer/candidate are completely untyped.
- The signaling server does not enforce structure.
- Peers must agree on a matching JSON shape.
- The client must implement WebRTC logic to interpret the received objects.