iOS Video Chat

Example of video chat client for iOS

This example can be used to participate in video conference for two participants on Web Call Server and allows to publish WebRTC stream.

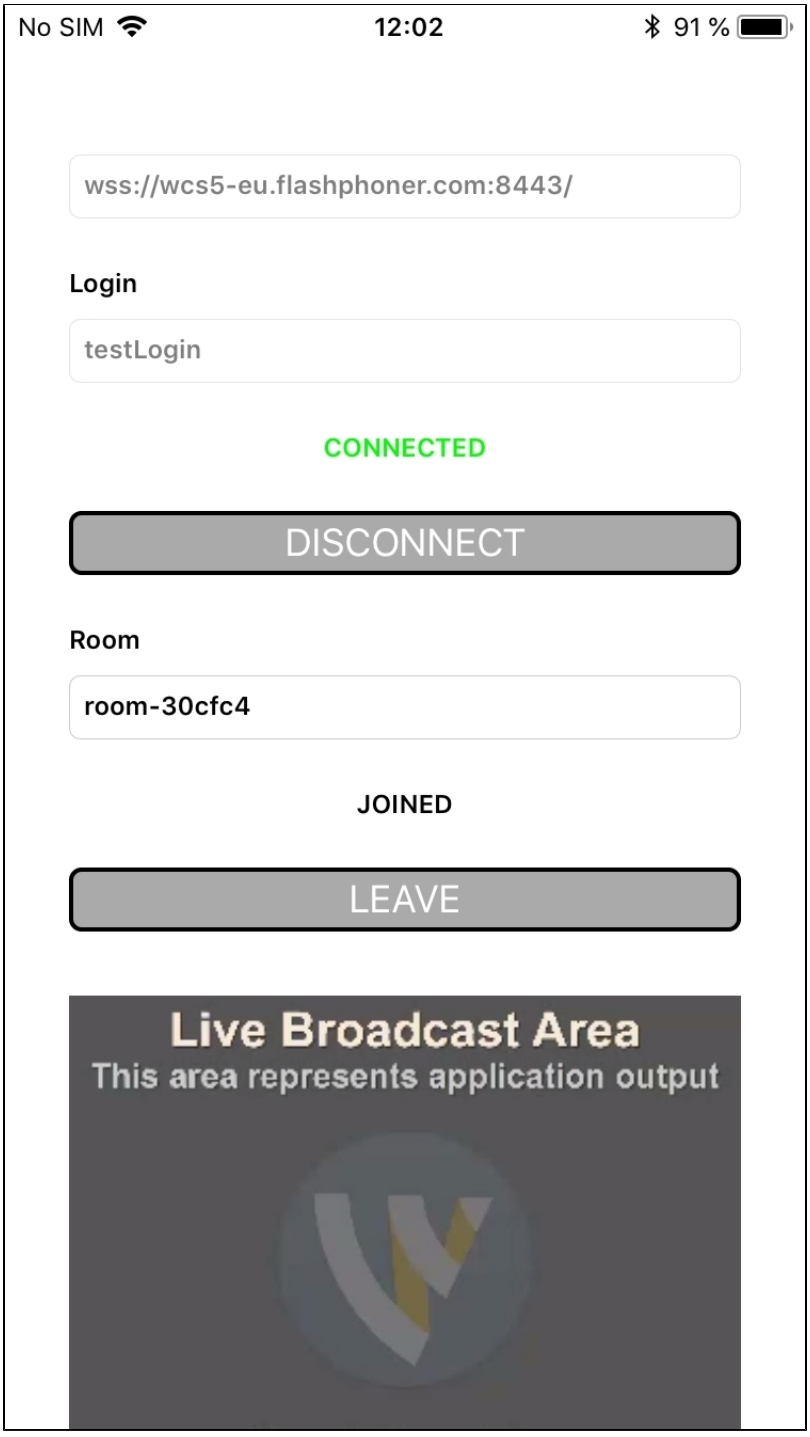
On the screenshot below the participant is connected, publishing a stream and playing streams from the other participant.

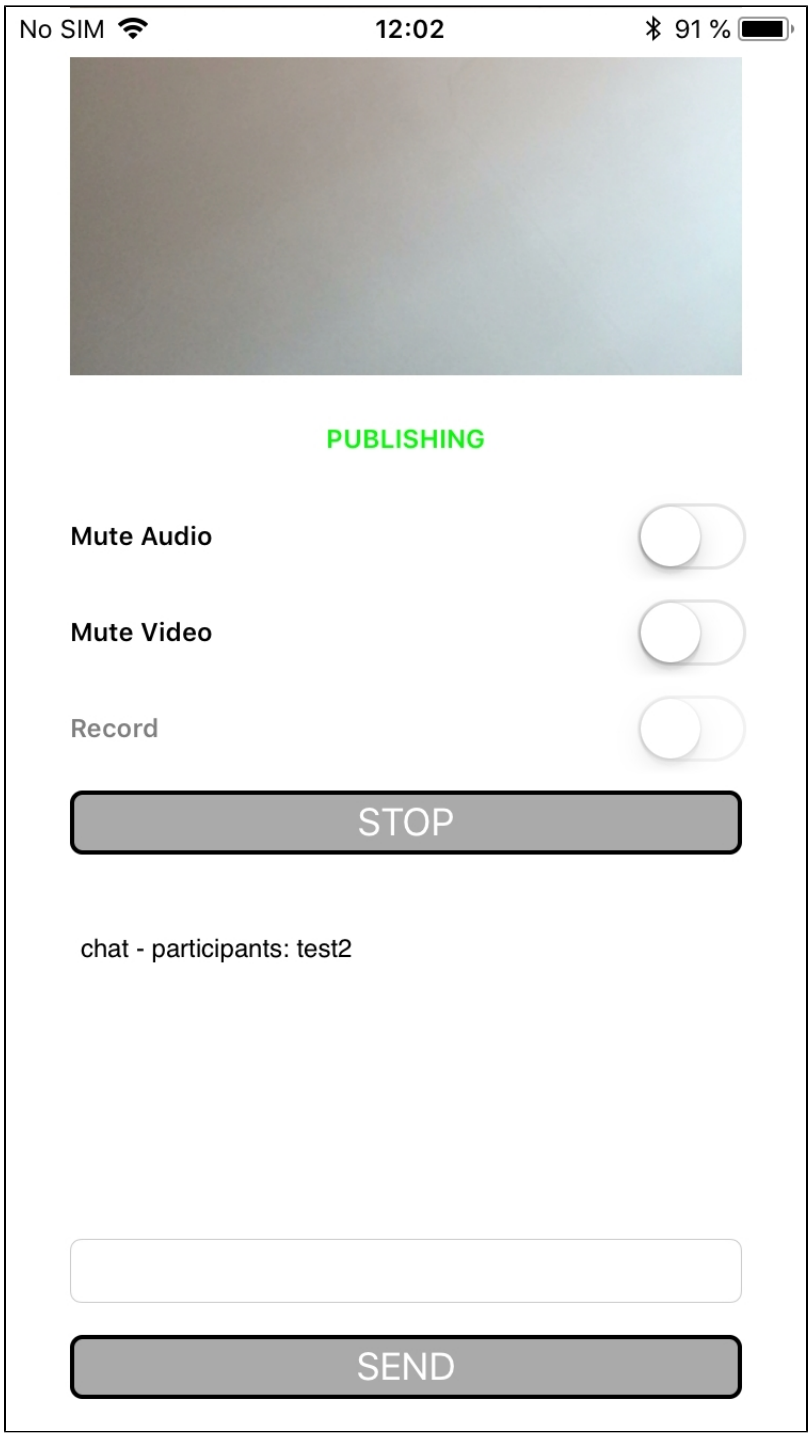
Input fields required for connecting to WCS server and joining conference

- 'WCS URL' is the address of the WCS server
- 'Login' is the username
- 'Room' is the name of conference room

Two videos are played

- video from the camera of this participant - the lower one
- video from the other participant


No SIM
12:02
91 %
wss://wcs5-eu.flashphoner.com:8443/
Login
testLogin
CONNECTED
DISCONNECT
Room
room-30cfc4
JOINED
LEAVE
Live Broadcast Area
This area represents application output



Analyzing the code of the example

To analyze the code, let's takeVideo Chat example, which can be downloaded with corresponding build2.5.2.

class for the main view of the application: ViewController (header fileViewController.h; implementation fileViewController.m).

1. Import of API.code

```
#import <FPWCSApi2/FPWCSApi2.h>
```

2. Connection to the server.

FPWCSApi2 createRoomManagercode

FPWCSApi2RoomManagerOptions object with the following parameters is passed to createRoomManager() method

- URL of WCS server
- username

```
- (void)connect {
    FPWCSApi2RoomManagerOptions *options = [[FPWCSApi2RoomManagerOptions alloc] init];
    options.urlServer = _connectUrl.text;
    options.username = _connectLogin.input.text;
    NSError *error;
    roomManager = [FPWCSApi2 createRoomManager:options error:&error];
    ...
}
```

3. Joining a conference.

FPWCSApi2RoomManager joincode

FPWCSApi2RoomOptions objectwith the name of the conference room is passed to the join() method

```
FPWCSApi2RoomOptions * options = [[FPWCSApi2RoomOptions alloc] init];
options.name = _joinRoomName.input.text;
room = [roomManager join:options];
```

4.Receiving the eventdescribing chat room state

FPWCSApi2Room onStateCallbackcode

On this event:

- the size of the collection of Participant objects returned by method getParticipants() is determined to get the number of already connected participants
- if the maximum allowed number of participants had already been reached, the user leaves the "room"
- otherwise, appropriate changes in the interface are done and playback of video stream published by the other participants is started

```
[room onStateCallback:^(FPWCSApi2Room *room) {
    NSDictionary *participants = [room getParticipants];
    if ([participants count] >= 2) {
        [room leave:nil];
        _joinStatus.text = @"Room is full";
        [self changeViewState:_joinButton enabled:YES];
        return;

    }
    NSString *chatState = @"participants: ";
    for (NSString* key in participants) {
        FPWCSApi2RoomParticipant *participant = [participants valueForKey:key];
        ParticipantView *pv = [freeViews pop];
        [busyViews setValue:pv forKey:[participant getName]];
        [participant play:pv.display];
        pv.login.text = [participant getName];
        chatState = [NSString stringWithFormat:@"%@%@, ", chatState, [participant getName]];
    }
    ...
}];
```

5. Video stream publishing.

FPWCSApi2Room publishcode

The next stream options is passed to publish method:

- view to display video
- record is set to 'true' to record stream published

```
- (void)publishButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        [room unpublish];
    } else {
        FPWCSApi2StreamOptions * options = [[FPWCSApi2StreamOptions alloc] init];
        options.record = [_record.control isOn];
        publishStream = [room publish:_localDisplay withOptions:options];
        ...
    }
}
```

6.Receiving the event notifying that other participant joined to the room

FPWCSApi2Room kFPWCSRoomParticipantEventJoined participantCallbackcode

```
[room on:kFPWCSRoomParticipantEventJoined participantCallback:^(FPWCSApi2Room *room, FPWCSApi2RoomParticipant
*participant) {
    ParticipantView *pv = [freeViews pop];
    if (pv) {
        pv.login.text = [participant getName];
        _messageHistory.text = [NSString stringWithFormat:@"%@\n%@ - %@", _messageHistory.text, participant.
getName, @"joined"];
        [busyViews setValue:pv forKey:[participant getName]];
    }
}];
```

7. Receiving the event notifying that other participant published video stream.

FPWCSApi2Room kFPWCSRoomParticipantEventPublished participantCallback, FPWCSApi2RoomParticipant playcode

On this event, video stream from other participant playback is started

```
[room on:kFPWCSRoomParticipantEventPublished participantCallback:^(FPWCSApi2Room *room,
FPWCSApi2RoomParticipant *participant) {
    ParticipantView *pv = [busyViews valueForKey:[participant getName]];
    if (pv) {
        [participant play:pv.display];
    }
}];
```

8. Receiving the event notifying that other participant sent a message.

FPWCSApi2Room onMessageCallbackcode

```
[room onMessageCallback:^(FPWCSApi2Room *room, FPWCSApi2RoomMessage *message) {
    _messageHistory.text = [NSString stringWithFormat:@"%@\n%@ - %@", _messageHistory.text, message.from,
message.text];
}];
```

9. Sending a message to other room participant.

FPWCSApi2RoomParticipant sendMessagecode

The message text is passed to the metod.

```
- (void)sendButton:(UIButton *)button {
    for (NSString *name in [room getParticipants]) {
        FPWCSApi2RoomParticipant *participant = [room getParticipants][name];
        [participant sendMessage:_messageBody.text];
    }
    _messageHistory.text = [NSString stringWithFormat:@"%@\n%@ - %@", _messageHistory.text, _connectLogin.input.
text, _messageBody.text];
    _messageBody.text = @"";
}
```

10. Mute/unmute audio and video for stream published.

FPWCSApi2Stream unmuteAudio, muteAudio, unmuteVideo, muteVideocode

```
- (void)muteAudioChanged:(id)sender {
    if (publishStream) {
        if (_muteAudio.control.isOn) {
            [publishStream muteAudio];
        } else {
            [publishStream unmuteAudio];
        }
    }
}

- (void)muteVideoChanged:(id)sender {
    if (publishStream) {
        if (_muteVideo.control.isOn) {
            [publishStream muteVideo];
        } else {
            [publishStream unmuteVideo];
        }
    }
}
```

11. Stop stream publishing.

FPWCSApi2Room unpublishcode

```
- (void)publishButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        [room unpublish];
    } else {
        ...
    }
}
```

12. Leaving chat room.

FPWCSApi2Room leaveкод

Server REST app response handler function is passed to the method.

```
if ([button.titleLabel.text isEqualToString:@"LEAVE"]) {
    if (room) {
        FPWCSApi2DataHandler *handler = [[FPWCSApi2DataHandler alloc] init];
        handler.onAccepted = ^(FPWCSApi2Session *session, FPWCSApi2Data *data){
            [self onUnpublished];
            [self onLeaved];
        };
        handler.onRejected = ^(FPWCSApi2Session *session, FPWCSApi2Data *data){
            [self onUnpublished];
            [self onLeaved];
        };
        [room leave:handler];
        room = nil;
    }
}
```

13. Disconnection.

FPWCSApi2RoomManager disconnectcode

```
- (void)connectButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"DISCONNECT"]) {
        if (roomManager) {
            [roomManager disconnect];
        }
        ...
    }
}
```