iOS Two-way Streaming

Example of iOS application with player and streamer

This streamer can be used to publish WebRTC video stream and play any of the following types of streams on Web Call Server

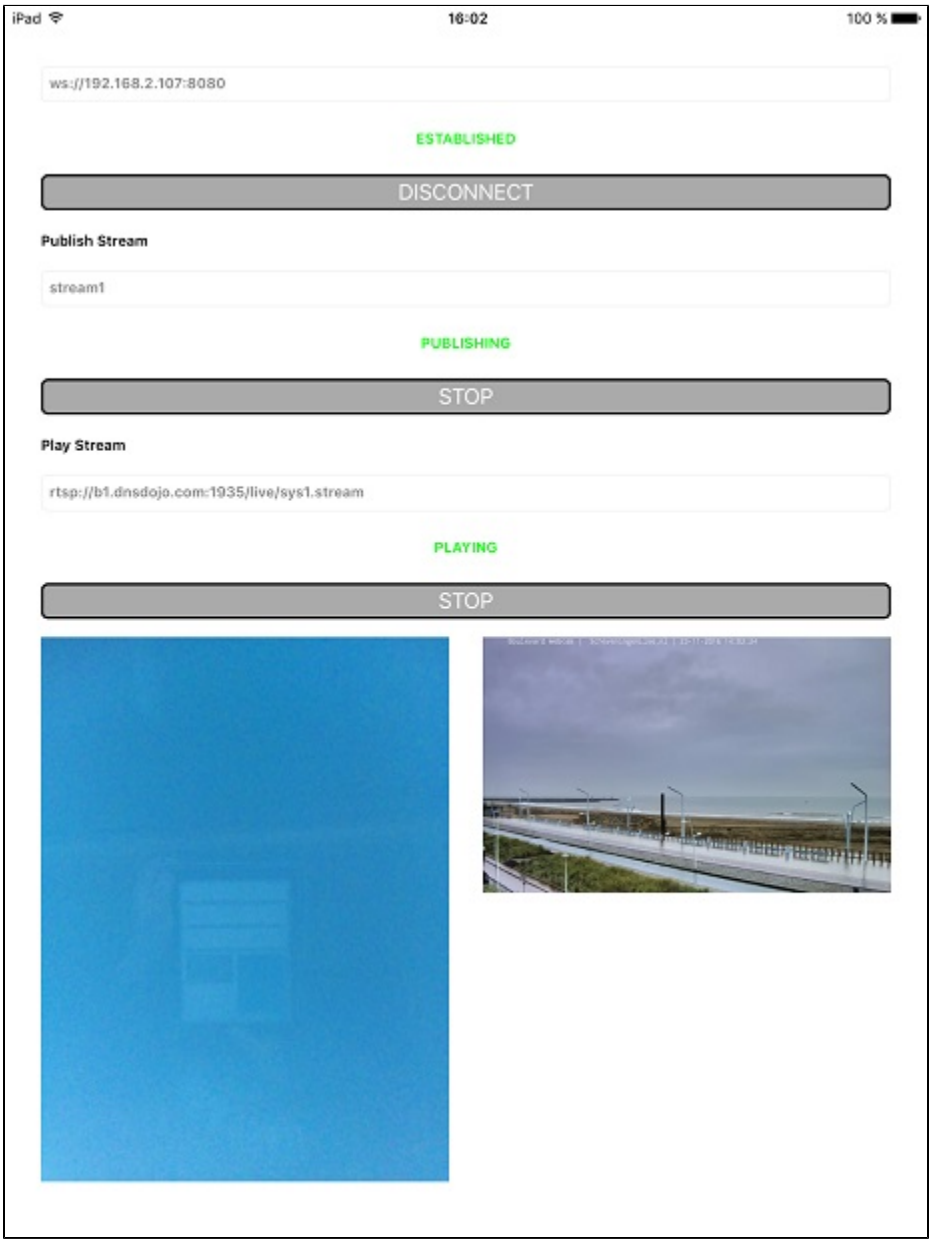
- RTSP
- WebRTC
- RTMP
- RTMFP

On the screenshot below the example is displayed when a stream is being published and another stream is being played.
Input fields

- 'WCS URL', where 192.168.2.107 is the address of the WCS server
- 'Publish Stream' - for the name of published stream
- 'Play Stream' - for the name of played stream

Two videos are played

- left - video from the camera
- right - the played video stream



Work with code of the example

To analyze the code, let's take TwoWayStreaming example, which can be downloaded with corresponding build 2.5.2.

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

1. Import of API.code

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

2. Session creation and connection to server.

FPWCSApi2 createSession, FPWCSApi2Session connectcode

The options include:

- URL of WCS server
- appKey of internal server-side application (defaultApp)

```
- (FPWCSApi2Session *)connect {
    FPWCSApi2SessionOptions *options = [[FPWCSApi2SessionOptions alloc] init];
    options.urlServer = _connectUrl.text;
    options.appKey = @"defaultApp";
    NSError *error;
    FPWCSApi2Session *session = [FPWCSApi2 createSession:options error:&error];
    ...
    [session connect];
    return session;
}
```

3. Stream publishing.

FPWCSApi2Session createStream, FPWCSApi2Stream publishcode

Object with next stream options is passed to createStream method:

- stream name
- view to display video
- 'true' for parameter 'record' to enable stream recording
- video constraints for iPad

```
- (FPWCSApi2Stream *)publishStream {
    FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
    FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
    options.name = _localStreamName.text;
    options.display = _localDisplay;
    if ( UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad ) {
        options.constraints = [[FPWCSApi2MediaConstraints alloc] initWithAudio:YES videoWidth:640 videoHeight:
480 videoFps:15];
    }
    NSError *error;
    FPWCSApi2Stream *stream = [session createStream:options error:&error];
    ...
    if(![stream publish:&error]) {
        UIAlertController * alert = [UIAlertController
                                     alertControllerWithTitle:@"Failed to publish"
                                     message:error.localizedDescription
                                     preferredStyle:UIAlertControllerStyleAlert];

        UIAlertAction* okButton = [UIAlertAction
                                   actionWithTitle:@"Ok"
                                   style:UIAlertActionStyleDefault
                                   handler:^(UIAlertAction * action) {
                                       [self onUnpublished];
                                   }];

        [alert addAction:okButton];
        [self presentViewController:alert animated:YES completion:nil];
    }
    return stream;
}
```

4.Switching camera while publishing stream

FPWCSApi2Stream switchCameracode

```
- (void)switchCameraButton:(UIButton *)button {
    if ([FPWCSApi2 getSessions].count) {
        FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
        NSArray *streams = [session getStreams];
        for (FPWCSApi2Stream *stream in streams ) {
            if ([stream isPublished]) {
                NSLog(@"Found published stream, switching camera");
                [stream switchCamera];
            }
        }
    } else {
        NSLog(@"No active sessions found");
    }
}
```

5. Stream playback.

FPWCSApi2Session createStream, FPWCSApi2Stream playcode

Object with next stream options is passed to createStream method:

- stream name
- view to display video

```
- (FPWCSApi2Stream *)playStream {
    FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
    FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
    options.name = _remoteStreamName.text;
    options.display = _remoteDisplay;
    NSError *error;
    FPWCSApi2Stream *stream = [session createStream:options error:nil];
    ...
    if(![stream play:&error]) {
        UIAlertController * alert = [UIAlertController
                                     alertControllerWithTitle:@"Failed to play"
                                     message:error.localizedDescription
                                     preferredStyle:UIAlertControllerStyleAlert];

        UIAlertAction* okButton = [UIAlertAction
                                   actionWithTitle:@"Ok"
                                   style:UIAlertActionStyleDefault
                                   handler:^(UIAlertAction * action) {

                                   }];

        [alert addAction:okButton];
        [self presentViewController:alert animated:YES completion:nil];
    }
    return stream;
}
```

6. Stop of stream playback.

FPWCSApi2Stream stopcode

```
- (void)playButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        if ([FPWCSApi2 getSessions].count) {
            FPWCSApi2Stream *stream;
            for (FPWCSApi2Stream *s in [[FPWCSApi2 getSessions][0] getStreams]) {
                if ([[s getName] isEqualToString:_remoteStreamName.text]) {
                    stream = s;
                    break;
                }
            }
            if (!stream) {
                NSLog(@"Stop playing, nothing to stop");
                [self onStopped];
                return;
            }
            NSError *error;
            [stream stop:&error];
        } else {
            NSLog(@"Stop playing, no session");
            [self onStopped];
        }
        ...
    }
}
```

7. Stop of stream publishing.

FPWCSApi2Stream stopcode

```
- (void)publishButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        if ([FPWCSApi2 getSessions].count) {
            FPWCSApi2Stream *stream;
            for (FPWCSApi2Stream *s in [[FPWCSApi2 getSessions][0] getStreams]) {
                if ([[s getName] isEqualToString:_localStreamName.text]) {
                    stream = s;
                    break;
                }
            }
            if (!stream) {
                NSLog(@"Stop publishing, nothing to stop");
                [self onUnpublished];
                return;
            }
            NSError *error;
            [stream stop:&error];
        } else {
            NSLog(@"Stop publishing, no session");
            [self onUnpublished];
        }
        ...
    }
}
```

8. Disconnection.

FPWCSApi2Session disconnectcode

```
- (void)connectButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"DISCONNECT"]) {
        if ([FPWCSApi2 getSessions].count) {
            FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
            NSLog(@"Disconnect session with server %@", [session getServerUrl]);
            [session disconnect];
        } else {
            NSLog(@"Nothing to disconnect");
            [self onDisconnected];
        }
        ...
    }
}
```