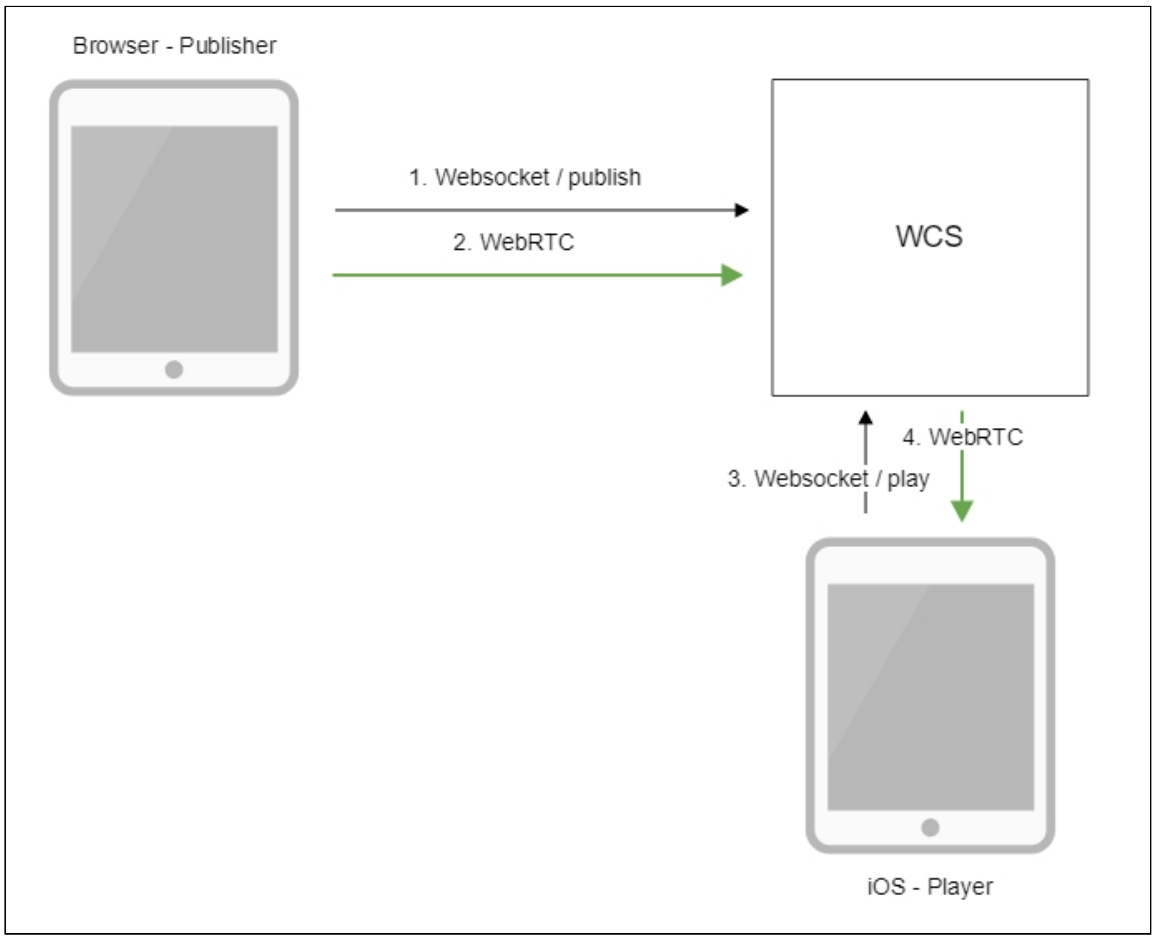
В мобильном приложении iOS по WebRTC

Описание

WCS предоставляет SDK для разработки клиентских приложений на платформе iOS

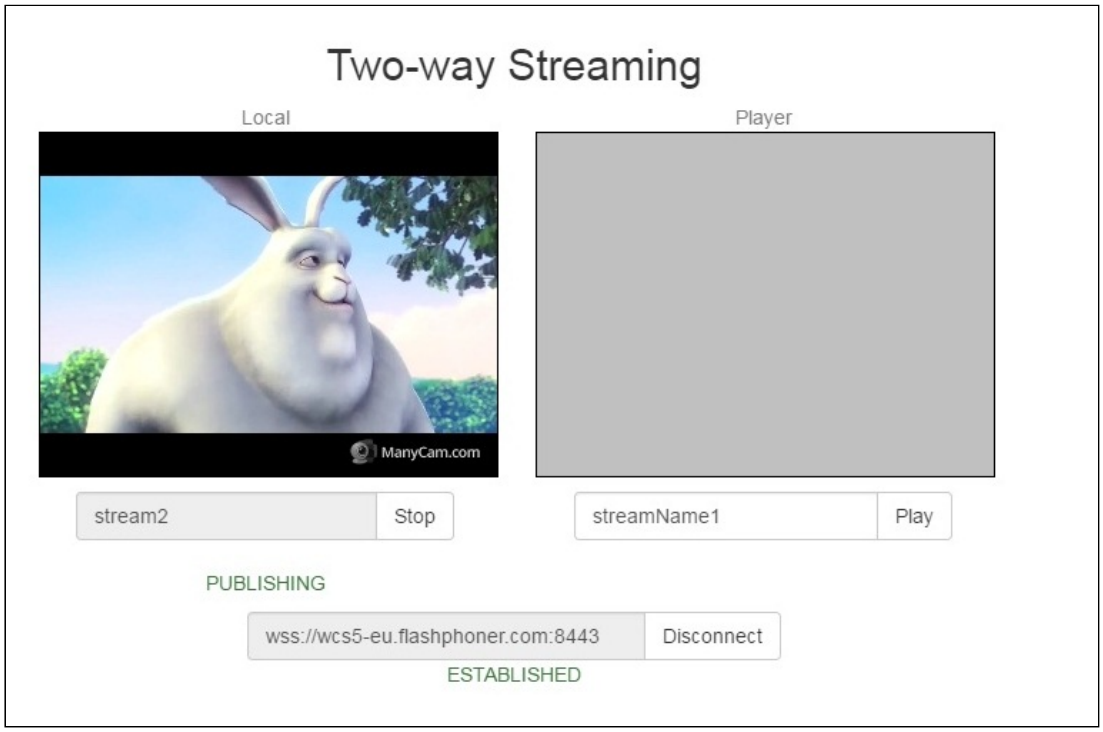
Схема работы



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду `publishStream`.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток на сервер.
3. iOS-устройство устанавливает соединение по Websocket и отправляет команду `playStream`.
4. iOS-устройство получает WebRTC поток и воспроизводит этот поток в приложении.

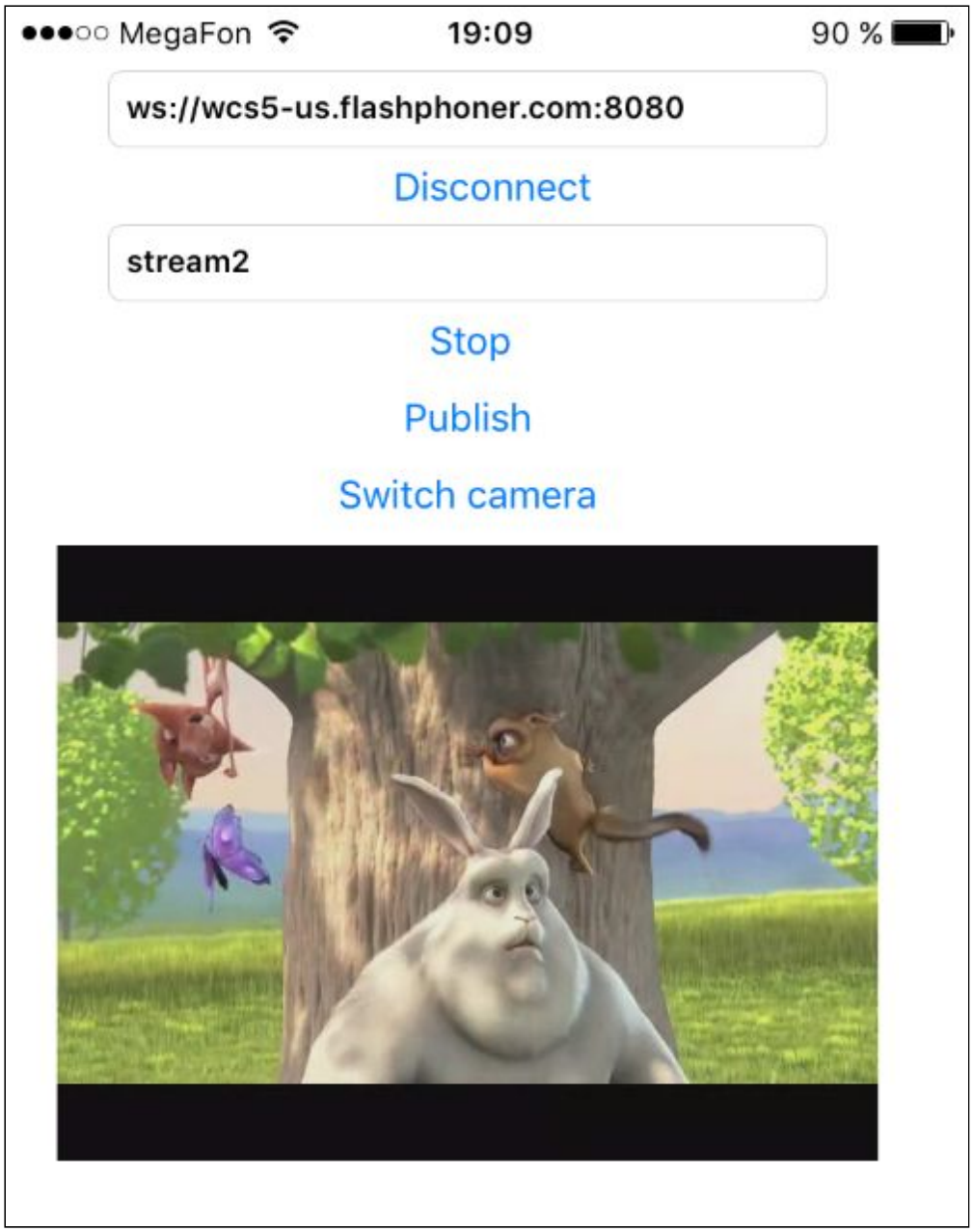
Краткое руководство по тестированию

1. Для теста используем:
2. демо-сервер `demo.flashphoner.com`;
3. веб-приложение Two Way Streaming для публикации потока;
4. мобильное приложение iOS из AppStore для воспроизведения потока
5. Откройте веб-приложение Two Way Streaming. Нажмите `Connect`, затем `Publish`. Скопируйте идентификатор потока:



6. Установите iOS приложение из AppStore на iPhone/ Запустите приложение. Введите URL WCS-сервера и имя опубликованного потока, нажмите `Play`. Начнется

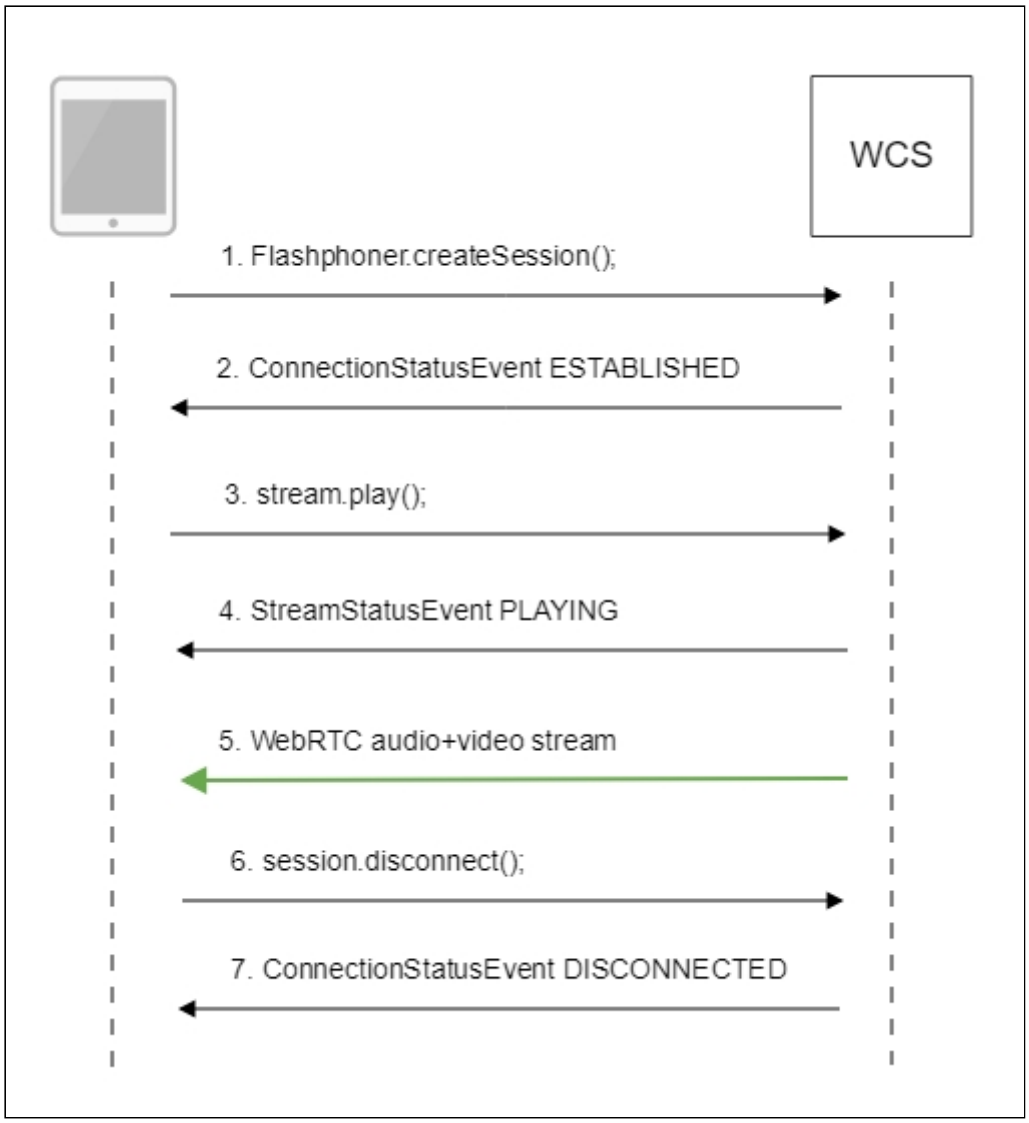
воспроизведение потока с сервера:



Последовательность выполнения операций

Ниже описана последовательность вызовов при использовании примера Player

ViewController.m



1. Установка соединения с сервером
   `FPWCSApi2.createSession()` code

```
FPWCSApi2SessionOptions *options = [[FPWCSApi2SessionOptions alloc] init];
options.urlServer = _connectUrl.text;
options.appKey = @"defaultApp";
NSError *error;
FPWCSApi2Session *session = [FPWCSApi2 createSession:options
error:&error];
```

2. Получение от сервера события, подтверждающего успешное соединение
   `FPWCSApi2Session.on:kFPWCSSessionStatusEstablished callback` code

```
[session on:kFPWCSSessionStatusEstablished callback:^(FPWCSApi2Session
*rSession){
    [self changeConnectionStatus:[rSession getStatus]];
    [self onConnected:rSession];
}];
```

3. Запуск воспроизведения потока
`FPWCSApi2Session.createStream()` code

```
- (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) {
        ...
        return nil;
    }
}
```

4. Получение от сервера события, подтверждающего успешное воспроизведение потока
`FPWCSApi2Stream.on:kFPWCSStreamStatusPlaying callback` code

```
[stream on:kFPWCSStreamStatusPlaying callback:^(FPWCSApi2Stream *rStream){
    [self changeStreamStatus:rStream];
    [self onPlaying:rStream];
}];
```

5. Прием аудио-видео потока по WebRTC

6. Остановка воспроизведения потока
`FPWCSApi2Session.disconnect()` code

```
if ([button.titleLabel.text isEqualToString:@"STOP"]) {
    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];
    }
    ...
}
```

7. Получение от сервера события, подтверждающего остановку воспроизведения потока
`FPWCSApi2Session.on:kFPWCSSessionStatusDisconnected callback` code

```
[session on:kFPWCSSessionStatusDisconnected callback:^(FPWCSApi2Session
*rSession){
    [self changeConnectionStatus:[rSession getStatus]];
    [self onDisconnected];
}];
```