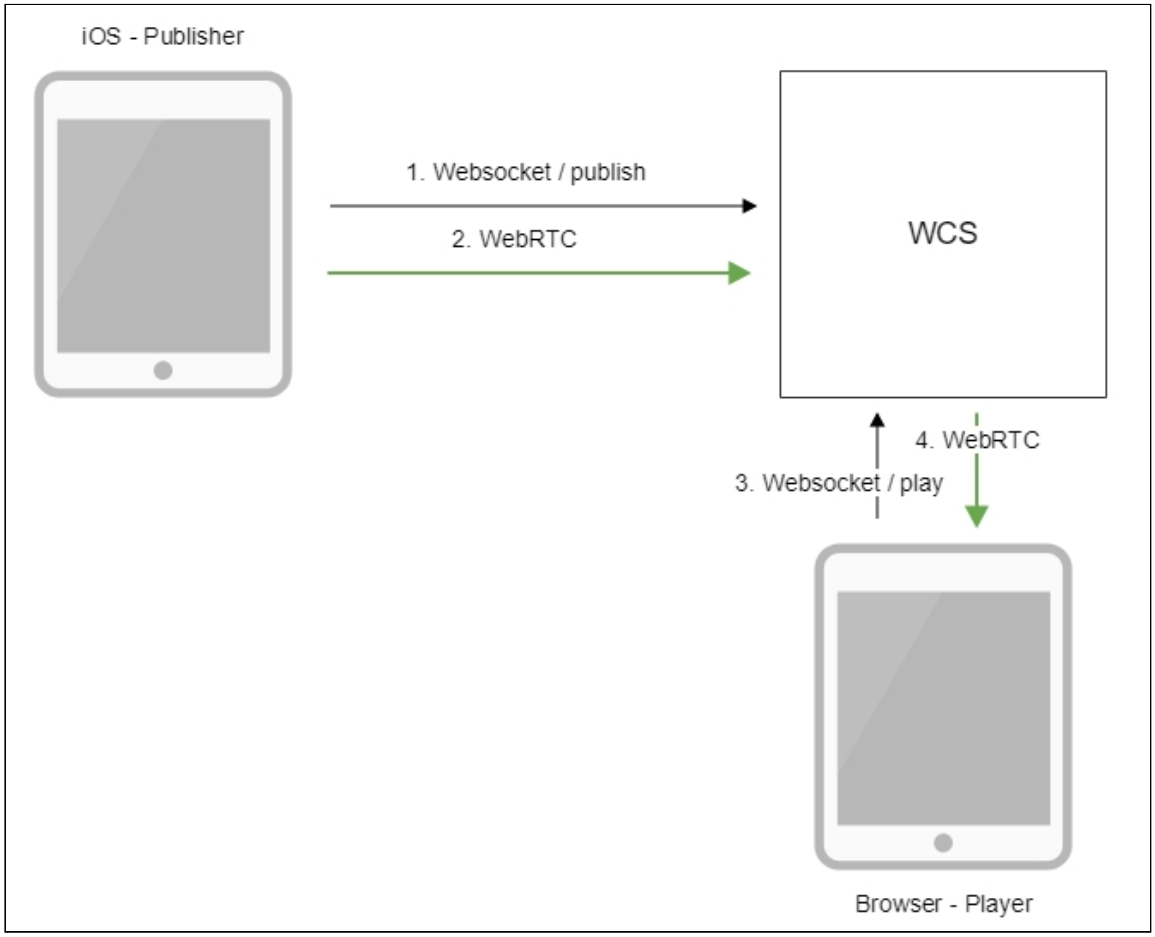
С мобильного приложения iOS по WebRTC

Описание

WCS предоставляет SDK для разработки клиентских приложений на платформе iOS

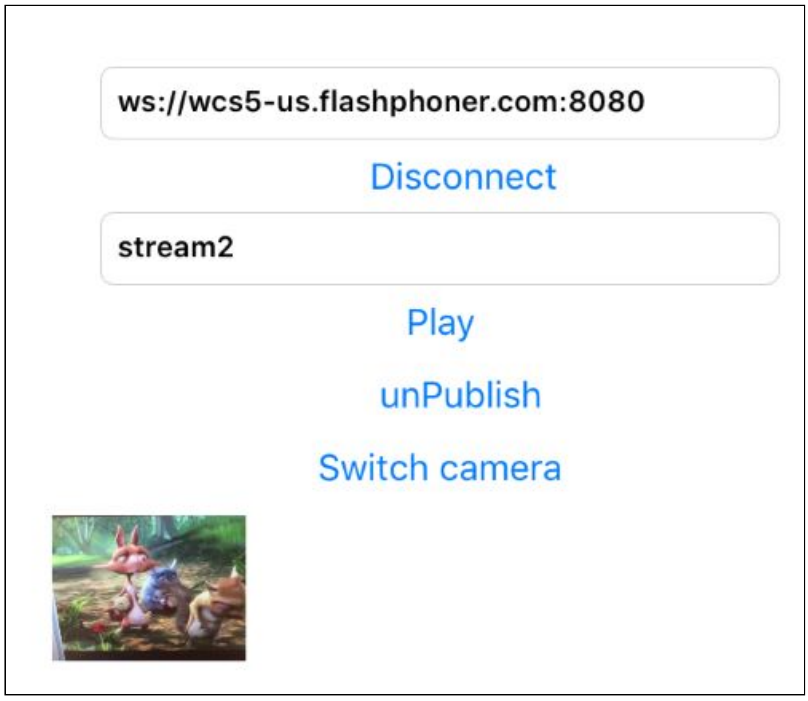
Схема работы



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

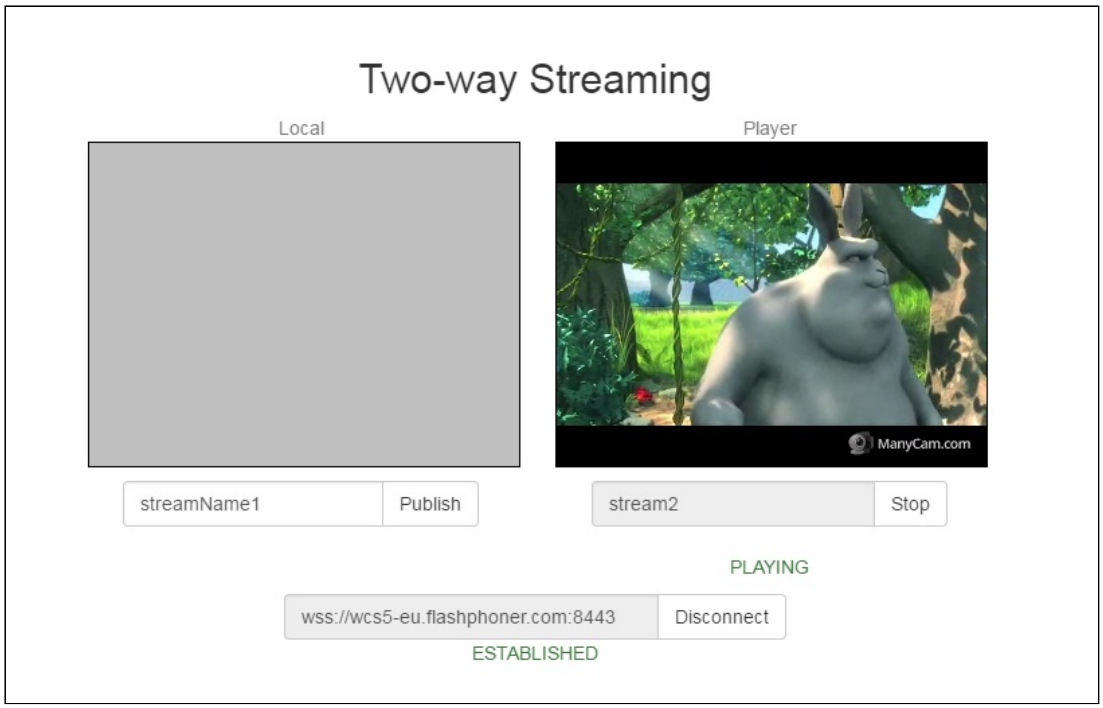
Краткое руководство по тестированию

1. Для теста используем:
2. демо-сервер `wcs5-us.flashphoner.com`;
3. мобильное приложение iOS;
4. веб-приложение Two Way Streaming для отображения захваченного потока
5. Запустите приложение на iPhone. Введите URL WCS-сервера и имя потока. Опубликуйте поток, нажав `Publish`:



6. Откройте веб-приложение Two Way Streaming. Укажите в поле под окном плеера имя потока, транслируемого с iPhone, и нажмите кнопку `Play`. Браузер начнет

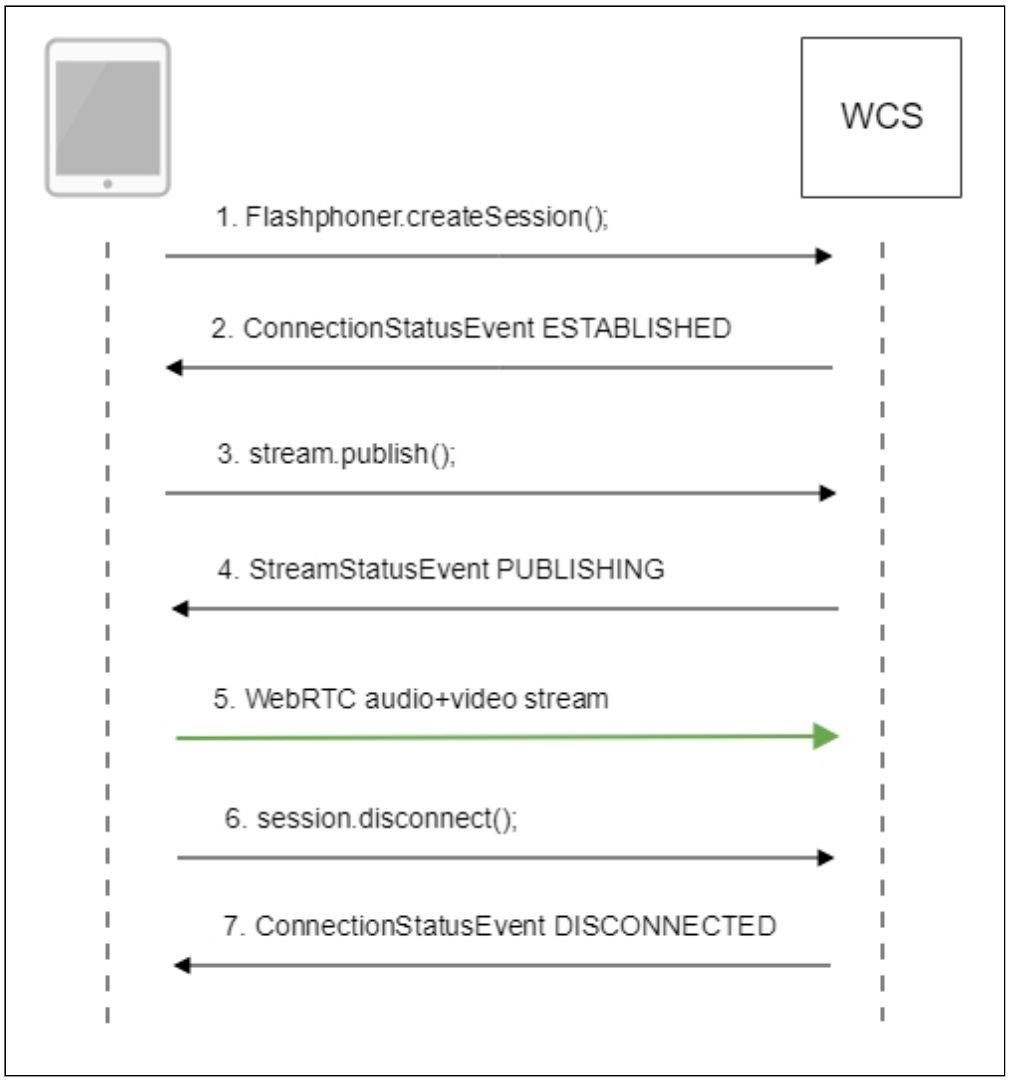
воспроизведение потока:



Последовательность выполнения операций

Ниже описана последовательность вызовов при использовании примера Streamer

ViewController.m



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

```
FPWCSApi2SessionOptions *options = [[FPWCSApi2SessionOptions alloc] init];
NSURL *url =[[NSURL alloc] initWithString:_connectUrl.text];
options.urlServer = [NSString stringWithFormat:@"%@://%@:%@", url.scheme,
url.host, url.port];
streamName = [url.path.stringByDeletingPathExtension
stringByReplacingOccurrencesOfString: @"/" withString:@""];
options.appKey = @"defaultApp";
NSError *error;
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. Публикация потока

`FPWCSApi2Session.createStream()` code

```
FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
options.name = streamName;
options.display = _videoView.local;
if ( UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad ) {
  options.constraints = [[FPWCSApi2MediaConstraints alloc]
initWithAudio:YES videoWidth:640 videoHeight:480 videoFps:15];
}
NSError *error;
publishStream = [session createStream:options error:&error];
```

4. Получение от сервера события, подтверждающего успешную публикацию потока

`FPWCSApi2Stream.on:kFPWCSStreamStatusPublishing callback` code

```
[publishStream on:kFPWCSStreamStatusPublishing callback:^(FPWCSApi2Stream
*rStream){
  [self changeStreamStatus:rStream];
  [self onPublishing:rStream];
}];
```

5. Отправка аудио-видео потока по WebRTC

6. Остановка публикации 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];
  }
} else {
  //todo check url is not empty
  [self changeViewState:_connectUrl enabled:NO];
  [self connect];
}
```

7. Получение от сервера события, подтверждающего остановку публикации потока

`FPWCSApi2Sessionon:kFPWCSSessionStatusDisconnected callback` code

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

```
    [self onDisconnected];
  }];
```