

Video Conference

- [Пример видеоконференции](#)
- [Код примера](#)
- [Работа с кодом примера](#)

Пример видеоконференции

Данный пример может использоваться для организации видеоконференции для трех участников на Web Call Server
Участник видеоконференции может публиковать WebRTC поток

На скриншоте ниже представлен пример клиента участника конференции, к которой присоединились два других участника.

Conference

WCS URL

ws://localhost:8080

Login

Alice

Leave

Record

ESTABLISHED



Bob

Stop

Audio

Unmuted



Cindy

Stop

Audio

Unmuted



PUBLISHING

Mute A

Mute V

Stop

9:24 chat - room is empty

9:25 Bob - joined

9:25 Cindy - joined

Send

Invite

<http://localhost:8081/client2/examples/demo/streaming/conference/conference.html?roomName=room-955abd>

На странице воспроизводятся три видео

- нижнее - видео с камеры данного участника (Alice)
- два верхних - видео от других двух участников (Bob и Cindy)

Код примера

Код данного примера находится на WCS-сервере по следующему пути:

```
/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/conference
```

conference.css - файл стилей
conference.html - страница участника конференции
conference.js - скрипт, обеспечивающий работу конференции

Тестировать данный пример можно по следующему адресу:

```
https://host:8888/client2/examples/demo/streaming/conference/conference.html
```

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла conference.js с хешем 90771d4, которая находится [здесь](#) и доступна для скачивания в соответствующей сборке [2.0.218](#).

Скрипт конференции использует RoomApi, предназначенное для видеочатов, конференций, вебинаров и других приложений, которые предполагают нахождение пользователей в одной виртуальной "комнате". Для того, чтобы использовать RoomApi, необходимо подключить скрипт flashphoner-room-api.js

```
<script type="text/javascript" src="../../../../flashphoner-room-api.js"></script>
```

При этом для обращения к стандартным методам Flashphoner необходимо использовать объект RoomApi.sdk

```
var Flashphoner = RoomApi.sdk;  
...  
Flashphoner.init();
```

При подключении пользователя к конференции, используется метод RoomApi.connect(), в отличие от прямого подключения к серверу методом createSession().

При присоединении к новой "комнате" методом connection.join(), создается объект room для работы с этой "комнатой". Для работы с участниками конференции используются объекты Participant.

Все события, происходящие в "комнате" (присоединение/выход пользователя, отправленные сообщения), транслируются другим участникам, подключенным к этой "комнате". Например, в следующем коде подключаемся к "комнате" и запрашиваем список других участников:

```
connection.join({name: getRoomName()}).on(ROOM_EVENT.STATE, function(room) {  
  var participants = room.getParticipants();  
  ...  
});
```

Здесь получаем данные другого участника, который только что присоединился:

```
}).on(ROOM_EVENT.JOINED, function(participant) {  
  installParticipant(participant);  
  addMessage(participant.name(), "joined");  
  ...  
});
```

1. Инициализация API.

Flashphoner.init() [code](#)

```
Flashphoner.init();
```

2. Запрос доступа к камере и микрофону

Flashphoner.getMediaAccess() [code](#)

```
Flashphoner.getMediaAccess(null, localDisplay).then(function() {
  createConnection(url, username);
}).catch(function(error) {
  console.error("User not allowed media access: "+error);
  $("#failedInfo").text("User not allowed media access. Refresh the page");
  onLeft();
});
```

3. Подключение к серверу.

`RoomApi.connect()` [code](#)

```
function createConnection(url, username) {
  connection = RoomApi.connect({urlServer: url, username: username}).on(SESSION_STATUS.FAILED, function
(session){
  ...
});
}
```

4. Получение от сервера события, подтверждающего успешное соединение

`ConnectionStatusEvent ESTABLISHED` [code](#)

```
connection = RoomApi.connect({urlServer: url, username: username}).on(SESSION_STATUS.FAILED, function(session){
  ...
}).on(SESSION_STATUS.DISCONNECTED, function(session) {
  ...
}).on(SESSION_STATUS.ESTABLISHED, function(session) {
  setStatus('#status', session.status());
  joinRoom();
});
```

5. Присоединение к конференции.

`connection.join()` [code](#)

При присоединении передается имя "комнаты" конференции (берется из параметра в URL страницы клиента, или генерируется случайное имя).

```
connection.join({name: getRoomName(), record: isRecord()}).on(ROOM_EVENT.STATE, function(room){
  ...
});
```

6. Получение от сервера события, описывающего статус комнаты

`RoomStatusEvent STATE` [code](#)

При получении данного события:

- определяется количество участников конференции с помощью метода `room.getParticipants()`, который возвращает массив объектов `participant`,
- если к конференции уже присоединилось максимально допустимое количество участников, производится выход из "комнаты" при помощи `room.leave()`
- если количество участников меньше максимально допустимого, начинается публикация видеопотока

```

connection.join({name: getRoomName(), record: isRecord()}).on(ROOM_EVENT.STATE, function(room){
    var participants = room.getParticipants();
    console.log("Current number of participants in the room: " + participants.length);
    if (participants.length >= _participants) {
        console.warn("Current room is full");
        $("#failedInfo").text("Current room is full.");
        room.leave().then(onLeft, onLeft);
        return false;
    }
    setInviteAddress(room.name());
    if (participants.length > 0) {
        var chatState = "participants: ";
        for (var i = 0; i < participants.length; i++) {
            installParticipant(participants[i]);
            chatState += participants[i].name();
            if (i != participants.length - 1) {
                chatState += ",";
            }
        }
        addMessage("chat", chatState);
    } else {
        addMessage("chat", " room is empty");
    }
    publishLocalMedia(room);
    onJoined(room);
}).on(ROOM_EVENT.JOINED, function(participant){
    ...
}).on(ROOM_EVENT.LEFT, function(participant){
    ...
}).on(ROOM_EVENT.PUBLISHED, function(participant){
    ...
}).on(ROOM_EVENT.FAILED, function(room, info){
    ...
}).on(ROOM_EVENT.MESSAGE, function(message){
    ...
});

```

7. Проверка режима Low Power Mode при публикации на мобильном устройстве

Flashphoner.playFirstVideo() [code](#)

```

if (Browser.isSafariWebRTC()) {
    var display = document.getElementById("localDisplay");
    Flashphoner.playFirstVideo(display, true, PRELOADER_URL).then(function() {
        publishLocalMedia(room);
    }).catch(function (error) {
        console.log("Can't atomatically publish local stream, use Publish button");
        for (var i = 0; i < display.children.length; i++) {
            if (display.children[i]) {
                console.log("remove cached instance id " + display.children[i].id);
                display.removeChild(display.children[i]);
            }
        }
        onMediaStopped(room);
    });
}

```

8. Публикация видеопотока.

room.publish() [code](#)

При публикации передаем div-элемент, в котором будет отображаться видео с камеры.

```

room.publish({
  display: display,
  constraints: constraints,
  record: false,
  receiveVideo: false,
  receiveAudio: false
}).on(STREAM_STATUS.FAILED, function (stream) {
  console.warn("Local stream failed!");
  setStatus("#localStatus", stream.status());
  onMediaStopped(room);
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
  setStatus("#localStatus", stream.status());
  onMediaPublished(stream);
}).on(STREAM_STATUS.UNPUBLISHED, function(stream) {
  setStatus("#localStatus", stream.status());
  onMediaStopped(room);
});

```

9. Получение от сервера события, сигнализирующего о присоединении пользователя к чат-комнате

RoomStatusEvent JOINED [code](#)

```

connection.join({name: getRoomName(), record: isRecord()}).on(ROOM_EVENT.STATE, function(room){
  ...
}).on(ROOM_EVENT.JOINED, function(participant){
  installParticipant(participant);
  addMessage(participant.name(), "joined");
}).on(ROOM_EVENT.LEFT, function(participant){
  ...
}).on(ROOM_EVENT.PUBLISHED, function(participant){
  ...
}).on(ROOM_EVENT.FAILED, function(room, info){
  ...
}).on(ROOM_EVENT.MESSAGE, function(message){
  ...
});

```

10. Получение от сервера события, сигнализирующего о публикации видеопотока другим участником

RoomStatusEvent PUBLISHED [code](#)

```

connection.join({name: getRoomName(), record: isRecord()}).on(ROOM_EVENT.STATE, function(room){
  ...
}).on(ROOM_EVENT.JOINED, function(participant){
  ...
}).on(ROOM_EVENT.LEFT, function(participant){
  ...
}).on(ROOM_EVENT.PUBLISHED, function(participant){
  playParticipantsStream(participant);
}).on(ROOM_EVENT.FAILED, function(room, info){
  ...
}).on(ROOM_EVENT.MESSAGE, function(message){
  ...
});

```

11. Проверка режима Low Power Mode при воспроизведении на мобильном устройстве

Flashphoner.playFirstVideo() [code](#)

```
        if (Browser.isSafariWebRTC()) {
            Flashphoner.playFirstVideo(pDisplay, false, PRELOADER_URL).then(function() {
                playStream(participant, pDisplay);
            }).catch(function (error) {
                // Low Power Mode detected, user action is needed to start playback in this mode #WCS-
2639
                console.log("Can't automatically play participant" + participant.name() + " stream, use
Play button");

                for (var i = 0; i < pDisplay.children.length; i++) {
                    if (pDisplay.children[i]) {
                        console.log("remove cached instance id " + pDisplay.children[i].id);
                        pDisplay.removeChild(pDisplay.children[i]);
                    }
                }
                onParticipantStopped(participant);
            });
        }
    }
}
```

12. Воспроизведение видеопотока.

`participant.play()` [code](#)

В метод передаются следующие параметры:

- `display` - div-элемент, в котором будет отображаться видео;
- `options.unmutePlayOnStart` - параметр, разрешающий (по умолчанию) или запрещающий (например, в Android Edge) автоматическое включение звука при проигрывании
- `options.constraints.audio.deviceId` - устройство вывода звука (в примере указано устройство по умолчанию)

Если автоматическое включение звука запрещено, пользователь должен нажать кнопку для включения звука

```

var options = {
  unmutePlayOnStart: true,
  constraints: {
    audio: {
      deviceId: 'default'
    }
  }
};
// Leave participant stream muted in Android Edge browser #WCS-3445
if (Browser.isChromiumEdge() && Browser.isAndroid()) {
  options.unmutePlayOnStart = false;
}
participant.getStreams()[0].play(display, options).on(STREAM_STATUS.PLAYING, function (playingStream) {
  var video = document.getElementById(playingStream.id());
  video.addEventListener('resize', function (event) {
    resizeVideo(event.target);
  });
  // Set up participant Stop/Play button
  if (playBtn) {
    $(playBtn).text("Stop").off('click').click(function() {
      $(this).prop('disabled', true);
      playingStream.stop();
    }).prop('disabled', false);
  }
  // Set up participant audio toggle button #WCS-3445
  if (audioBtn) {
    $(audioBtn).text("Audio").off('click').click(function(){
      if (playingStream.isRemoteAudioMuted()) {
        playingStream.unmuteRemoteAudio();
      } else {
        playingStream.muteRemoteAudio();
      }
    }).prop('disabled', false);
  }
  // Start participant audio state checking timer #WCS-3445
  participantState.startMutedCheck(playingStream);
}).on(STREAM_STATUS.STOPPED, function () {
  onParticipantStopped(participant);
}).on(STREAM_STATUS.FAILED, function () {
  onParticipantStopped(participant);
});

```

13. Остановка публикации видеопотока.

stream.stop() [code](#)

```

function onMediaPublished(stream) {
  $("#localStopBtn").text("Stop").off('click').click(function(){
    $(this).prop('disabled', true);
    stream.stop();
  }).prop('disabled', false);
  ...
}

```

14. Получение от сервера события, подтверждающего остановку публикации.

StreamStatusEvent UNPUBLISHED [code](#)


```

room.publish({
  display: display,
  constraints: constraints,
  record: false,
  receiveVideo: false,
  receiveAudio: false
}).on(STREAM_STATUS.FAILED, function (stream) {
  ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
  ...
}).on(STREAM_STATUS.UNPUBLISHED, function(stream) {
  setStatus("#localStatus", stream.status());
  onMediaStopped(room);
});

```

15. Выход из комнаты конференции.

`room.leave()` [code](#)

```

function onJoined(room) {
  $("#joinBtn").text("Leave").off('click').click(function(){
    $(this).prop('disabled', true);
    room.leave().then(onLeft, onLeft);
  }).prop('disabled', false);
  ...
}

```

16. Включение/выключение аудио и видео для публикуемого потока.

`stream.isAudioMuted()`, `stream.isVideoMuted()`, `stream.muteAudio()`, `stream.unmuteAudio()`, `stream.muteVideo()`, `stream.unmuteVideo()` [code](#)

```

function onMediaPublished(stream) {
  ...
  $("#localAudioToggle").text("Mute A").off('click').click(function(){
    if (stream.isAudioMuted()) {
      $(this).text("Mute A");
      stream.unmuteAudio();
    } else {
      $(this).text("Unmute A");
      stream.muteAudio();
    }
  }).prop('disabled', false);
  $("#localVideoToggle").text("Mute V").off('click').click(function() {
    if (stream.isVideoMuted()) {
      $(this).text("Mute V");
      stream.unmuteVideo();
    } else {
      $(this).text("Unmute V");
      stream.muteVideo();
    }
  }).prop('disabled', false);
}

```

17. Отправка текстового сообщения.

`participant.sendMessage()` [code](#)

При нажатии на кнопку Send

- определяется массив участников конференции с помощью метода `room.getParticipants()`
- отправляется сообщение каждому участнику

```
function onJoined(room) {
  ...
  $('#sendMessageBtn').off('click').click(function(){
    var message = field('message');
    addMessage(connection.username(), message);
    $('#message').val("");
    //broadcast message
    var participants = room.getParticipants();
    for (var i = 0; i < participants.length; i++) {
      participants[i].sendMessage(message);
    }
  }).prop('disabled',false);
  $('#failedInfo').text("");
}
```