Работа с комнатами

Описание

Web Call Server позволяет внедрить видеочат, который будет работать в большинстве современных веб-браузеров без установки дополнительного ПО, а также на мобильных устройствах, в ваш веб-проект.

Поддерживаемые платформы и браузеры

	Chrome	**Firefox**	**Safari**	**Edge**
Windows	✅	✅	❌	✅
Mac OS	✅	✅	✅	✅
Android	✅	✅	❌	✅
iOS	✅	✅	✅	✅

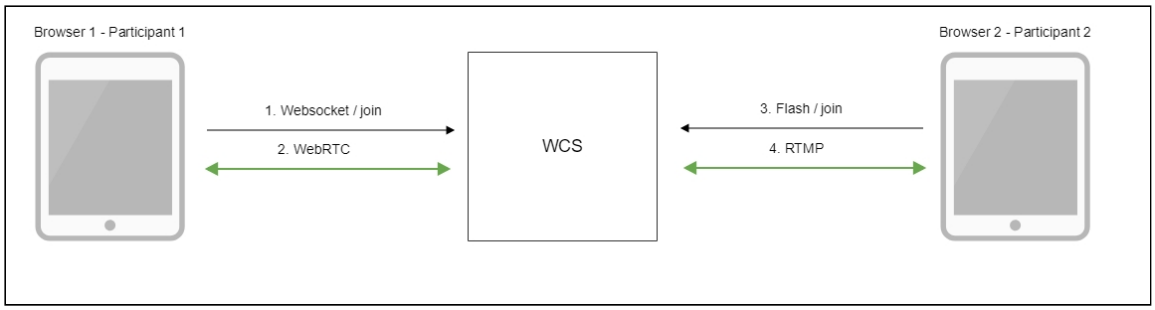
Поддерживаемые кодеки

- Видео: H.264, VP8
- Аудио: Opus, G.711

Функции

- Видеочат
- Текстовый чат
- Видеоконференция
- Видеоконференция с отображением экрана пользователя

Схема работы

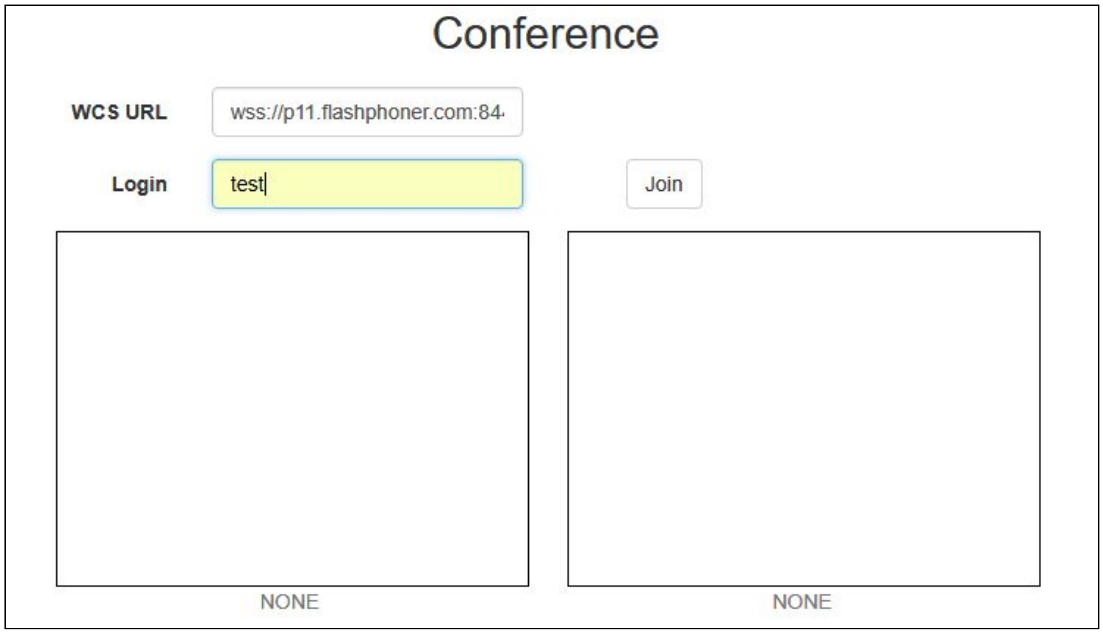


1. Браузер участника 1 соединяется с сервером по Websocket и отправляет команду `join`
2. Браузер участника 1 может передавать поток по WebRTC для публикации в чат-комнате и получать потоки, опубликованные в комнате
3. Браузер участника 2 соединяется с сервером при помощи Flash и отправляет команду `join`
4. Браузер участника 2 может передавать поток по RTMP для публикации в чат-комнате и получать потоки, опубликованные в комнате

Тестирование

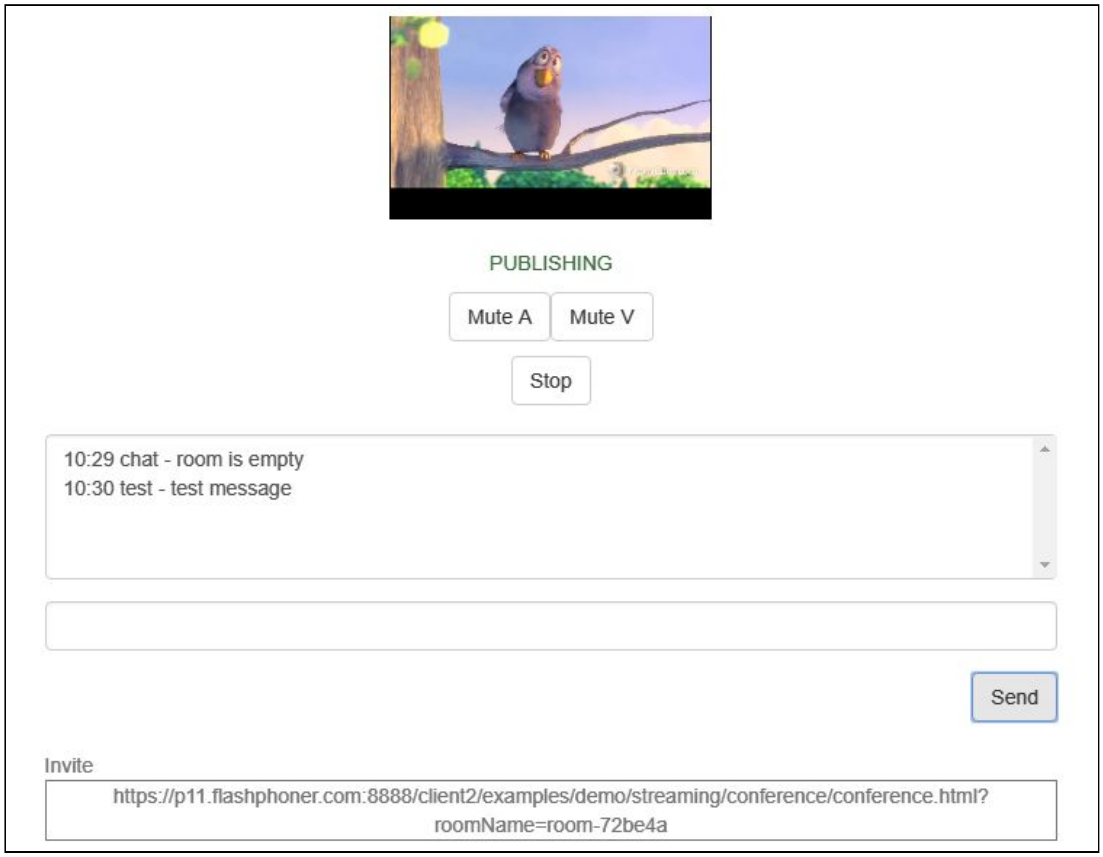
Тест видеоконференции

1. Для теста используем:
2. демо-сервер `demo.flashphoner.com`
3. веб-приложение Conference для организации видеоконференции
4. Откройте веб-приложение `Conference`. В поле `Login` введите произвольное имя пользователя, например `test`:



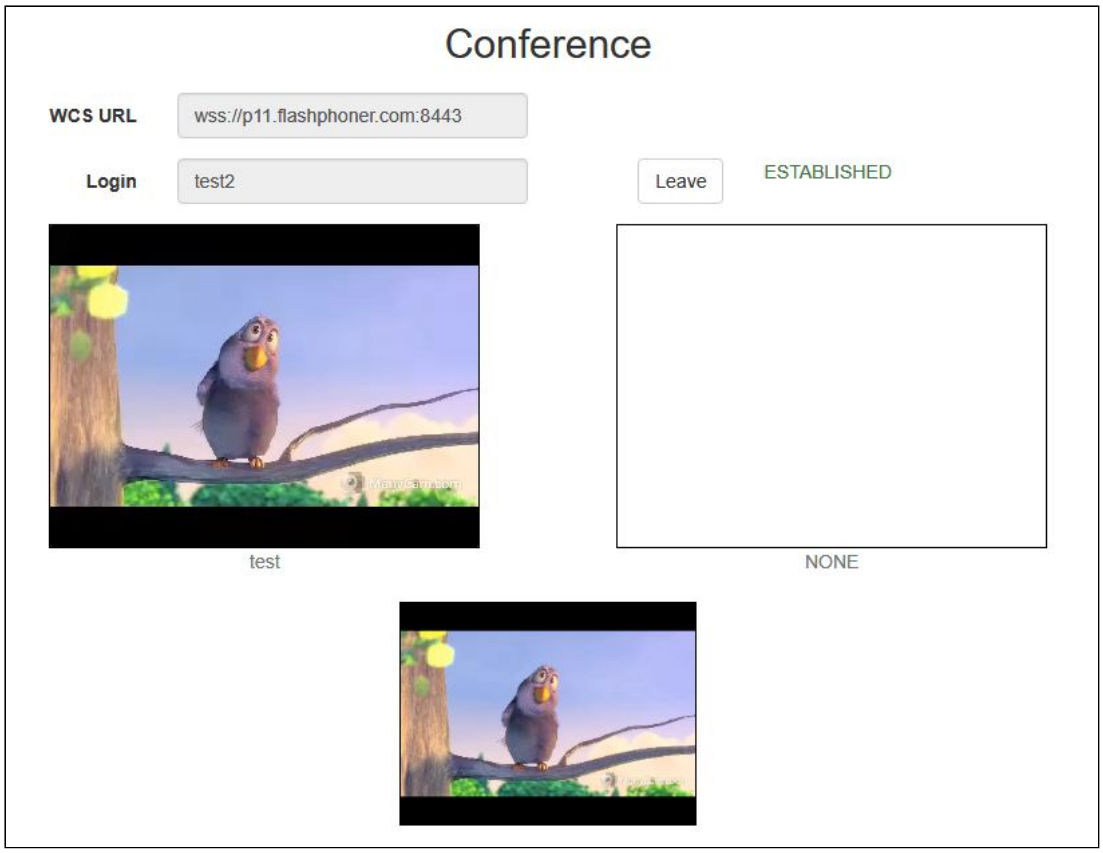
5. Нажмите кнопку `Join`. Установится соединение с сервером, отобразится надпись `ESTABLISHED`, будет автоматически создана чат-комната:

Внизу экрана отобразится изображение с веб-камеры, текстовый чат и ссылка на комнату для приглашения других пользователей:

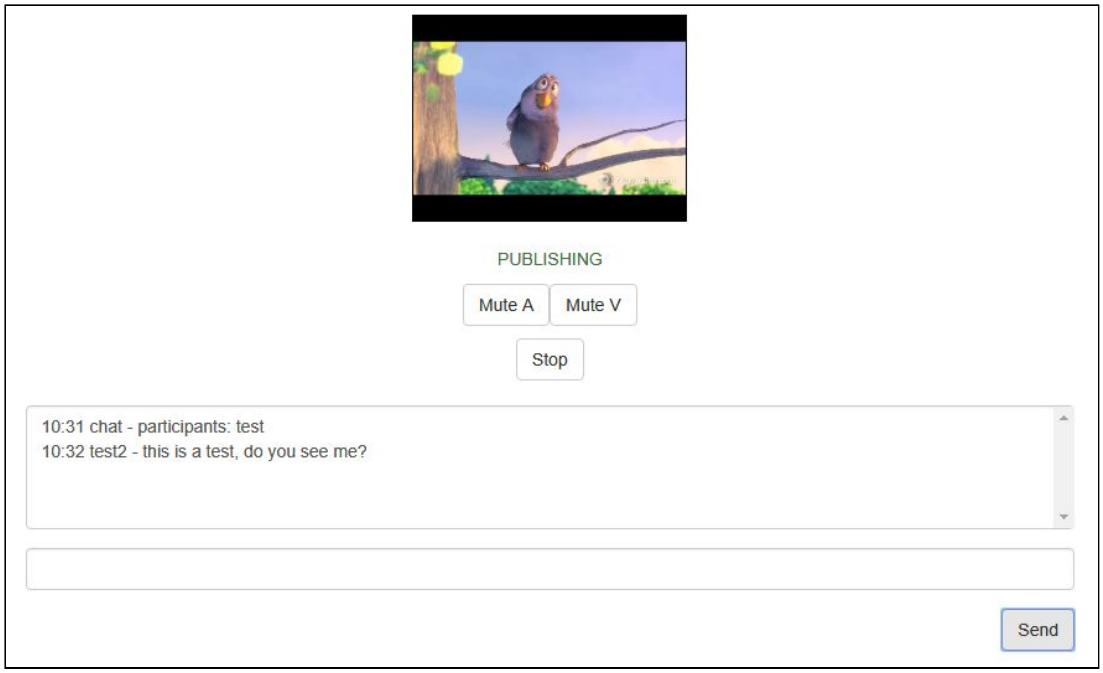


6. Скопируйте ссылку на чат-комнату и откройте ее в другой вкладке браузера. Введите имя пользователя, которое должно отличаться от имени создателя чат-комнаты, например, `test2`, и нажмите кнопку `Join`. На странице будет показано изображение с веб-камеры участника `test` (слева) и с веб-камеры участника

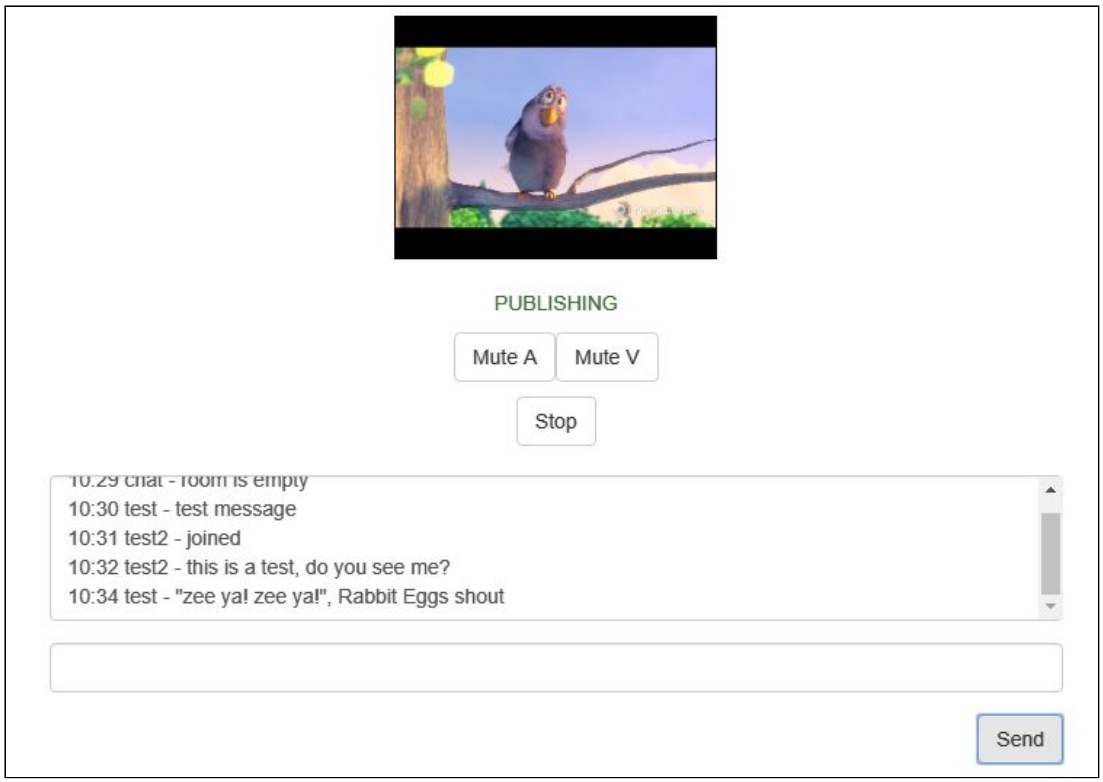
`test2` (внизу):



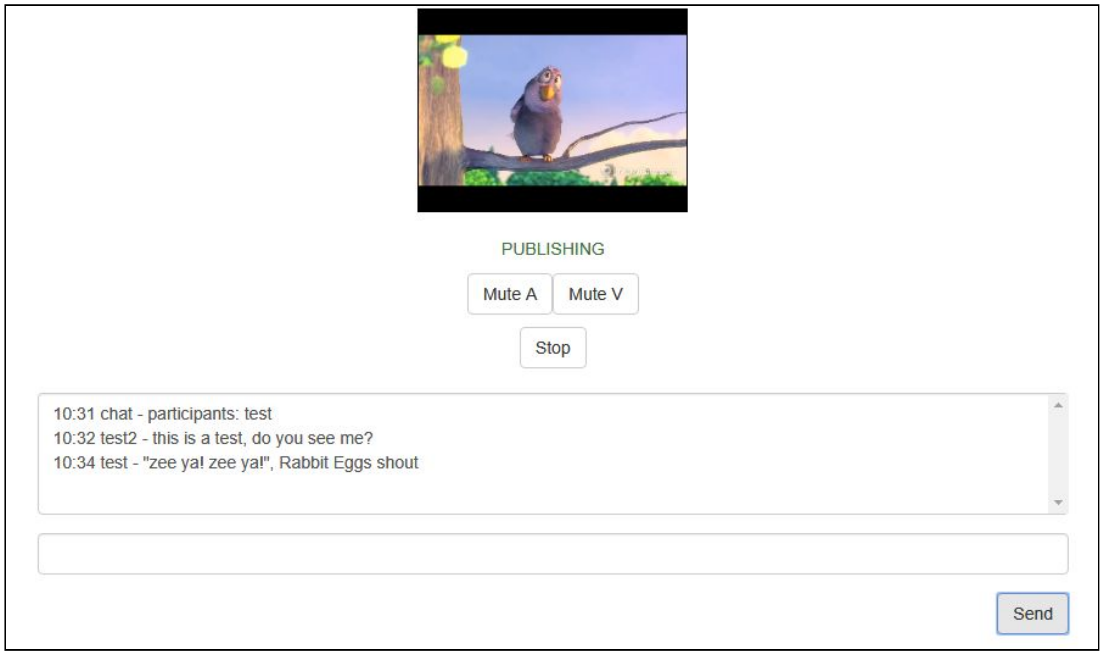
7. Введите в окне текстового чата участника `test2` сообщение и нажмите кнопку `Send`:



8. На вкладке участника `test` введите ответное сообщение:



9. Убедитесь, что ответ получен:



10. Для выхода из-чат-комнаты нажмите кнопку `Leave`

Тест видеочата

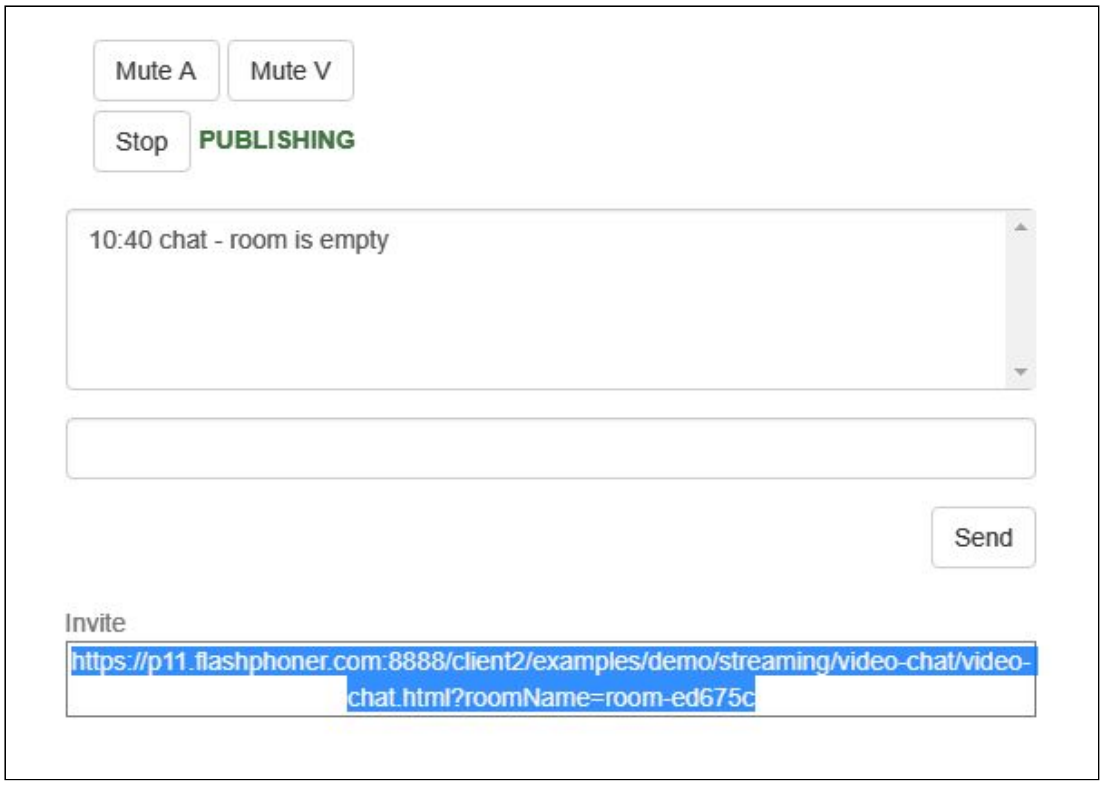
1. Для теста используем:
2. демо-сервер `demo.flashphoner.com`

3. веб-приложение Two Way Video Chat для организации видеочата
4. Откройте веб-приложение Two Way Video Chat. В поле `Login` введите произвольное имя пользователя, например `test`:

5. Нажмите кнопку `Join`. Установится соединение с сервером, отобразится надпись `ESTABLISHED`, будет автоматически создана чат-комната. Будет показано изображение с веб-камеры:

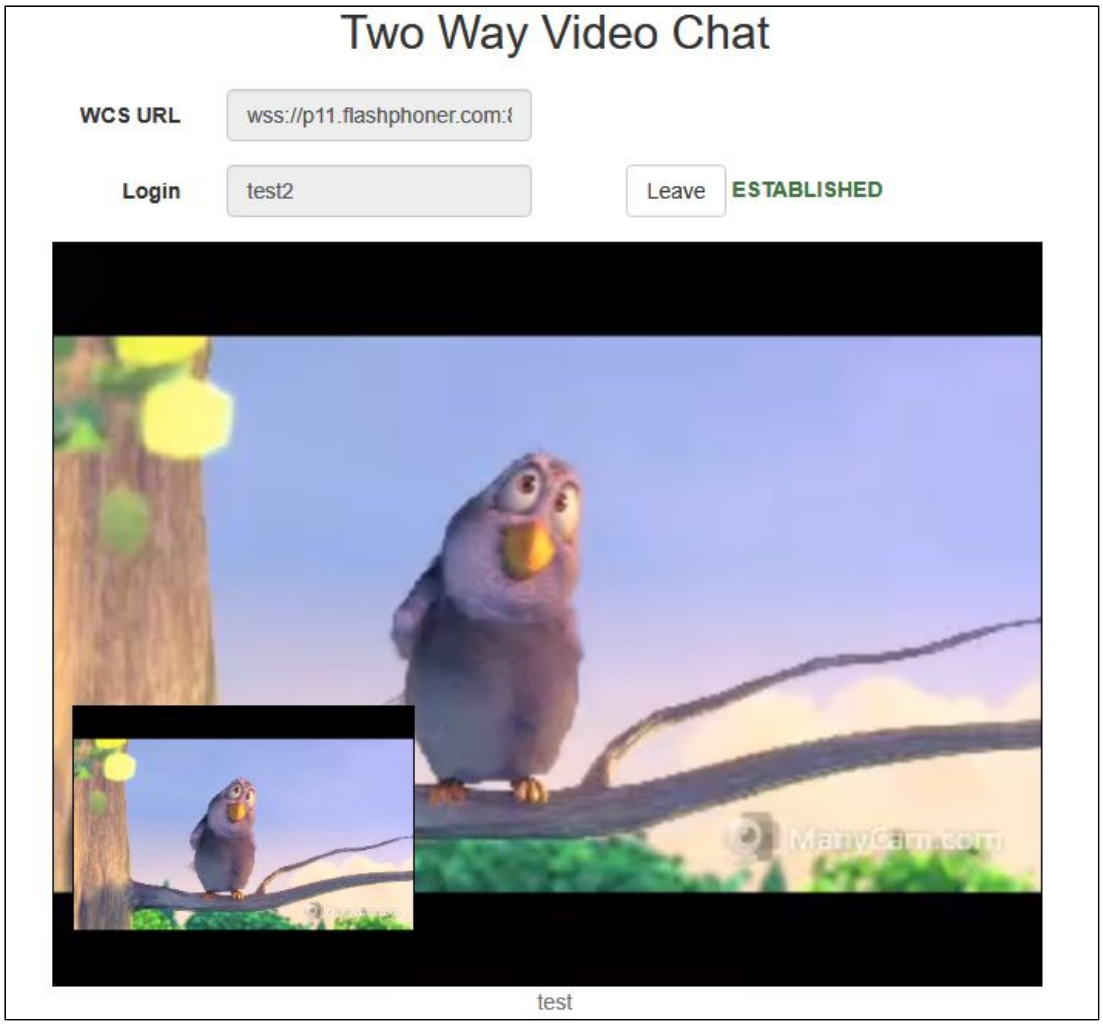
Внизу экрана отобразится текстовый чат и ссылка на комнату для приглашения

других пользователей:

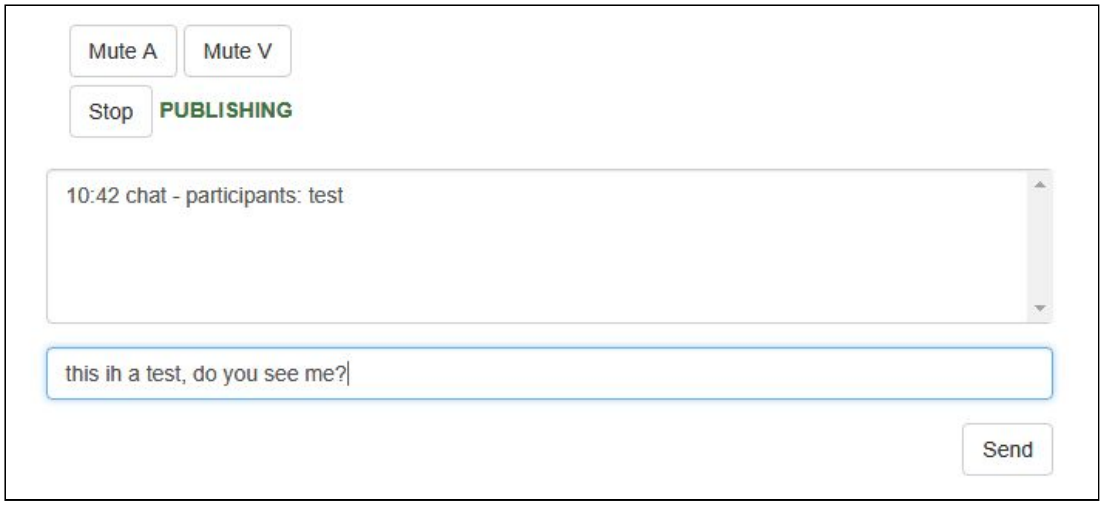


6. Скопируйте ссылку на чат-комнату и откройте ее в другой вкладке браузера. Введите имя пользователя, которое должно отличаться от имени создателя чат-комнаты, например, `test2`, и нажмите кнопку `Join`. На странице будет показано изображение с веб-камеры пользователя `test` крупно и с веб-камеры

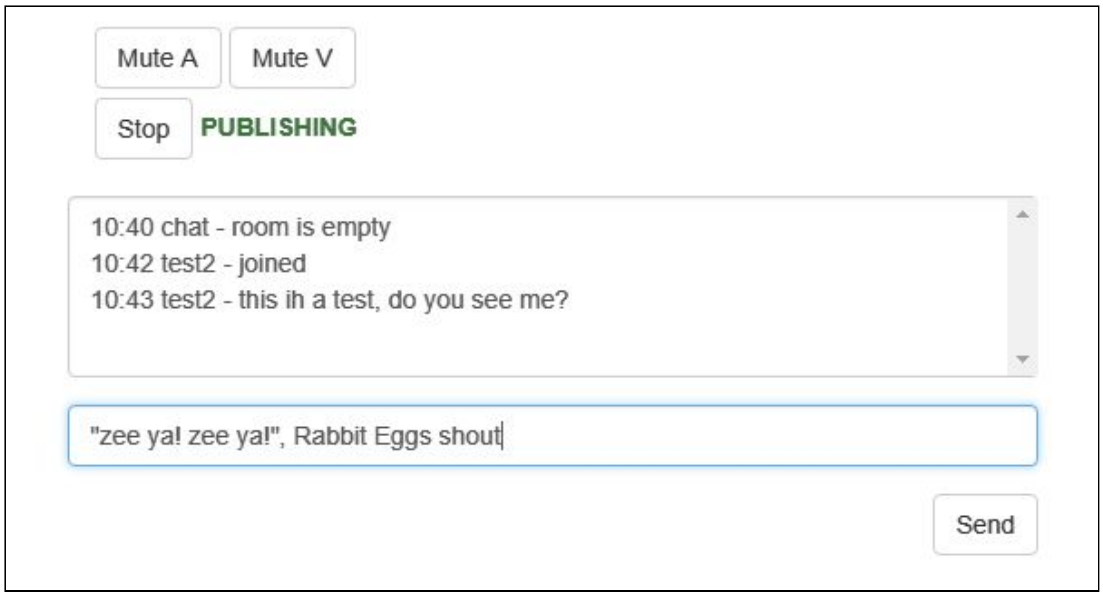
пользователя `test2` (внизу слева):



7. Введите в окне текстового чата сообщение и нажмите кнопку `Send`:



8. На вкладке пользователя test введите ответное сообщение:



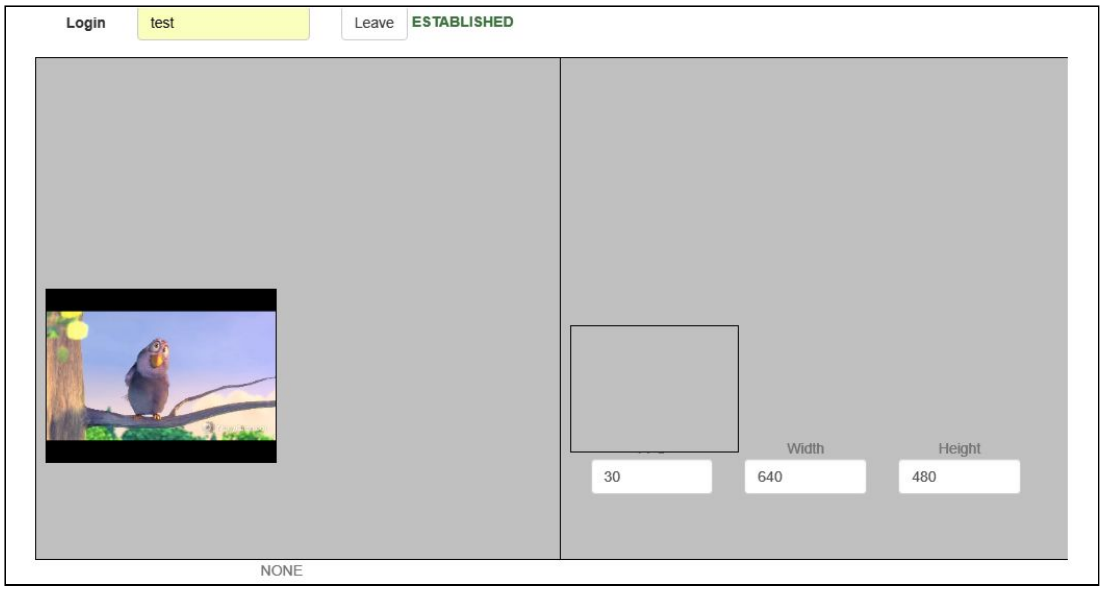
9. Убедитесь, что ответ получен:

8 Для выхода из-чат-комнаты нажмите кнопку `Leave`

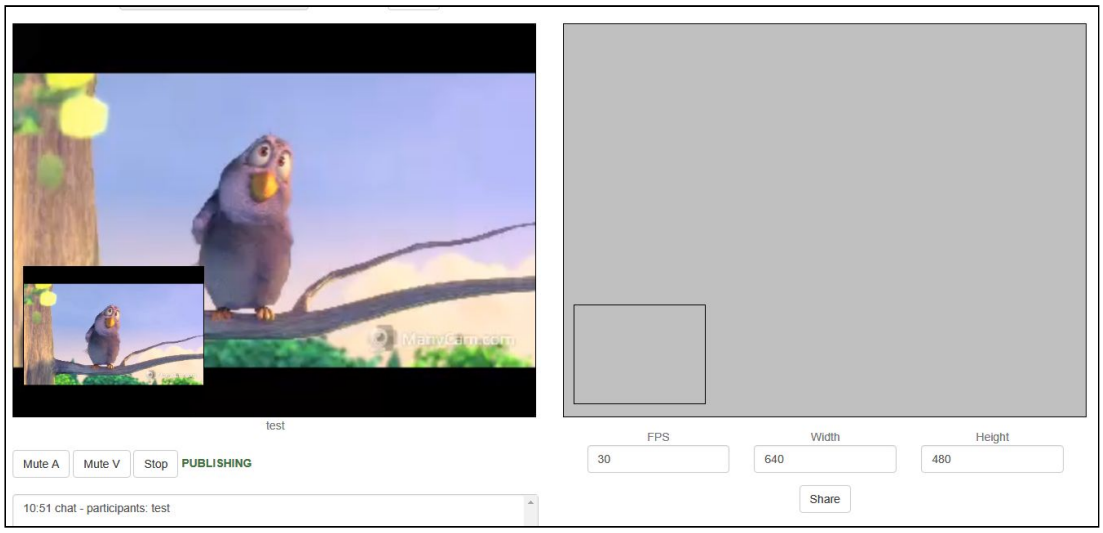
Тест видеоконференции с отображением экрана компьютера

1. Для теста используем:
2. демо-сервер `demo.flashphoner.com`
3. веб-приложение Two Way Video Chat and Screen Sharing для организации видеоконференции
4. браузер Chrome
5. Откройте веб-приложение Two Way Video Chat and Screen Sharing. В поле `Login` введите произвольное имя пользователя, например `test`. Нажмите кнопку `Join`. Установится соединение с сервером, отобразится надпись `ESTABLISHED`, будет

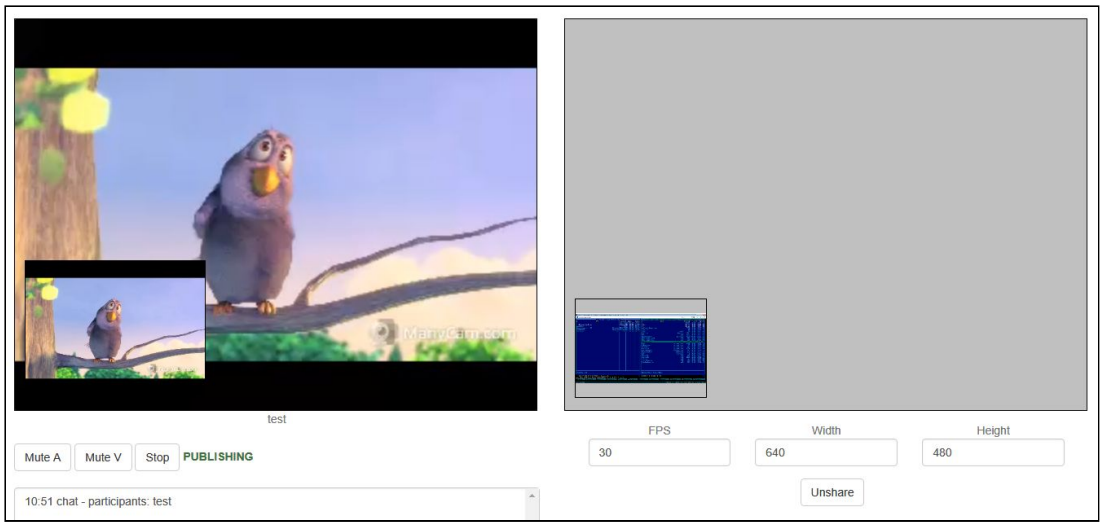
автоматически создана чат-комната. Будет показано изображение с веб-камеры:



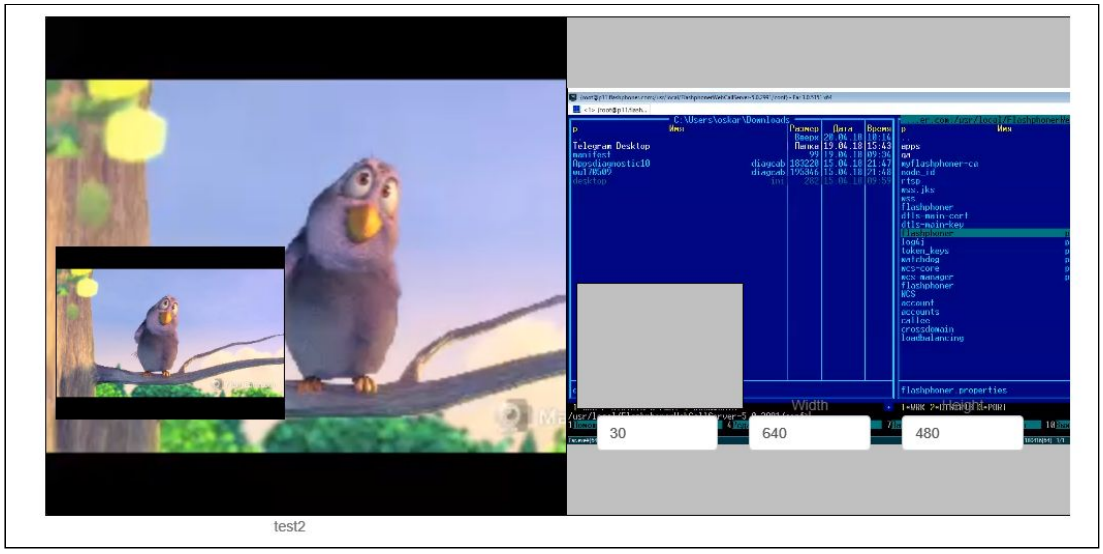
6. Скопируйте ссылку на чат-комнату и откройте ее в другой вкладке браузера. Введите имя пользователя, которое должно отличаться от имени создателя чат-комнаты, например, `test2`, и нажмите кнопку `Join`. На странице будет показано изображение с веб-камеры



7. Нажмите кнопку `Share` и разрешите браузеру доступ к экрану или к окну приложения:



8. На вкладке пользователя `test` отобразится экран или окно приложения, к которому был разрешен доступ:



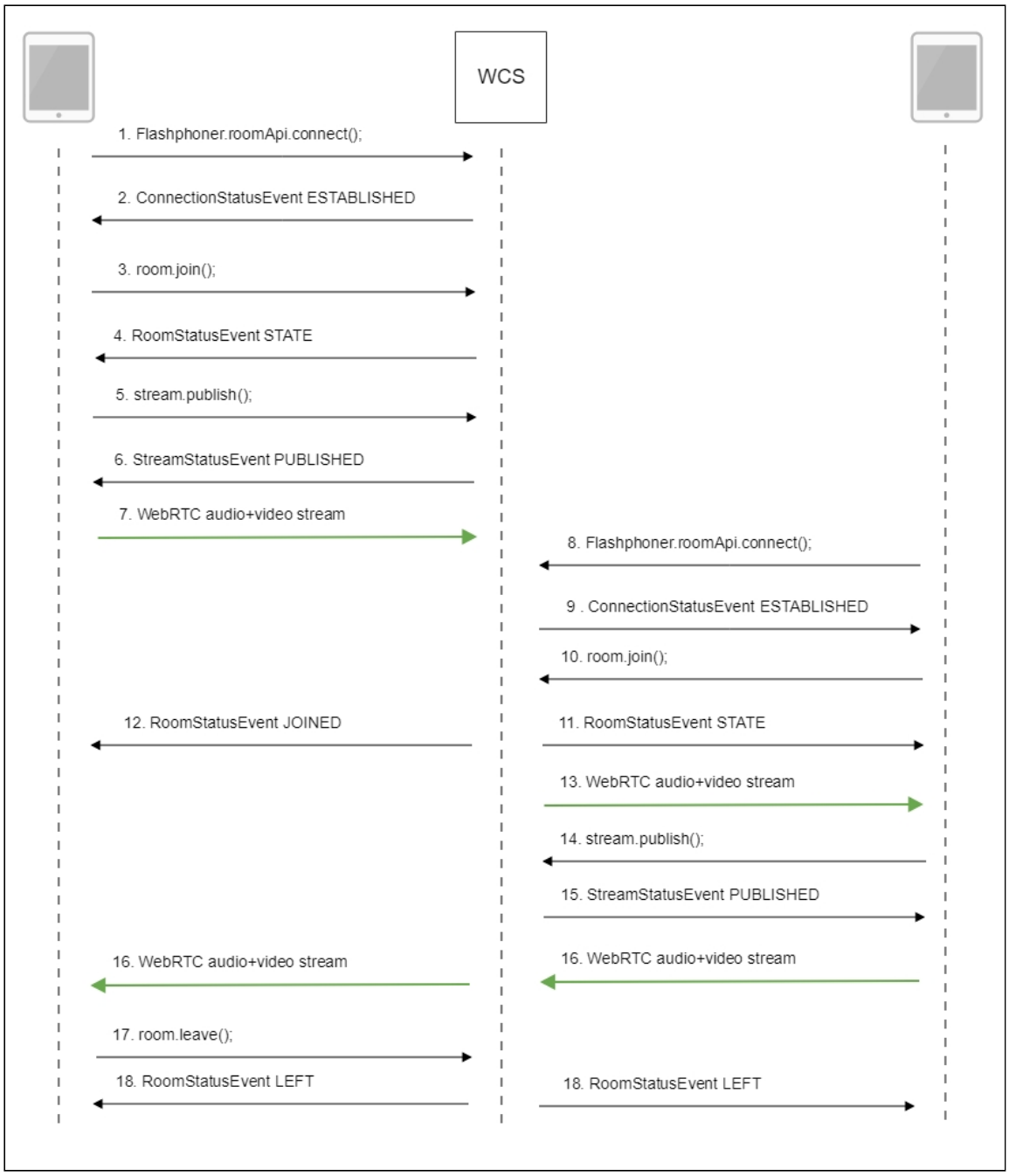
9. Для выхода из-чат-комнаты нажмите кнопку `Leave`

Последовательность выполнения операций

Ниже описана последовательность вызовов при использовании примера Conference

conference.html

conference.js



1. Установка участником 1 соединения с сервером

`RoomApi.connect()` code

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

2. Получение участником 1 от сервера события, подтверждающего успешное соединение
`SESSION_STATUS.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();
});
```

3. Вход в чат-комнату участника 1 `Session.join()` code

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

4. Получение участником 1 от сервера события, описывающего состояние комнаты
`ROOM_EVENT.STATE` code

```
connection.join({name: getRoomName()}).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);
  ...
});
```

5. Публикация медиапотока участником 1
`Room.publish()` code

```
room.publish({
  display: display,
  constraints: constraints,
  record: false,
  receiveVideo: false,
  receiveAudio: false
  ...
});
```

6. Получение участником 1 от сервера события, подтверждающего успешную публикацию потока
`STREAM_STATUS.PUBLISHING` code

```
room.publish({
  ...
}).on(STREAM_STATUS.FAILED, function (stream) {
  ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
  setStatus("#localStatus", stream.status());
  onMediaPublished(stream);
}).on(STREAM_STATUS.UNPUBLISHED, function(stream) {
  ...
});
```

7. Отправка участником 1 потока по WebRTC
8. Установка участником 2 соединения с сервером
9. Получение участником 2 от сервера события, подтверждающего успешное соединение
10. Вход в чат-комнату участника 2
11. Получение участником 2 от сервера события, описывающего состояние комнаты
12. Получение участником 1 от сервера события о присоединении участника 2
`ROOM_EVENT.JOINED` code

```
connection.join({name: getRoomName()}).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){
    ...
});
```

13. Получение участником 2 потока, опубликованного участником 1
14. Публикация медиапотока участником 2
15. Получение участником 2 от сервера события, подтверждающего успешную публикацию потока
16. Отправка участником 2 потока по WebRTC и получение его участником 1
17. Выход участника 1 из чат-комнаты
    `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);
    ...
}
```

18. Получение участниками комнаты от сервера события о выходе участника 1
    `ROOM_EVENT.LEFT` code

```
connection.join({name: getRoomName()}).on(ROOM_EVENT.STATE, function(room)
{
    ...
}).on(ROOM_EVENT.JOINED, function(participant){
    ...
}).on(ROOM_EVENT.LEFT, function(participant){
    //remove participant
    removeParticipant(participant);
    addMessage(participant.name(), "left");
}).on(ROOM_EVENT.PUBLISHED, function(participant){
    ...
}).on(ROOM_EVENT.FAILED, function(room, info){
    ...
}).on(ROOM_EVENT.MESSAGE, function(message){
    ...
});
```

Запись потоков, опубликованных участниками конференции

Видеопотоки, опубликованные каждым из участников конференции, могут быть записаны. Для этого необходимо установить параметр `record` в `true` при публикации потока:

```
room.publish({
    display: display,
    constraints: constraints,
    record: true,
    ...
});
```

Поток от каждого участника записывается в отдельный файл. Особенность этих файлов при дальнейшей обработке в том, что публикация начинается не одновременно.

Синхронизация потоков, опубликованных участниками комнаты

> **Warning**
>
> Данная возможность не поддерживается, начиная со сборки 5.2.142. Используйте микширование или запись потоков в один файл.

Для того, чтобы дать возможность объединить потоки участников, потоки комнаты могут быть синхронизированы по первому опубликованному потоку. Эта возможность включается при помощи параметра в файле flashphoner.properties

```
enable_empty_shift_writer=true
```

Например, если участник `User1` начал публиковать поток в 00:00:10, а участник `User2` в 00:00:55, то второй пользователь получит в начале записи 45 секунд пустого видео (черный экран и тишина). Таким образом, файлы записи потоков `User1.mp4` и `User2.mp4` будут одинаковы по времени, и их можно будет объединить.

Объединение синхронизированных записей потоков при помощи ffmpeg

Синхронизированные файлы записей потоков могут быть объединены при помощи ffmpeg с сохранением хронологического порядка. Для этого при создании потока на стороне сервера фиксируется его сдвиг относительно времени создания комнаты. Записанные таким образом файлы потоков объединяются командой (пример для двух участников)

```
ffmpeg -i stream1.mp4 -i stream2.mp4 -filter_complex "[0:v]pad=iw*2:ih[int];
[int][1:v]overlay=W/2:0[vid];[0:a][1:a]amerge[a]" -map [vid] -map "[a]" -ac 2
-strict -2 -c:v libx264 -crf 23 -preset veryfast output.mp4
```

Здесь

- `stream1` - поток первого участника
- `stream2` - поток второго участника

Запись потоков комнаты в один файл с последующим микшированием

В сборке WCS 5.2.1012 и сборке WebSDK 2.0.190 добавлена возможность записывать все потоки комнаты в один файл, с его автоматическим микшированием по окончании конференции. Для этого первый участник при создании комнаты должен указать опцию `record`:

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

В этом случае все потоки в комнате будут записаны в один файл. При завершении комнаты запись также будет завершена, и автоматически запустится скрипт, указанный в настройке

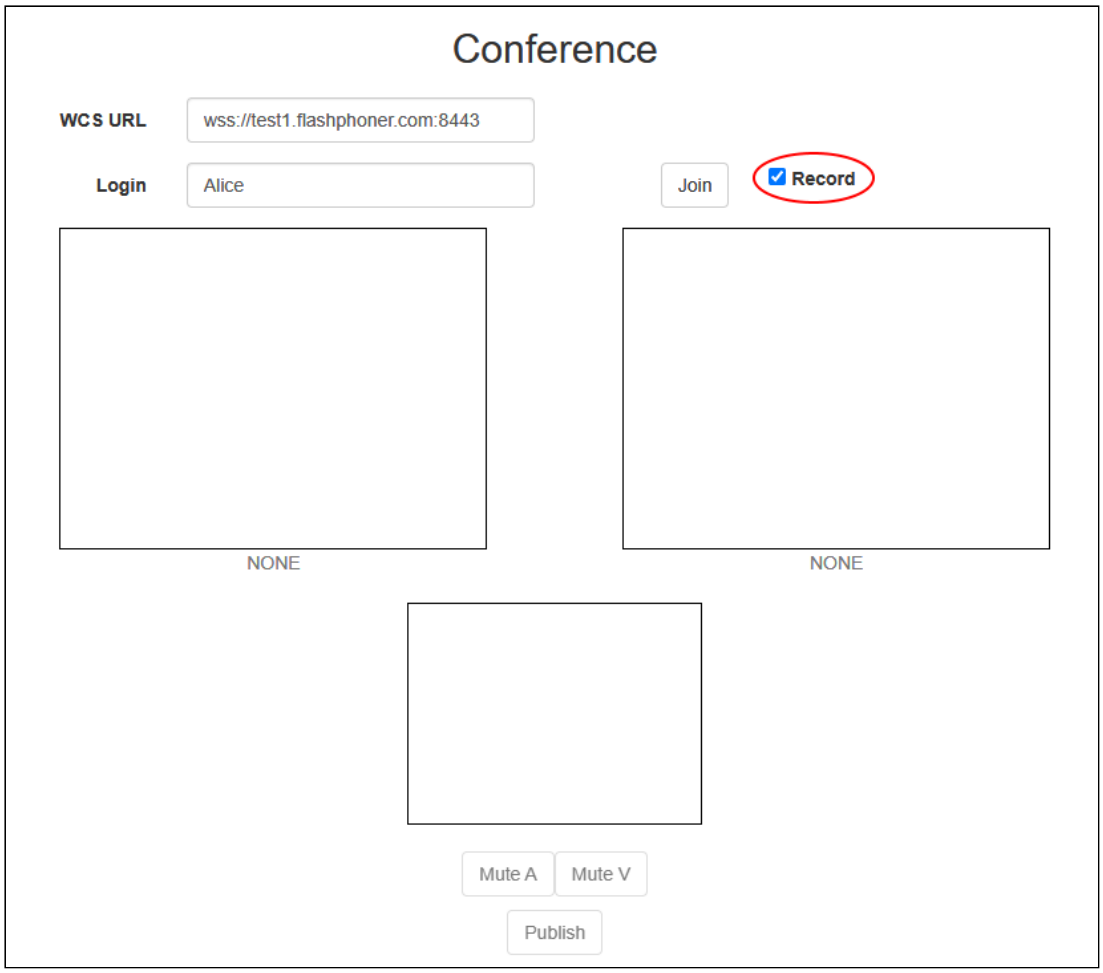
```
on_multiple_record_hook_script=on_multiple_record_hook.sh
```

который смикширует потоки в соответствии с настройками микшера, заданными в файле `/usr/local/FlashphonerWebCallServer/conf/offline_mixer.json`, по умолчанию

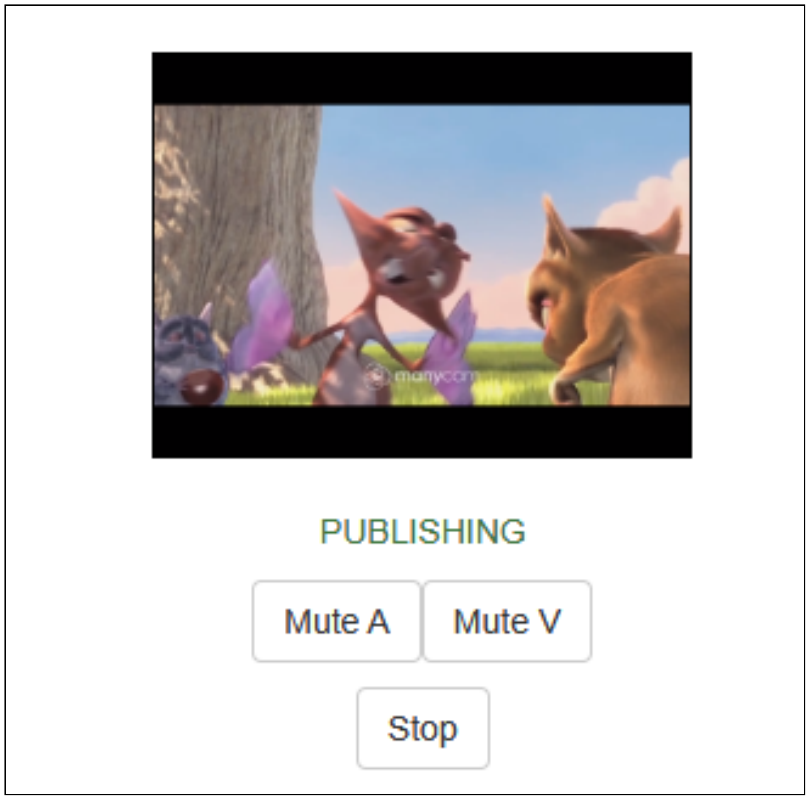
```
{
    "hasVideo": "true",
    "hasAudio": "true",
    "mixerDisplayStreamName": true
}
```

Тестирование записи комнаты

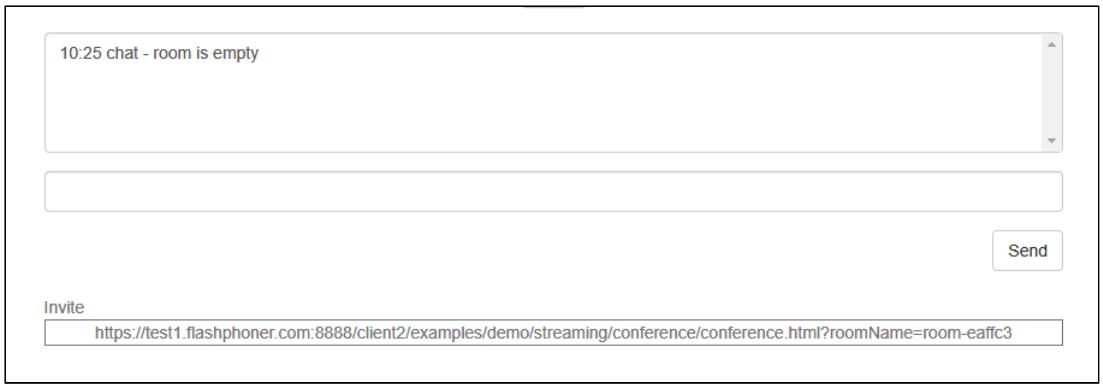
1. Для теста используем:
2. ваш WCS сервер, например `test1.flashphoner.com`
3. веб-приложение Conference
4. Откройте пример Conference в браузере, введите имя участника `Alice` и взведите переключатель `Record`



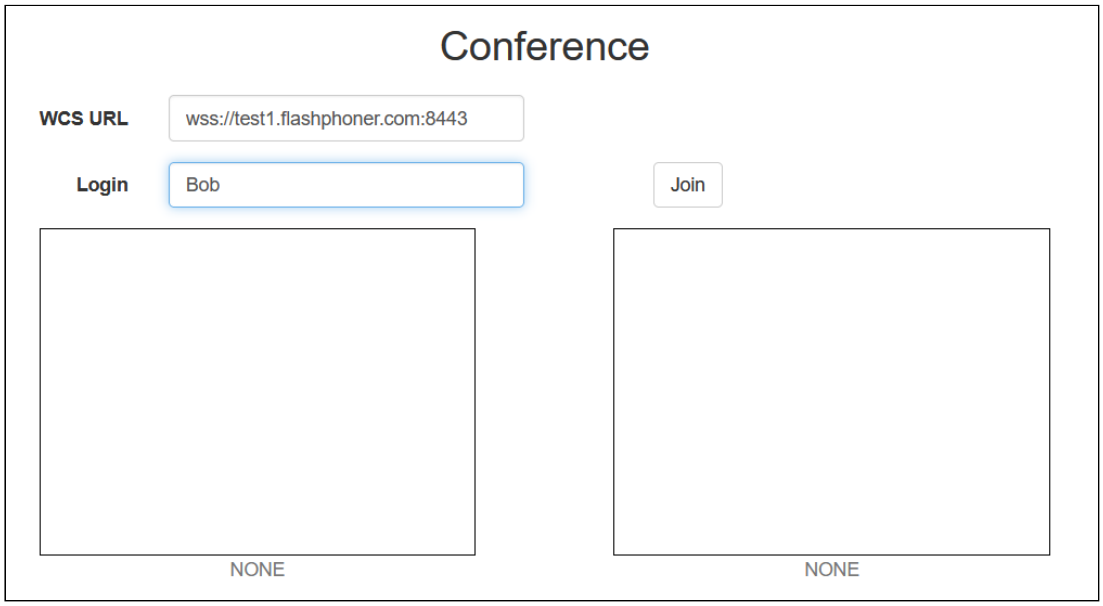
5. Нажмите `Join`. Начнется публикация потока



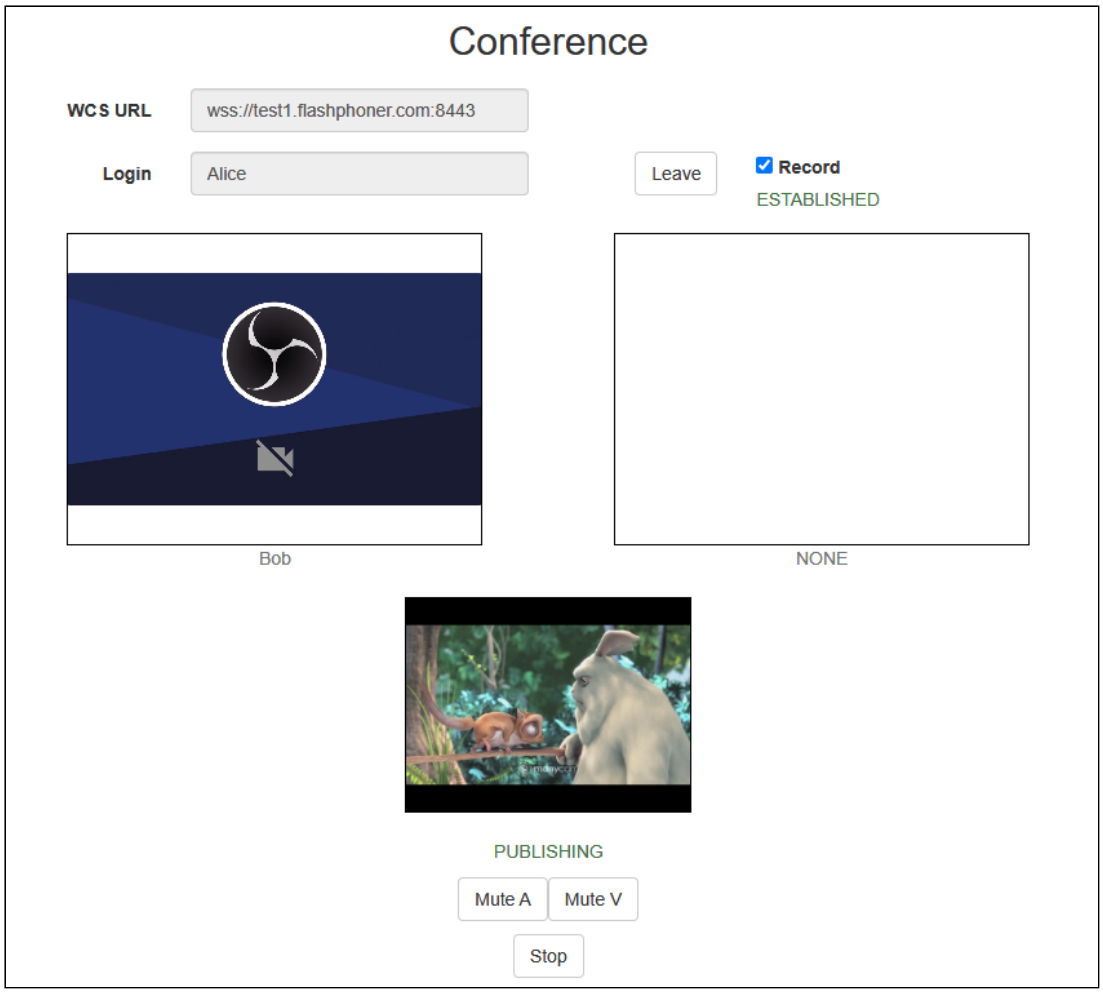
6. В другом окна браузера откройте ссылку из поля `Invite`



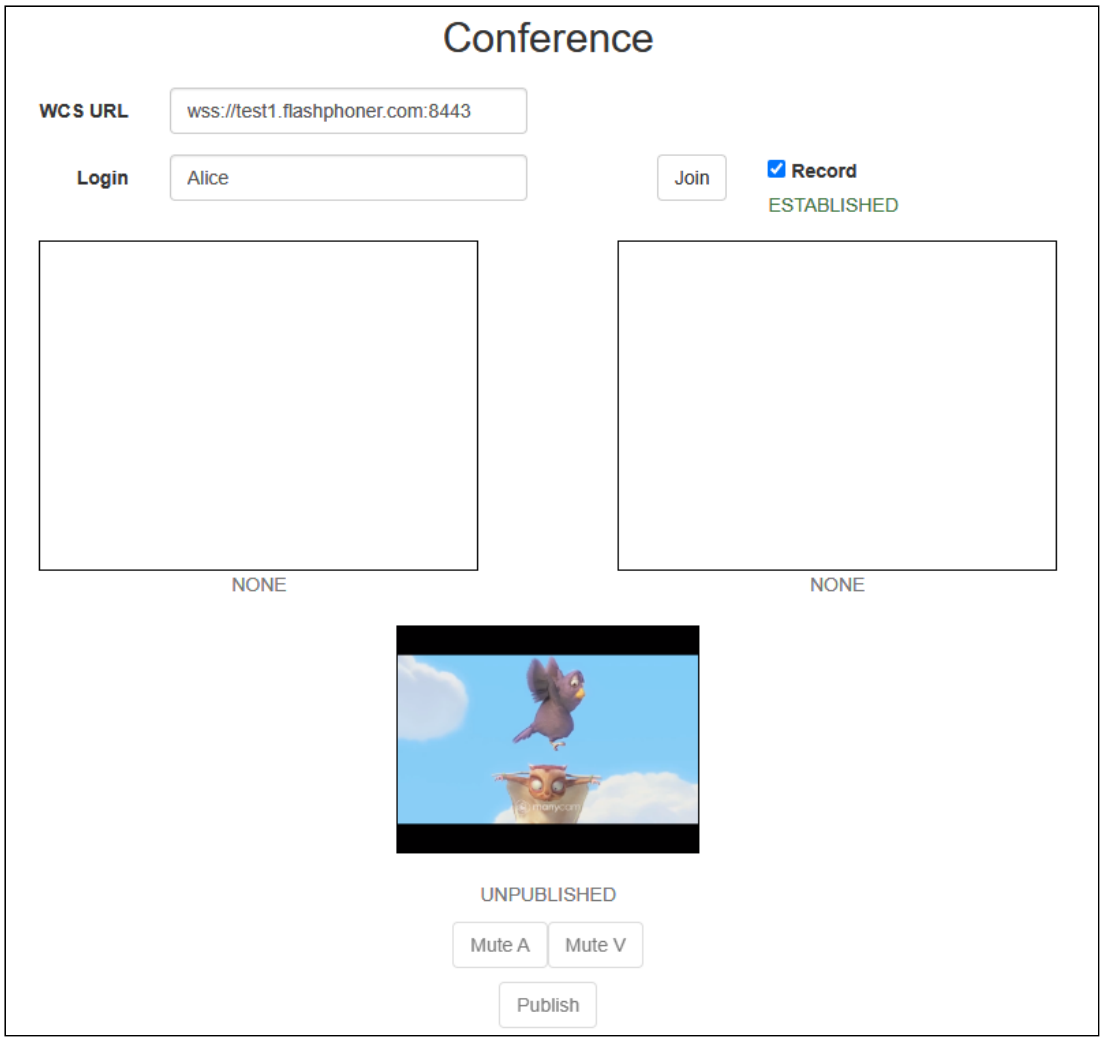
7. Введите имя пользователя `Bob` и нажмите `Join`



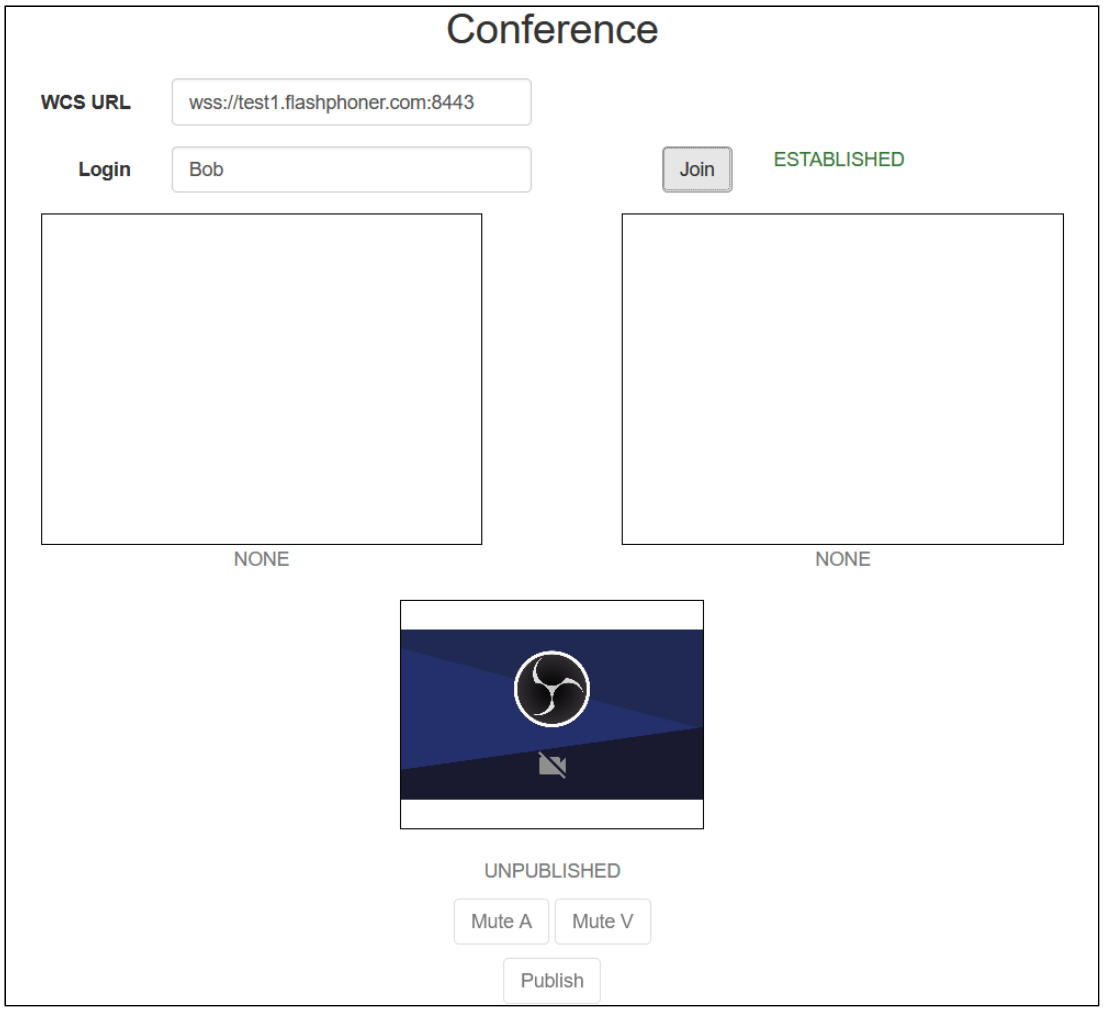
8. Bob присоединился к комнате



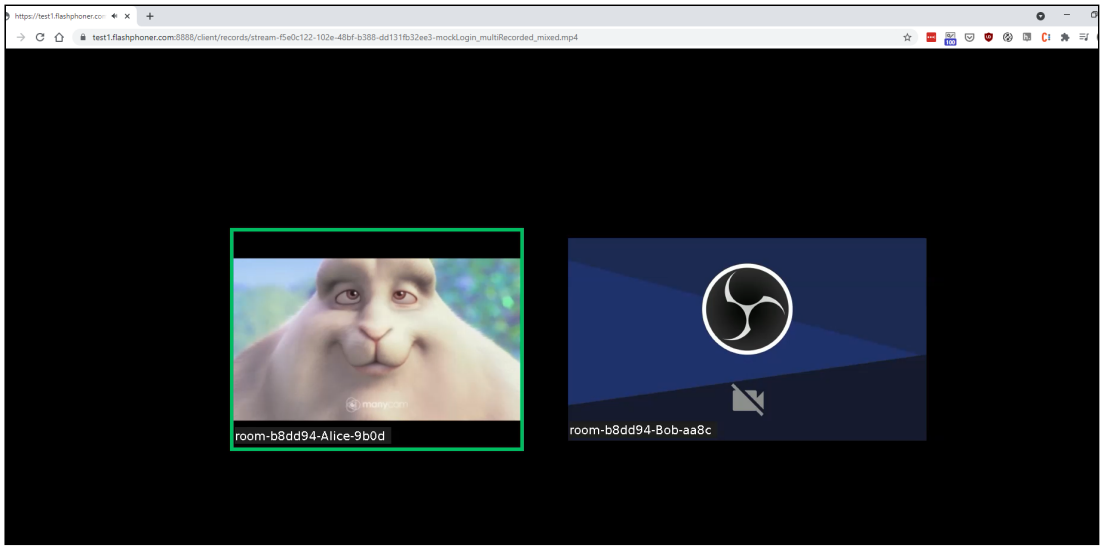
9. Нажмите `Leave` в окне пользователя `Alice`



и в окне пользователя `Bob`



10. Микширование может занять продолжительное время, в зависимости от длительности записи, производительности процессора и жесткого диска сервера. По его окончании, загрузите файл из каталога `/usr/local/FlashphonerWebCallServer/records` или откройте в браузере по ссылке



Завершение комнаты

Комната существует на сервере до тех пор, пока в ней есть хотя бы один участник. Когда последний участник вызывает функцию `Room.leave()`, комната завершается.

Если последний участник обновляет страницу или теряет соединение с сервером без вызова функции `Room.leave()`, комната остается активной в течение времени, заданного настройкой в миллисекундах

```
room_idle_timeout=60000
```

По умолчанию, время составляет 60 секунд. По истечении этого времени, комната завершается.

Известные проблемы

1. При обмене текстовыми сообщениями необходимо кодирование не-латинских символов

Симптомы

При отправке сообщения, содержащего не-латинские символы, эти символы преобразуются в знаки вопроса при получении

Решение

Использовать функции JavaScript `encodeURIComponent()` при отправке сообщения

```
var participants = room.getParticipants();
for (var i = 0; i < participants.length; i++) {
   participants[i].sendMessage(encodeURIComponent(message));
}
```

и `decodeURIComponent()` при его получении

```
connection.join({name: getRoomName(), record:
isRecord()}).on(ROOM_EVENT.STATE, function(room) {
...
}).on(ROOM_EVENT.MESSAGE, function(message){
    addMessage(message.from.name(), decodeURIComponent(message.text));
});
```

2. Следует избегать быстрого последовательного вызова `Room.leave()` и затем `Session.join()`

При быстром вызове `Room.leave()` и затем `Session.join()` для одного и того же участника возможна отправка серверу команды `join`, в то время как сервер еще не обработал предыдущую команду `leave` для этого пользователя

Симптомы

При вызове `Session.join()` сразу после `Room.leave()` клиент получает сообщение

```
Room already has user with such login
```

Решение

Использовать интервал не менее 1 секунды между последовательными вызовами `Room.leave()` и `Session.join()`