Работа с комнатами

- Описание
  - Поддерживаемые платформы и браузеры
  - Поддерживаемые кодеки
  - Функции
  - Схема работы
- Видеоконференция
- Видеочат
- Видеоконференция с отображением экрана компьютера (screen sharing)
- Последовательность выполнения операций (Call Flow)
- Запись потоков, опубликованных участниками конференции
  - Синхронизация потоков, опубликованных участниками комнаты
  - Тестирование записи потоков с синхронизацией
  - Объединение синхронизированных записей потоков при помощи ffmpeg
- Запись потоков комнаты в один файл с последующим микшированием
  - Тестирование
- Завершение комнаты
- Известные проблемы

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 11	Chromium Edge
Windows	+	+		+
Mac OS	+	+	+	
Android	+	+		+
iOS	+ (iOS 14.6+)	+ (iOS 14.6+)	+	

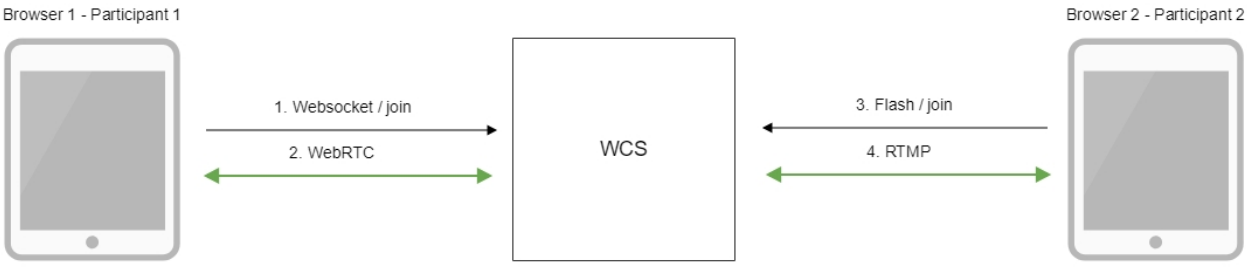
Поддерживаемые кодеки

- 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.

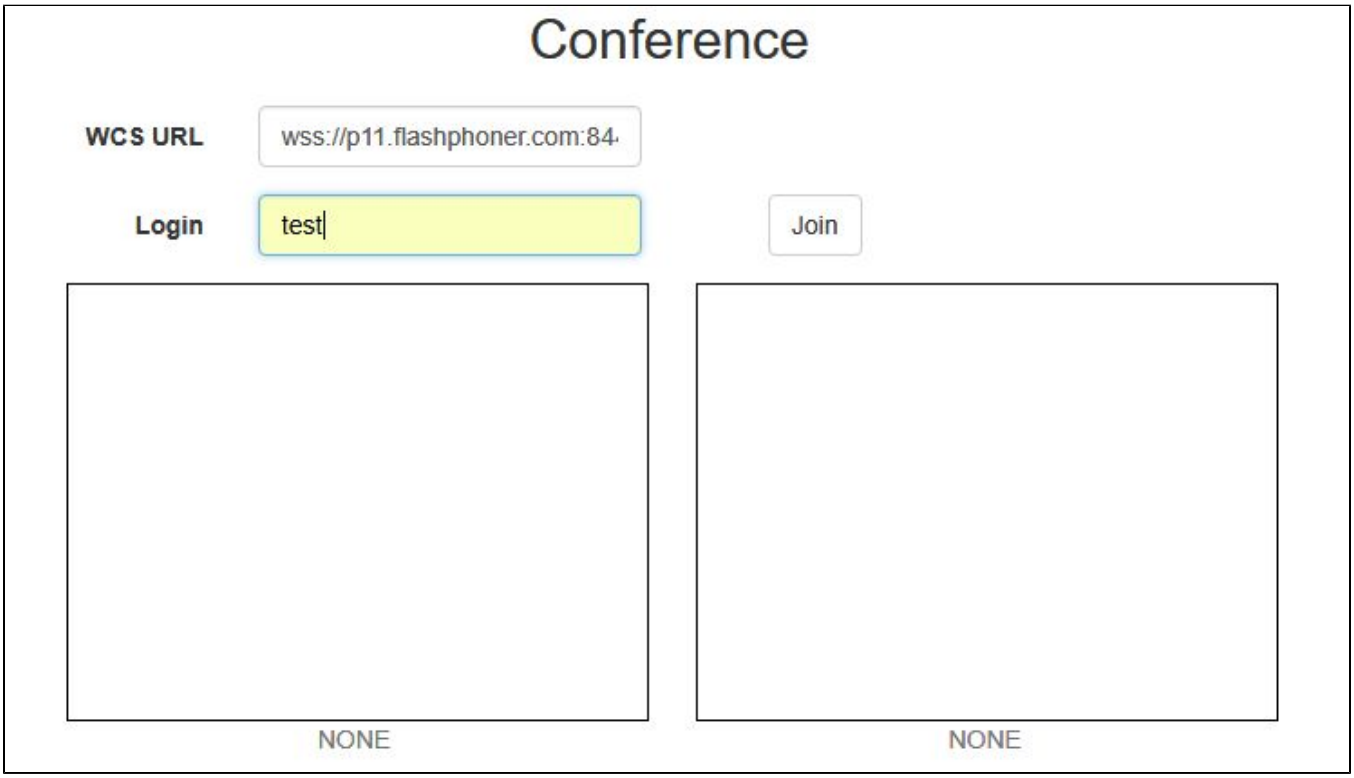
4. Браузер участника 2 может передавать поток по RTMP для публикации в чат-комнате и получать потоки, опубликованные в комнате

Видеоконференция

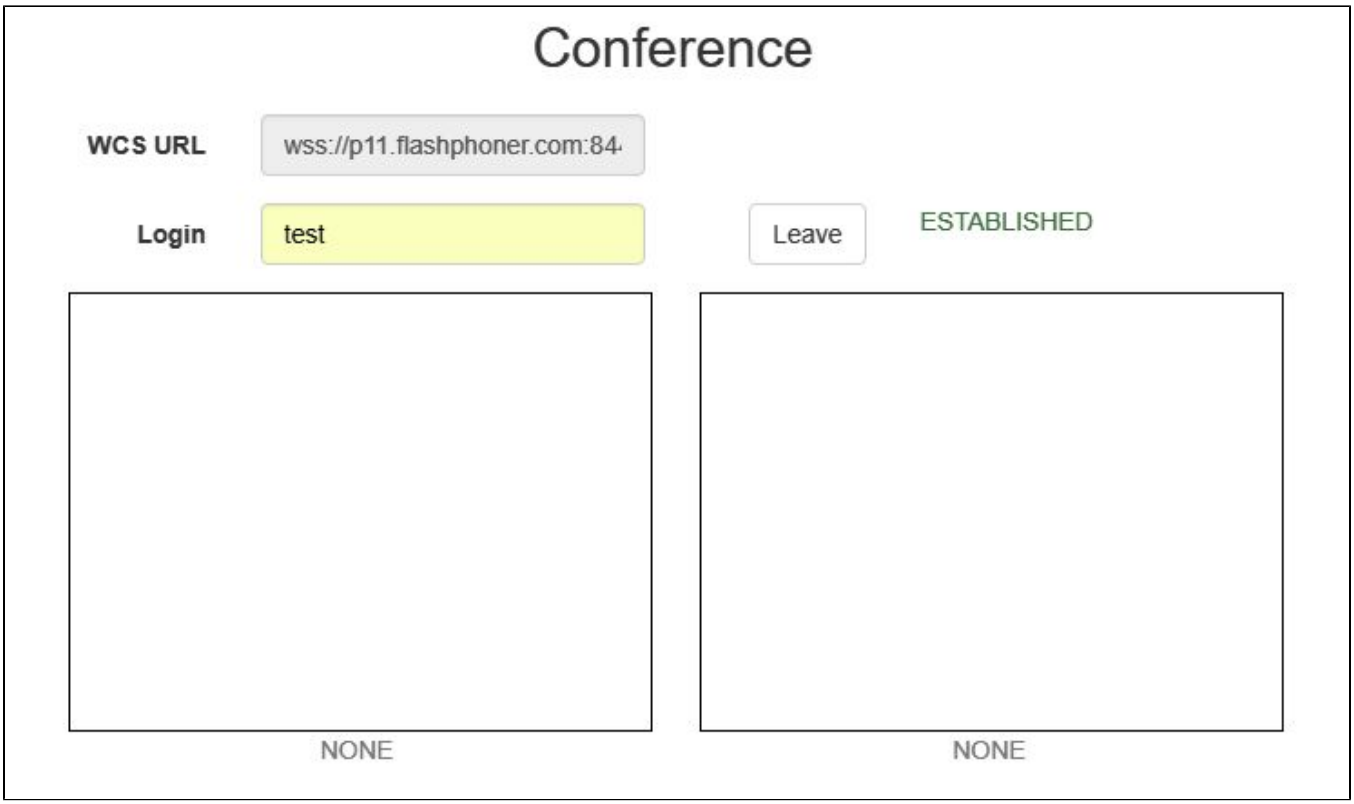
1. Для теста используем:

- демо-сервер demo.flashphoner.com;
- веб-приложениеConferenceдля организации видеоконференции

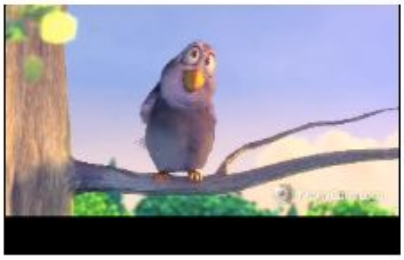
2. Откройте веб-приложение Conference. В поле "Login" введите произвольное имя пользователя, например test:



3. 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:



PUBLISHING

Mute A | Mute V

Stop

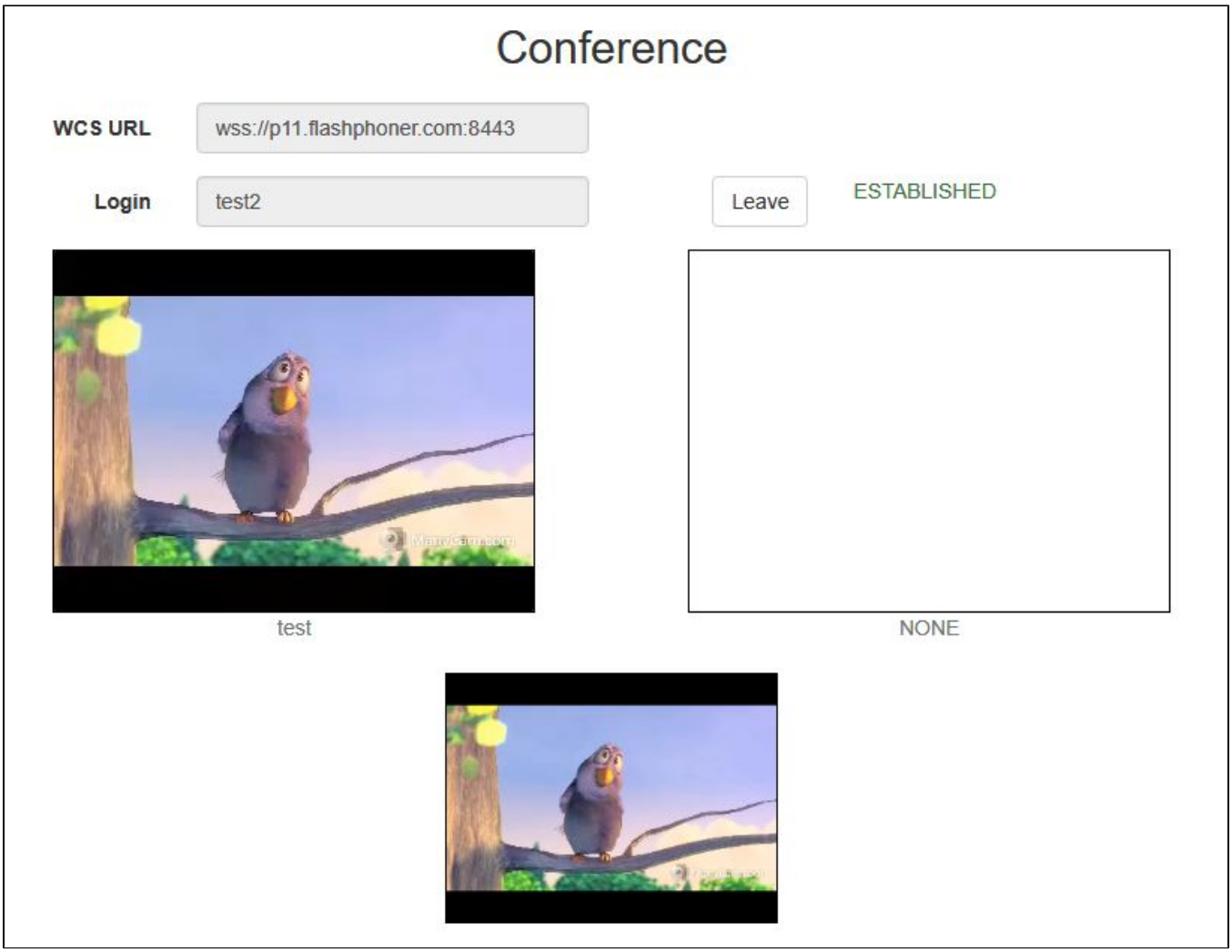
10:29 chat - room is empty
10:30 test - test message

Send

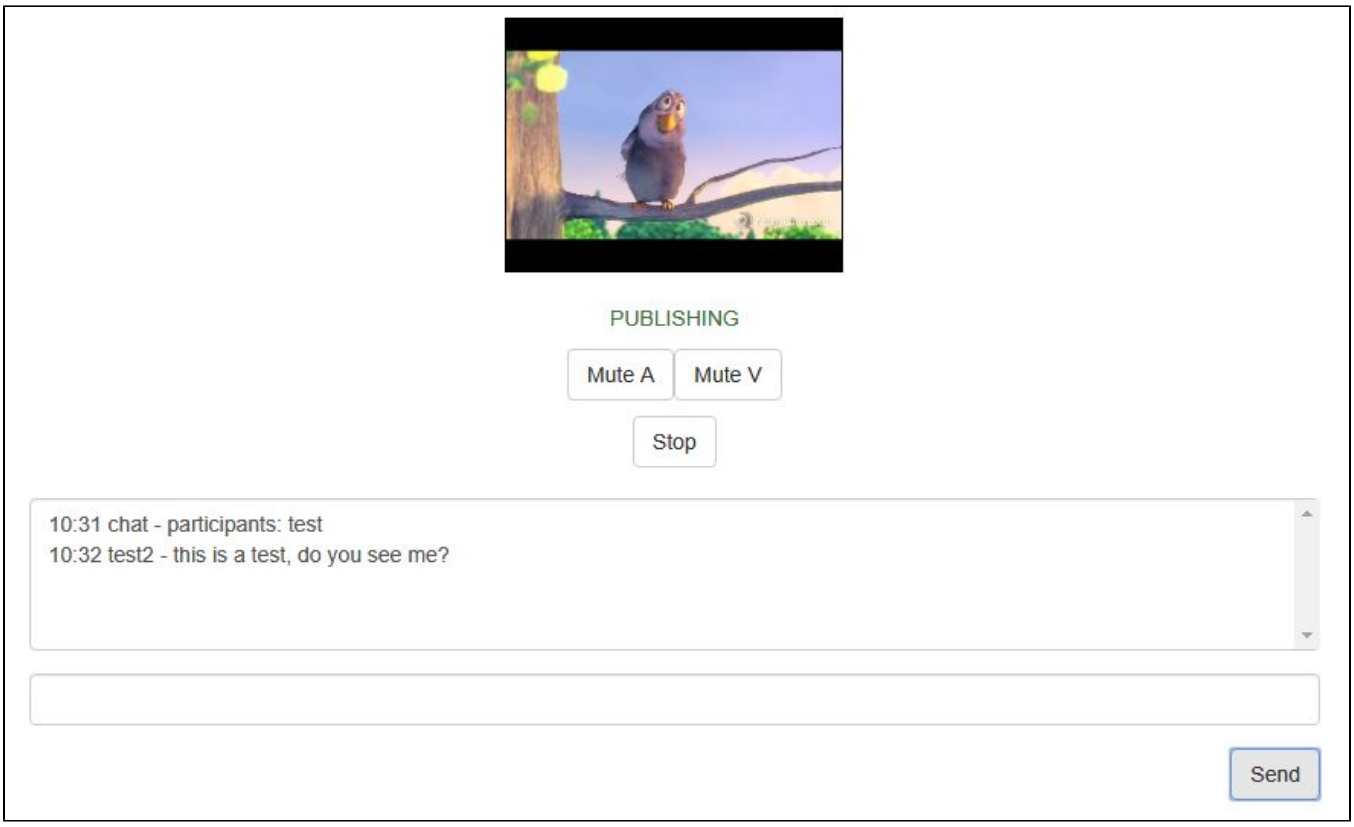
Invite

https://p11.flashphoner.com:8888/client2/examples/demo/streaming/conference/conference.html?roomName=room-72be4a

4. 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):



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



6. На вкладке участника test введите ответное сообщение:



PUBLISHING

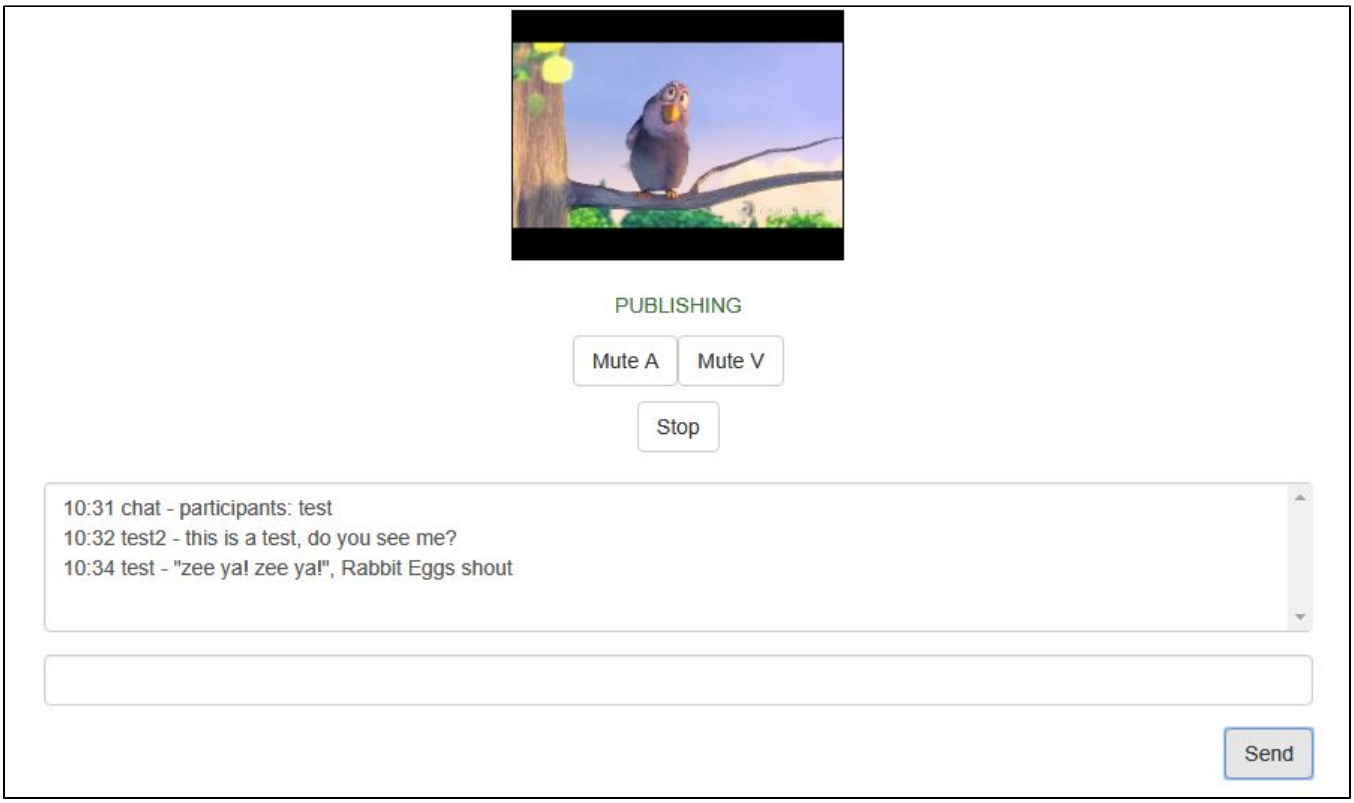
Mute A | Mute V

Stop

10:29 chat - room is empty
10:30 test - test message
10:31 test2 - joined
10:32 test2 - this is a test, do you see me?
10:34 test - "zee ya! zee ya!", Rabbit Eggs shout

Send

7. Убедитесь, что ответ получен:



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

Видеочат

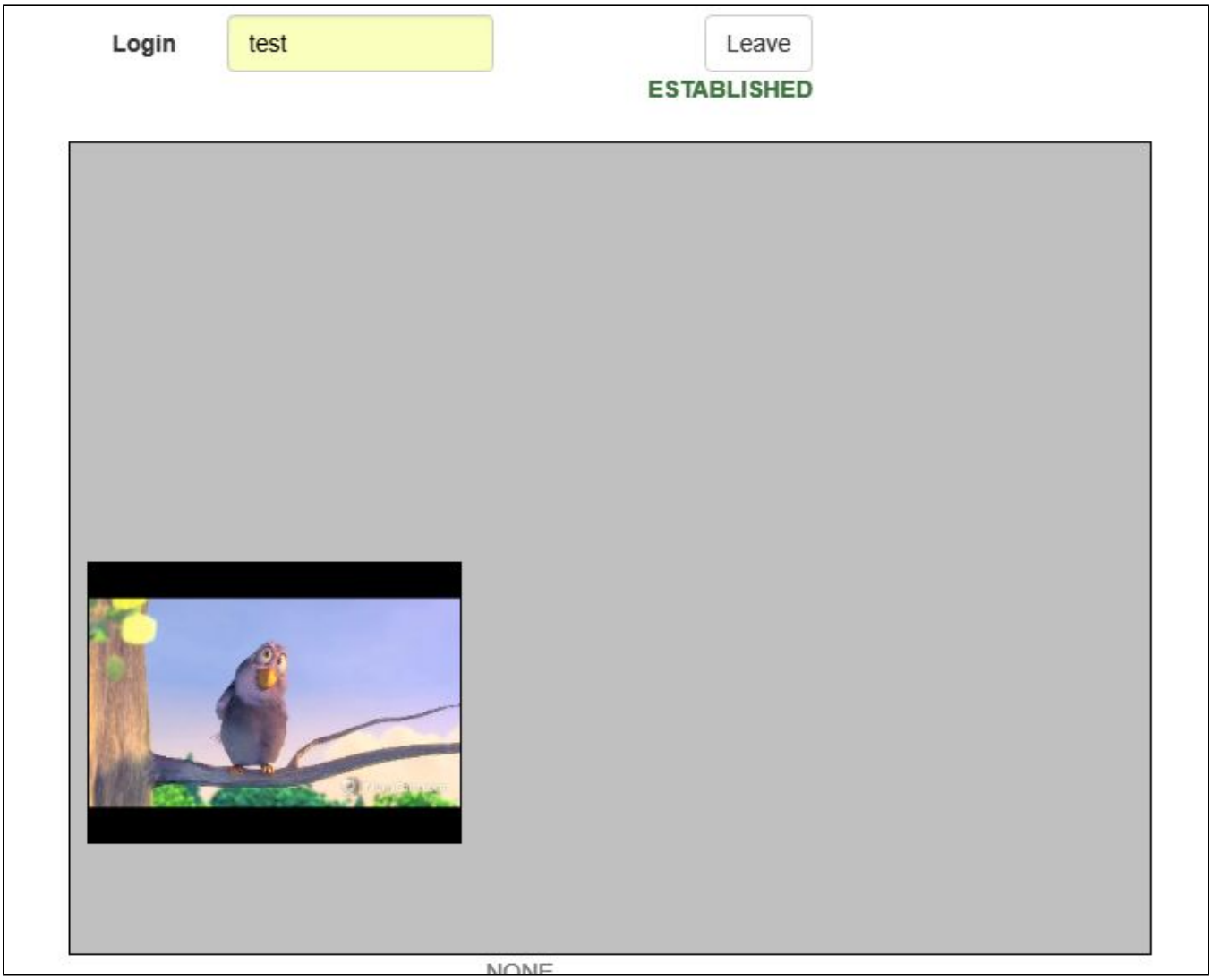
1. Для теста используем:

- демо-сервер demo.flashphoner.com;
- веб-приложениеTwo Way Video Chatдля организации видеочата

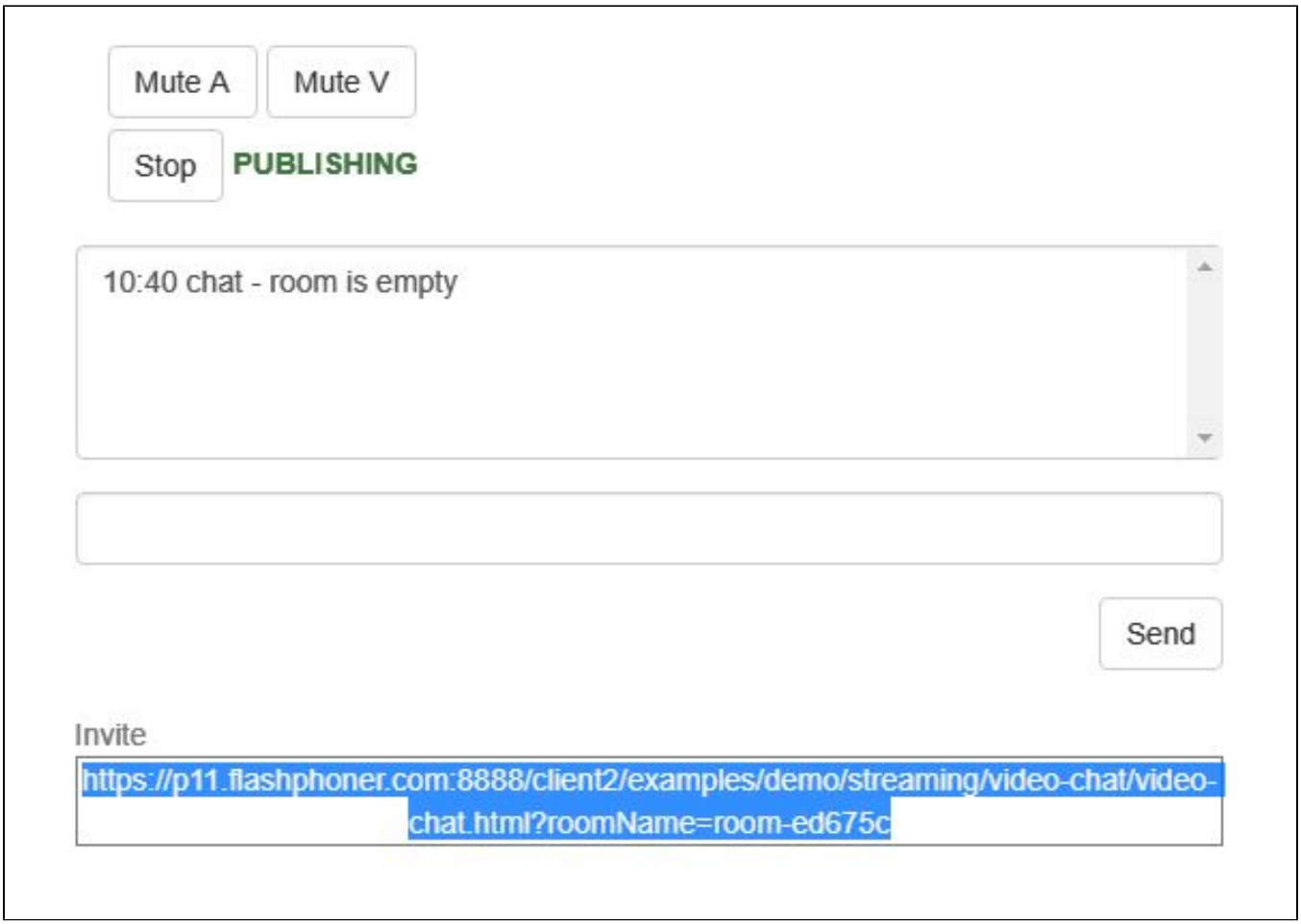
2. Откройте веб-приложение Two Way Video Chat. В поле "Login" введите произвольное имя пользователя, например test:



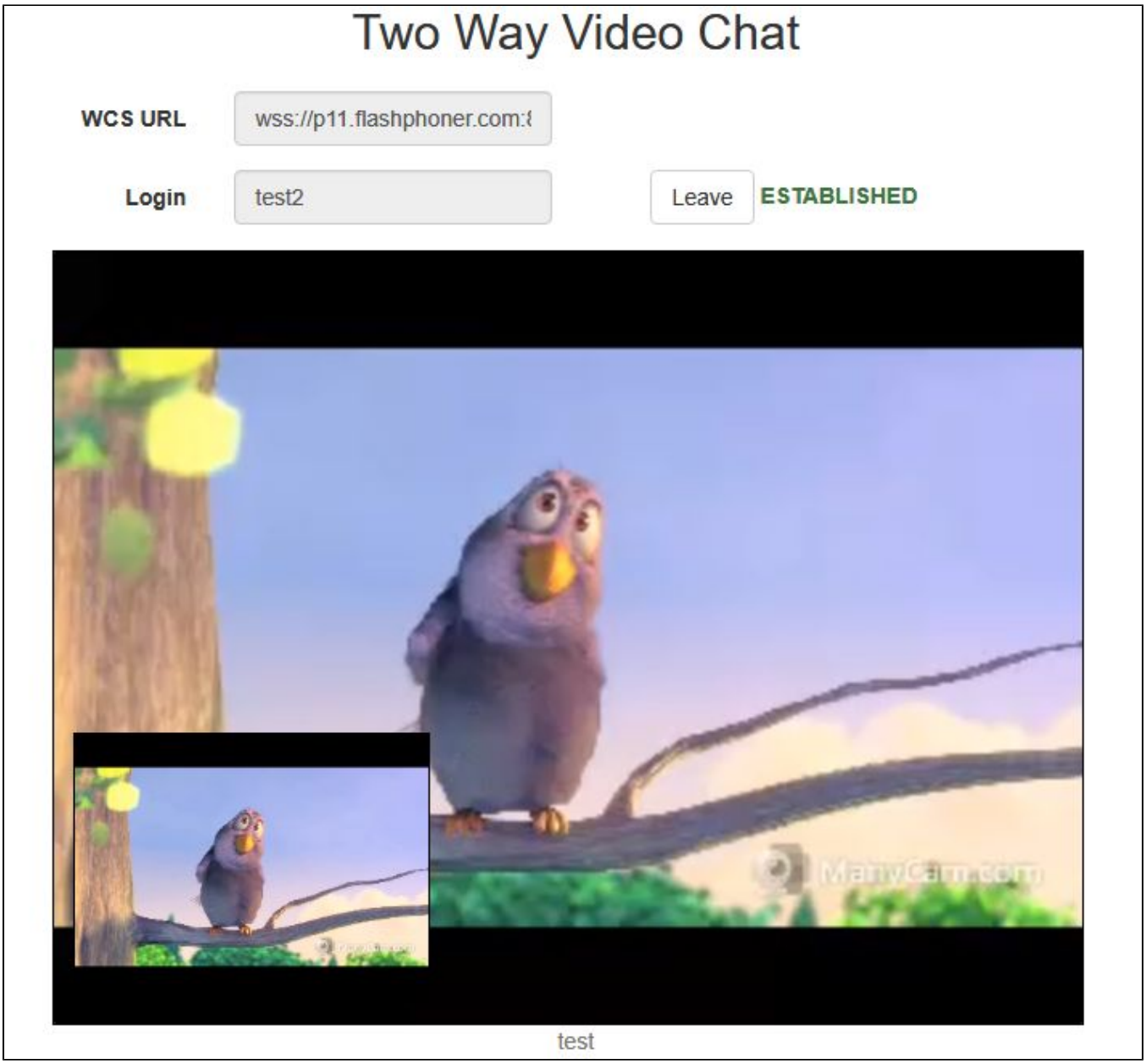
3. Нажмите кнопку Join. Установится соединение с сервером, отобразится надпись "ESTABLISHED", будет автоматически создана чат-комната. Будет показано изображение с веб-камеры:



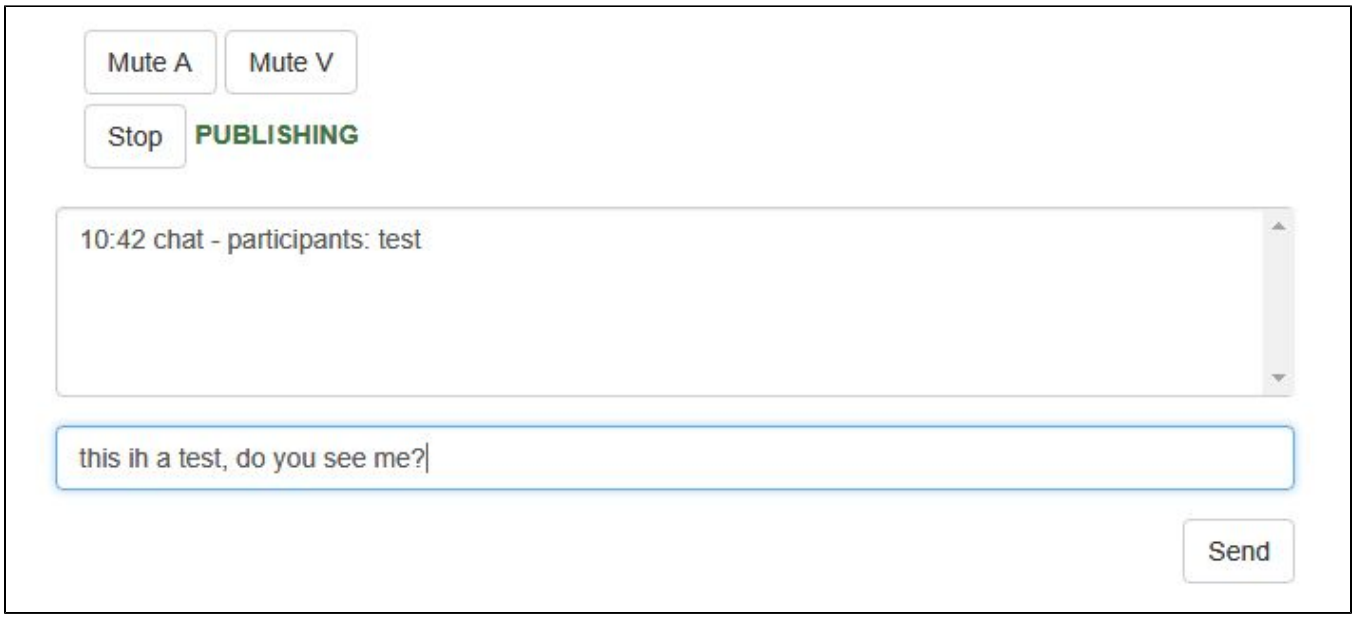
Внизу экрана отобразится текстовый чат и ссылка на комнату для приглашения других пользователей:



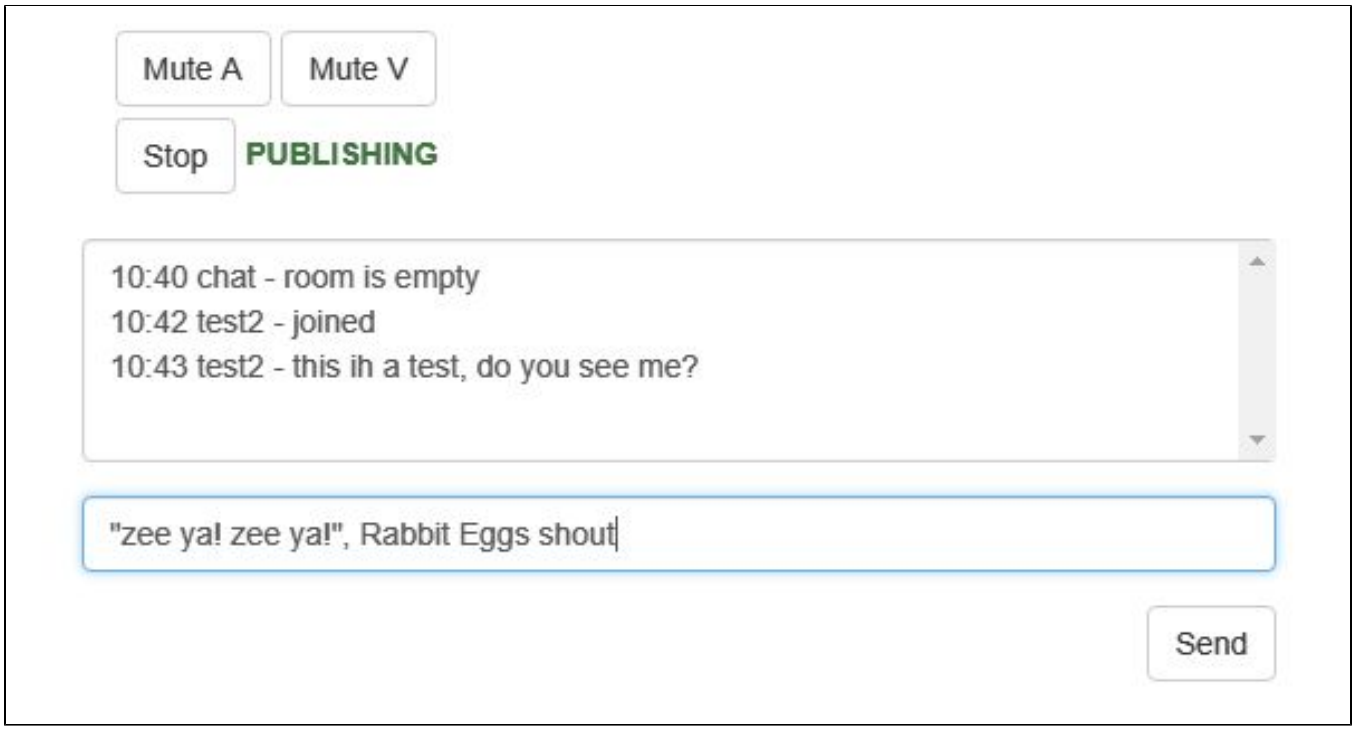
4. Скопируйте ссылку на чат-комнату и откройте ее в другой вкладке браузера. Введите имя пользователя, которое должно отличаться от имени создателя чат-комнаты, например, test2, и нажмите кнопку Join. На странице будет показано изображение с веб-камеры пользователя test крупно и с веб-камеры пользователя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



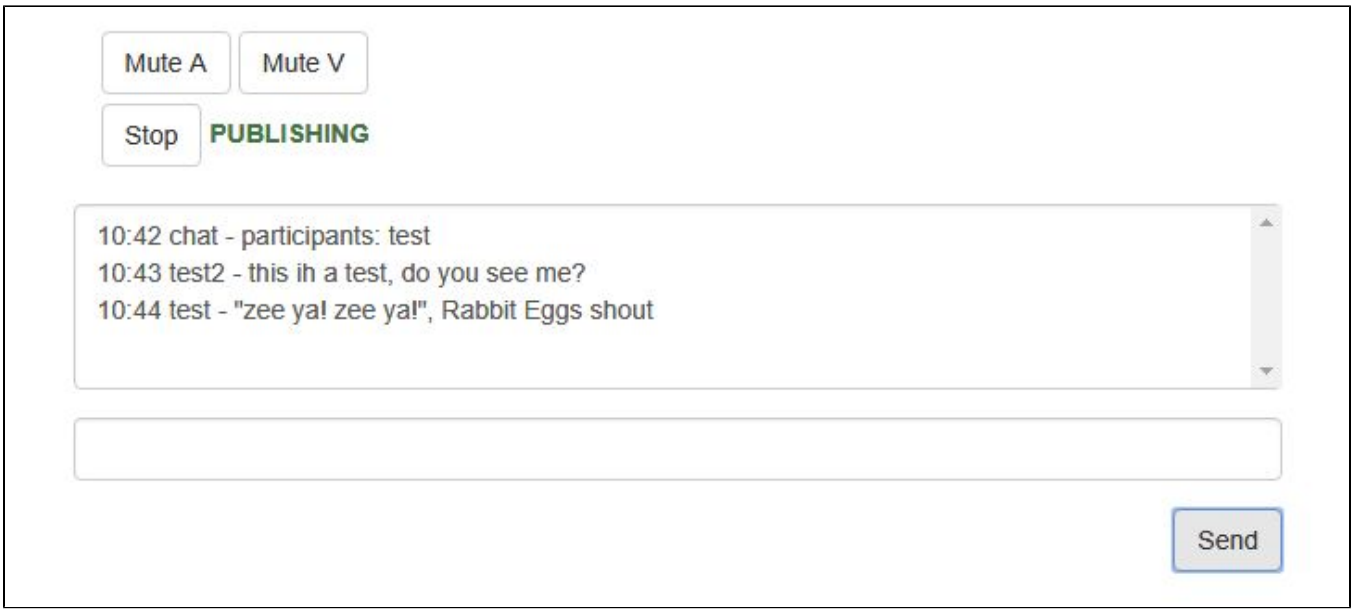
5. Введите в окне текстового чата сообщение и нажмите кнопку Send:



6. На вкладке пользователя test введите ответное сообщение:



7. Убедитесь, что ответ получен:



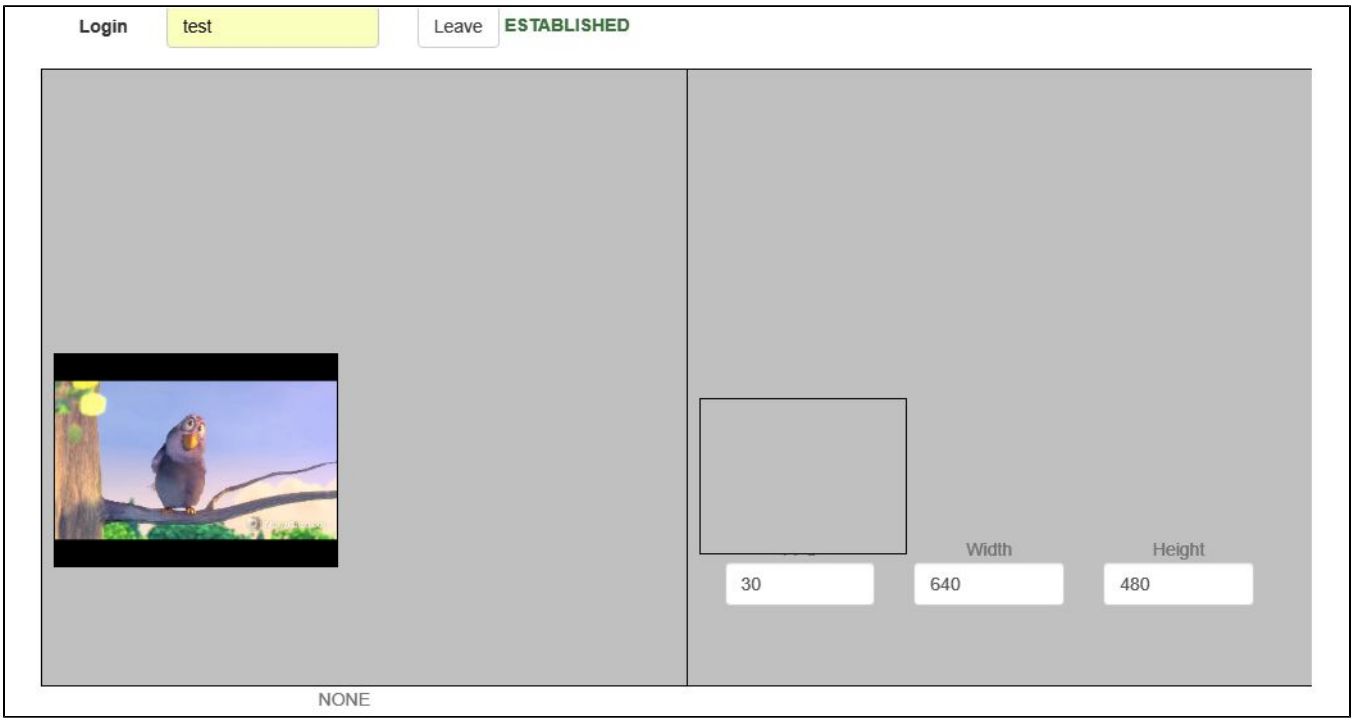
8. Для выхода из-чат-комнаты нажмите кнопку Leave.

Видеоконференция с отображением экрана компьютера (screen sharing)

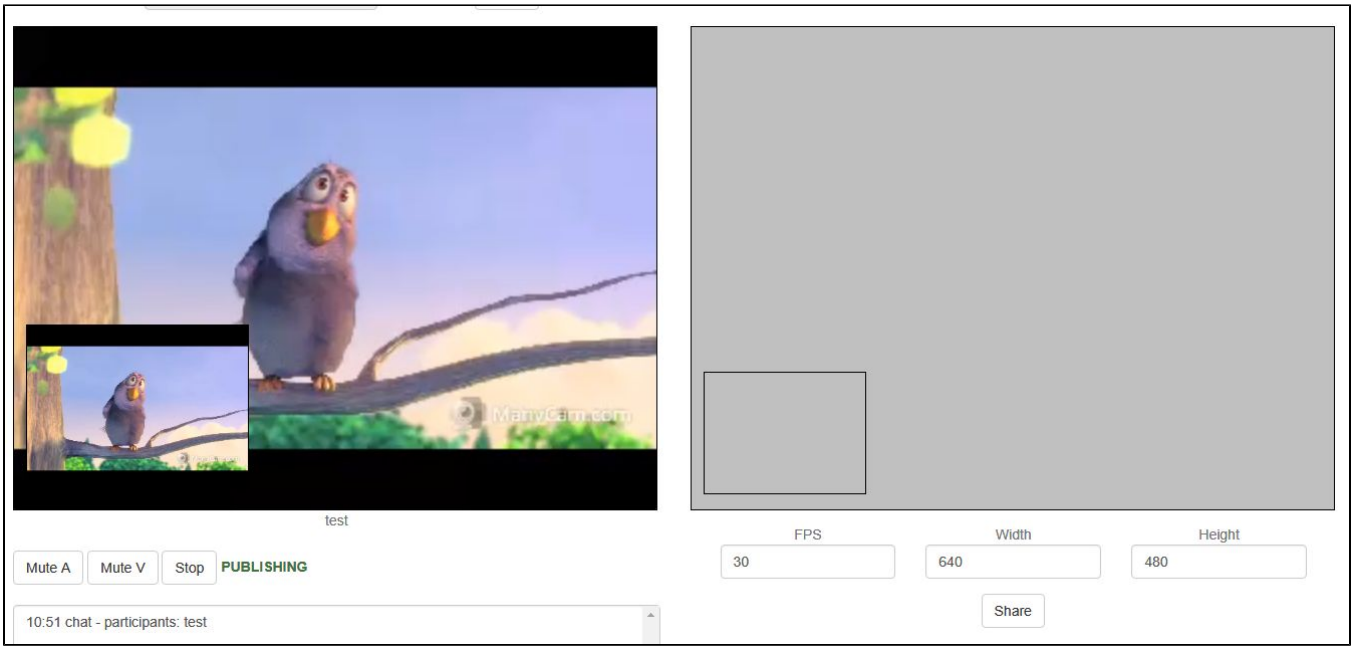
1. Для теста используем:

- демо-сервер demo.flashphoner.com;
- веб-приложениеTwo Way Video Chat and Screenдля организации видеоконференции;
- браузер Chrome.

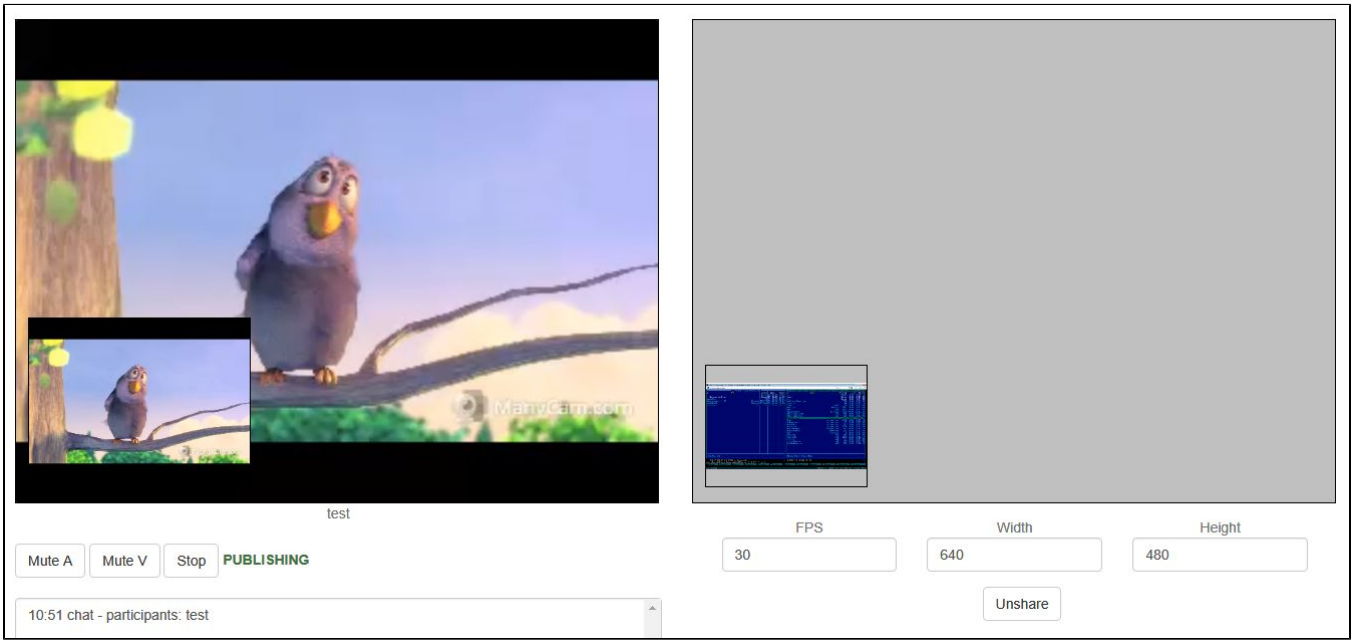
2. Откройте веб-приложение "Two Way Video Chat & Screen". Если активна кнопка Install Now, нажмите ее и установите расширение. В поле "Login" введите произвольное имя пользователя, например test. Нажмите кнопку Join. Установится соединение с сервером, отобразится надпись "ESTABLISHED", будет автоматически создана чат-комната. Будет показано изображение с веб-камеры:



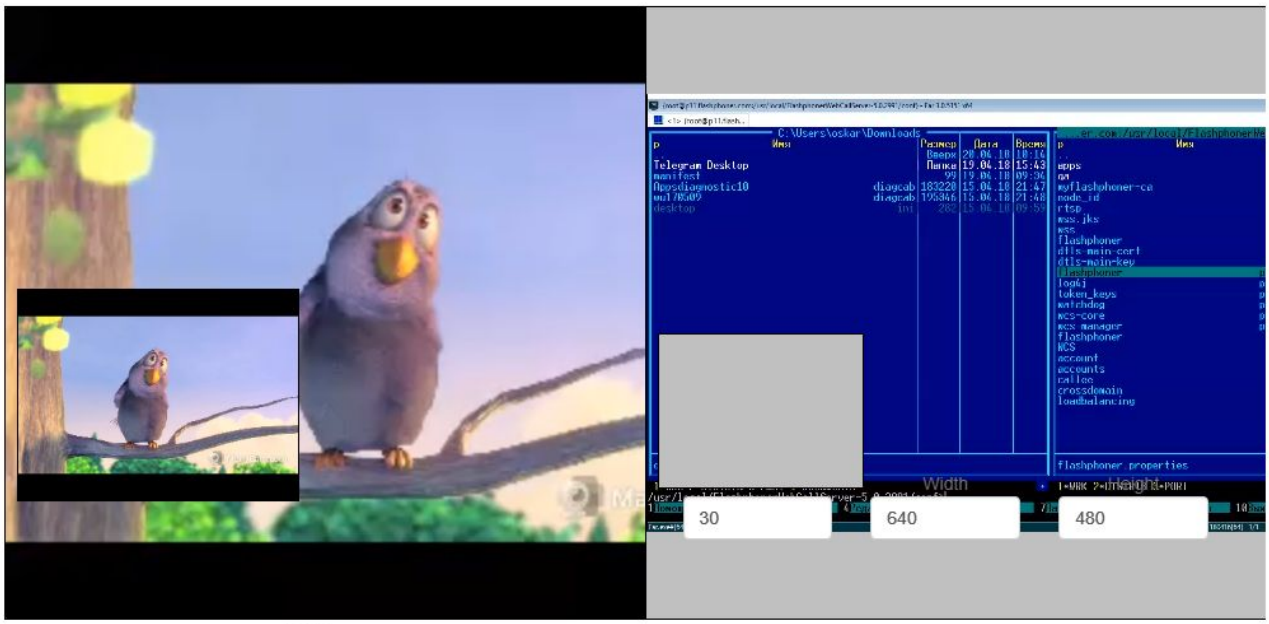
3. Скопируйте ссылку на чат-комнату и откройте ее в другой вкладке браузера. Введите имя пользователя, которое должно отличаться от имени создателя чат-комнаты, например, test2, и нажмите кнопку Join. На странице будет показано изображение с веб-камеры:



4. Нажмите кнопку "Share" и разрешите браузеру доступ к экрану или к окну приложения:



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



6. Для выхода из-чат-комнаты нажмите кнопку "Leave".

Последовательность выполнения операций (Call Flow)

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.

ConnectionStatusEvent ESTABLISHEDcode

```
    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.

connection.join();code

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

4. Получение участником 1 от сервера события, описывающего состояние комнаты.

RoomStatusEvent 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.

StreamStatusEvent PUBLISHINGcode

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

RoomStatusEvent JOINEDcode

```
    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.

RoomStatusEvent LEFTcode

```
    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,
        receiveVideo: false,
        receiveAudio: false
        ...
    });
```

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

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.

По умолчанию, синхронизация потоков отключена. В этом случае для объединения потоков участников ихможно направлять в микшер во время публикации.



Начиная с версии 5.2.142 данная возможность не поддерживается. Для объединения потоков участников их необходимонаправлять в микшер во время публикации.

Тестирование записи потоков с синхронизацией

1. Для теста используем:

- ваш WCS сервер, например test2.flashphoner.com;
- веб-приложениеConference

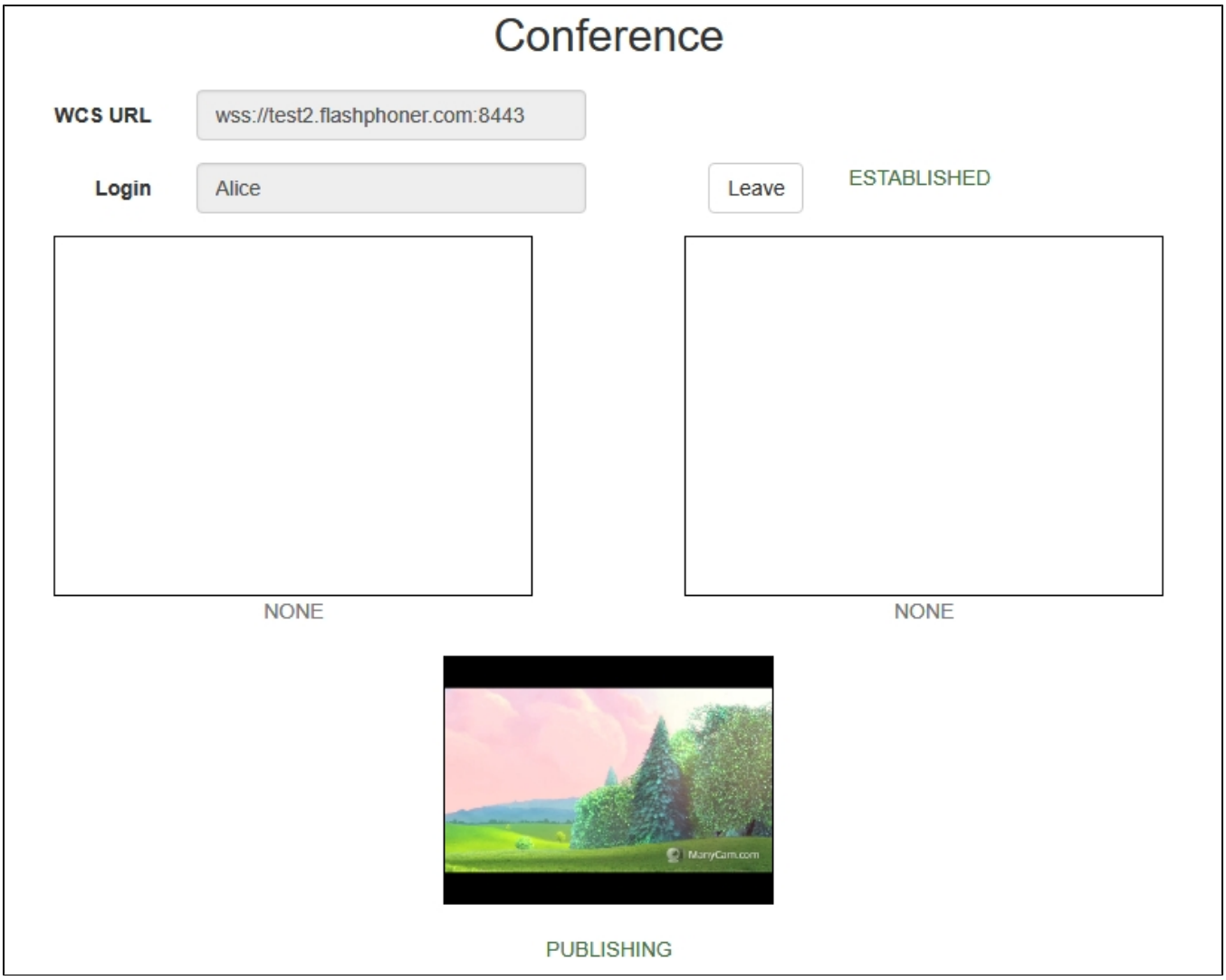
2. Включите запись потоков в веб-приложении Conference

3. Включите синхронизацию потоков, опубликованных в комнате

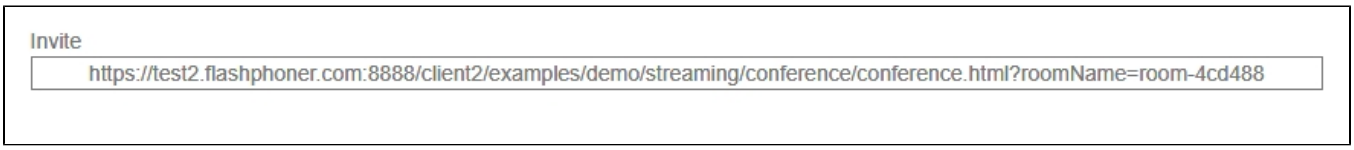
```
enable_empty_shift_writer=true
```

Перезапустите WCS.

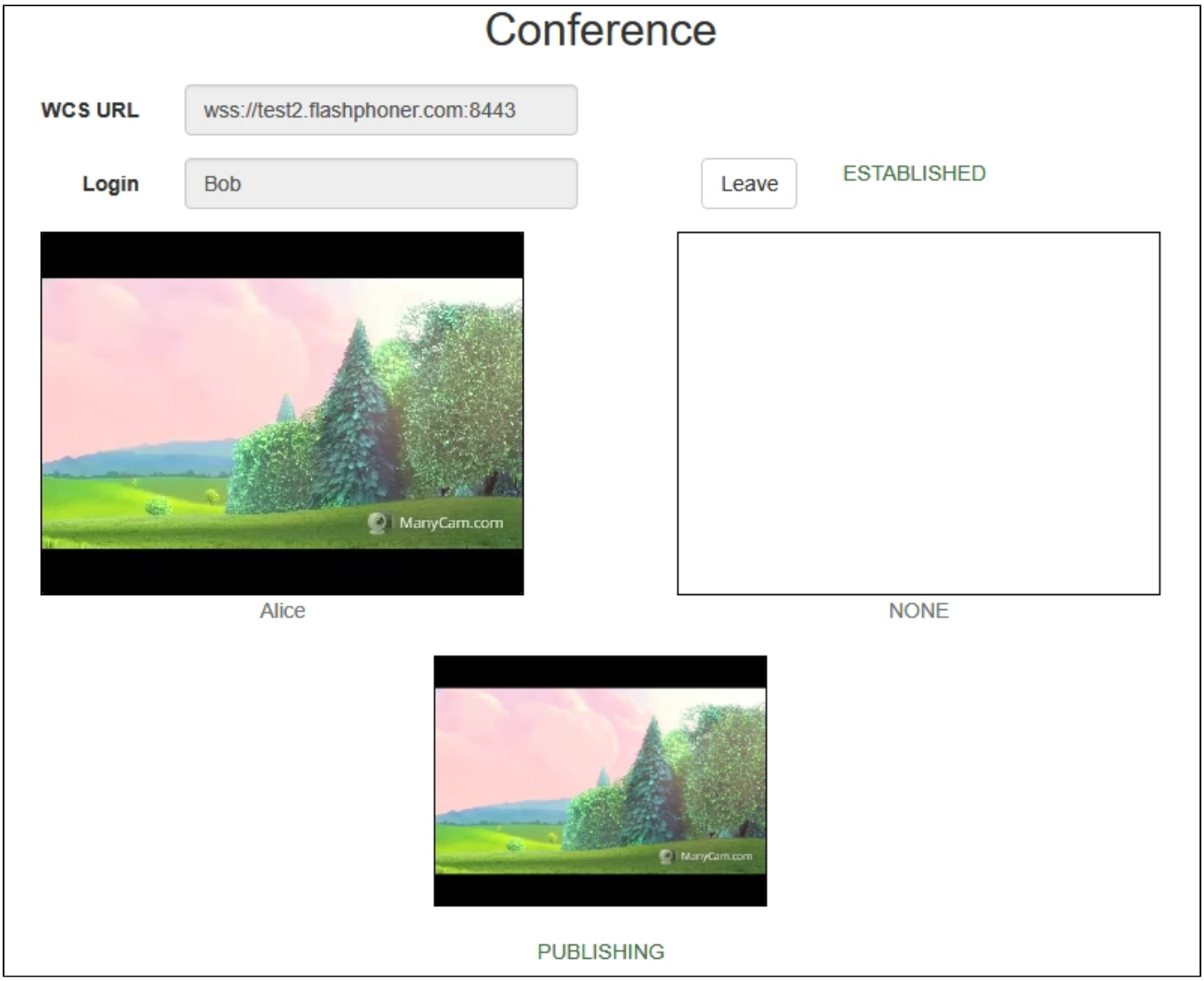
4. Откройте веб-приложение Conference. В поле "Login" введите имя пользователя Alice и нажмите 'Join'. Опубликуется поток от пользователя Alice:



5. Скопируйте ссылку из поля "Invite":



6. В новом окне браузера перейдите по данной ссылке.В поле "Login" введите имя пользователя Bob и нажмите 'Join'. Отобразится поток от пользователя Alice, и опубликуется поток от пользователя Bob:



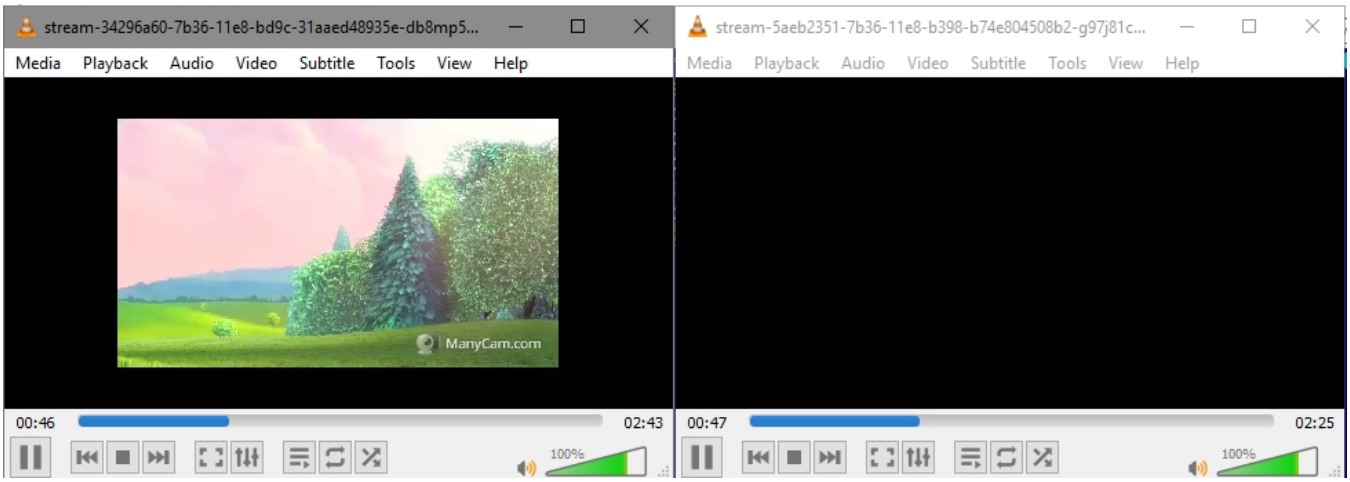
7. Нажмите 'Leave' на вкладке пользователя Bob, чтобы выйти из комнаты. Нажмите 'Leave' на вкладке пользователя Alice, чтобы завершить конференцию.

8. В каталоге/usr/local/FlashphonerWebCallServer/records располагаются файлы записи потоков:

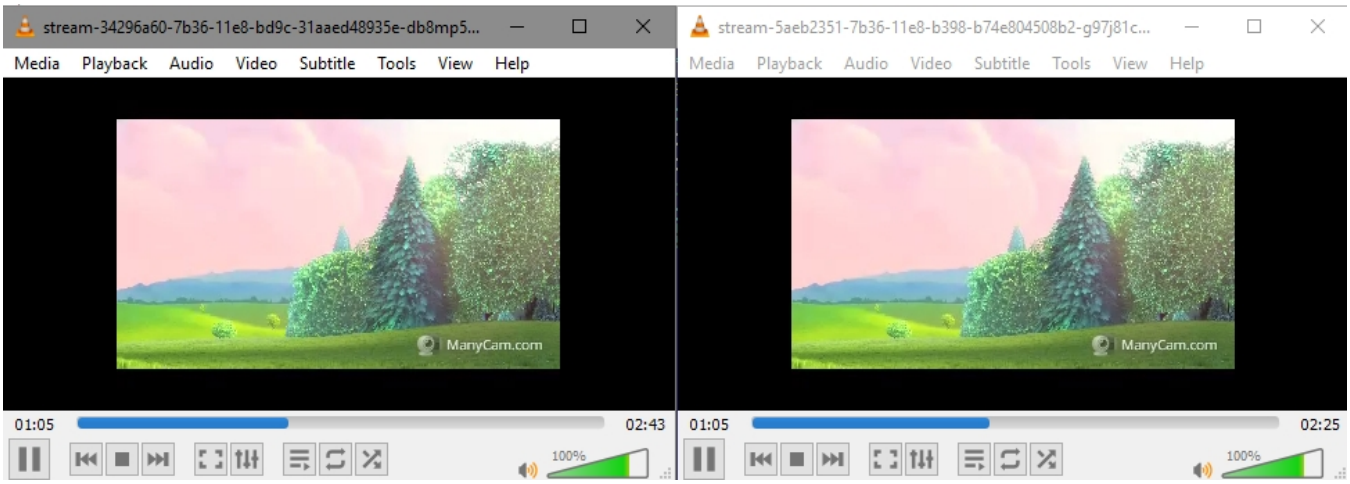
```
-rw-r--r-- 1 root root 1569566 Jun 29 07:51 stream-34296a60-7b36-11e8-bd9c-31aaed48935e-
db8mp51bajcidn9qmcnda3967k.mp4
-rw-r--r-- 1 root root 516509 Jun 29 07:51 stream-5aeb2351-7b36-11e8-b398-b74e804508b2-
g97j81cgrf8h1m7jl7184fa788.mp4
```

Файл от пользователя Bob меньшего размера, поскольку в начале файла идет пустое видео для синхронизации.Загрузите файлы на ПК и воспроизведите.

9. Поток от Alice опубликован, Bob еще не вошел в комнату



10. Опубликован поток от Bob



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
```

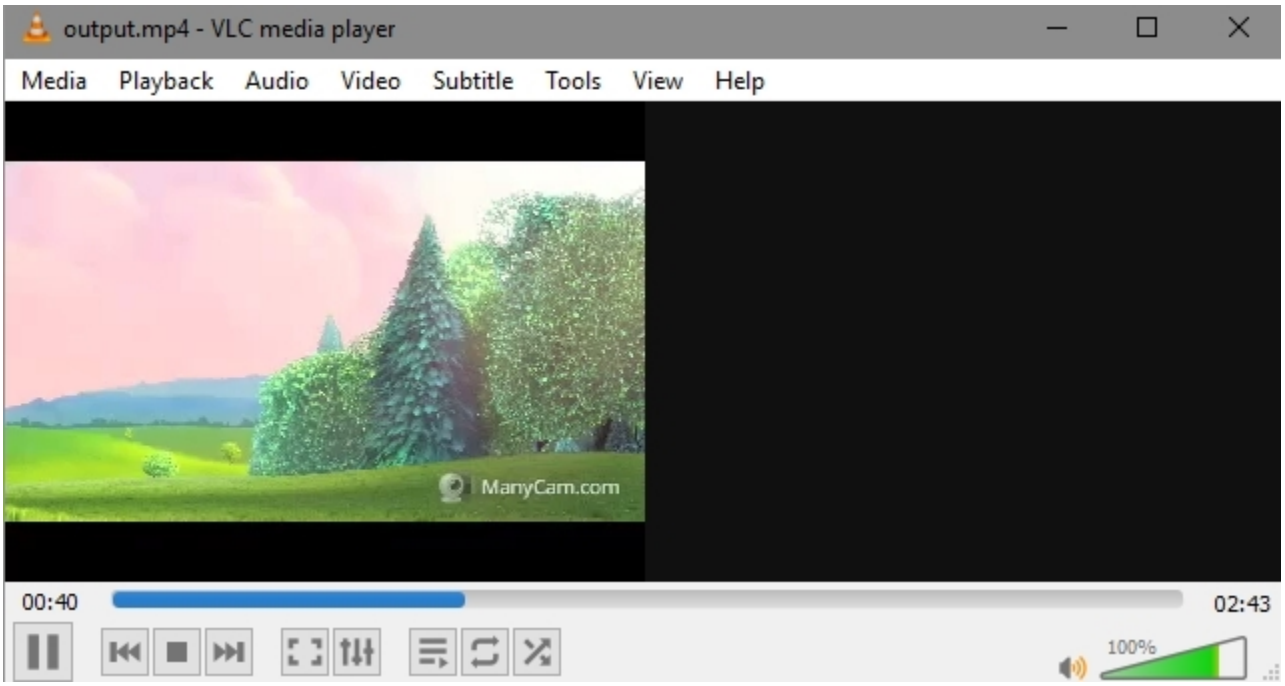
Здесь

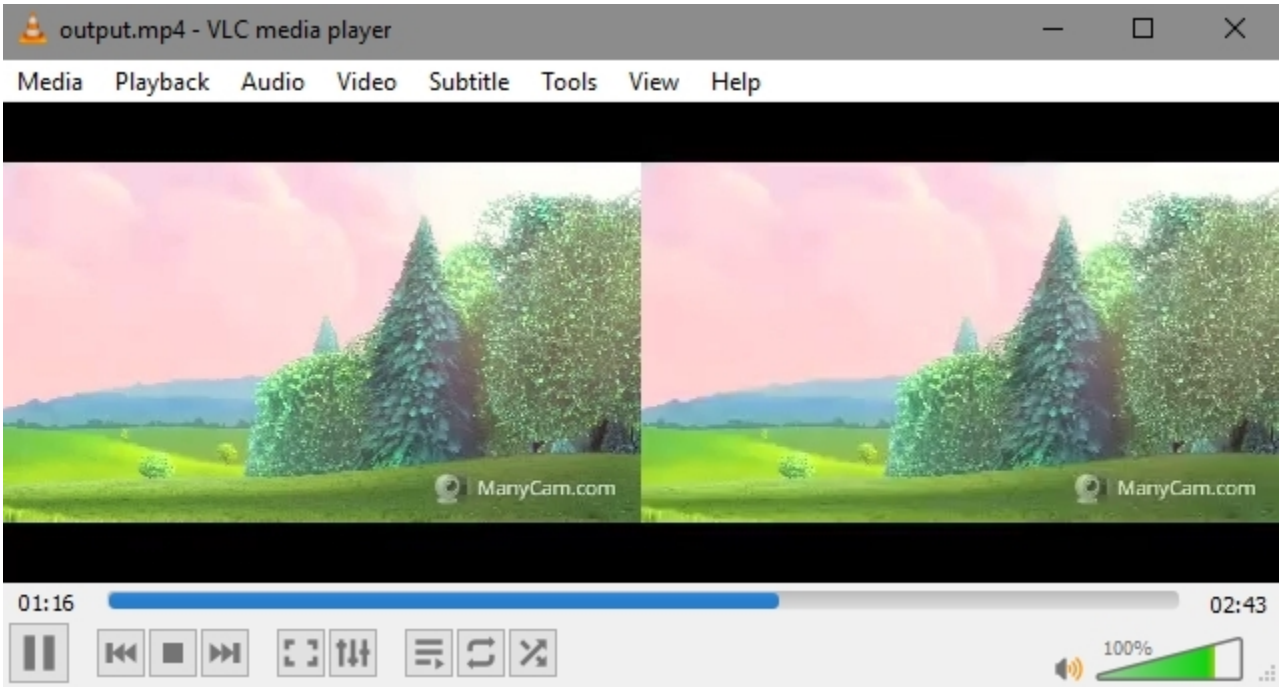
- stream1 - поток первого участника
- stream2 - поток второго участника

Для того, чтобы объединить файлы, полученные в ходе тестирования, введите команду:

```
ffmpeg -i stream-34296a60-7b36-11e8-bd9c-31aaed48935e-db8mp51bajcidn9qmcnda3967k.mp4 -i stream-5aeb2351-7b36-
11e8-b398-b74e804508b2-g97j81cgrf8h1m7jl7184fa788.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
```

Воспроизведите файл output.mp4:





Запись потоков комнаты в один файл с последующим микшированием

В сборке WCS5.2.1012и сборке WebSDK 2.0.190добавлена 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

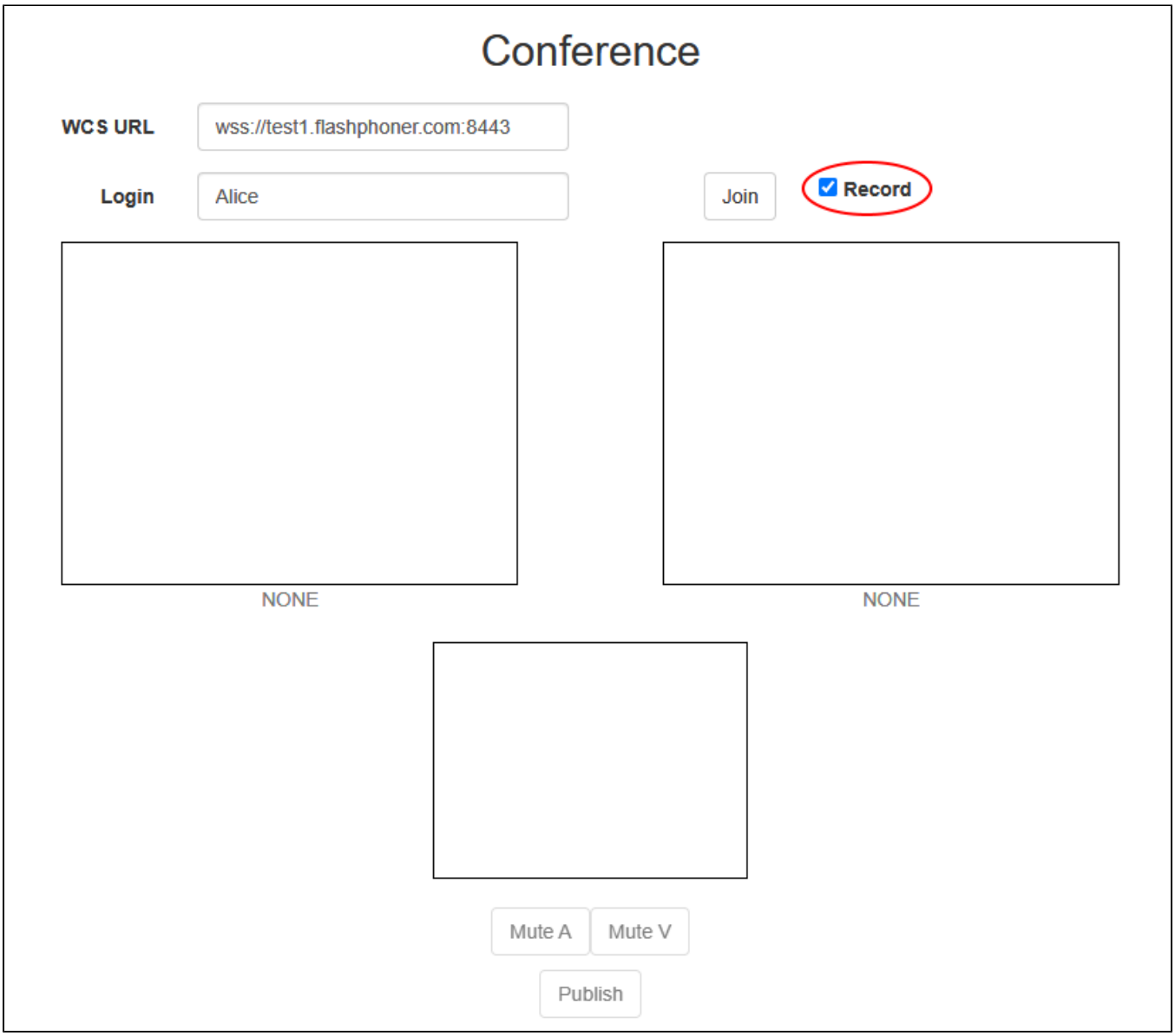
```
{
  "hasVideo": "true",
  "hasAudio": "true",
  "mixerDisplayStreamName": true
}
```

Тестирование

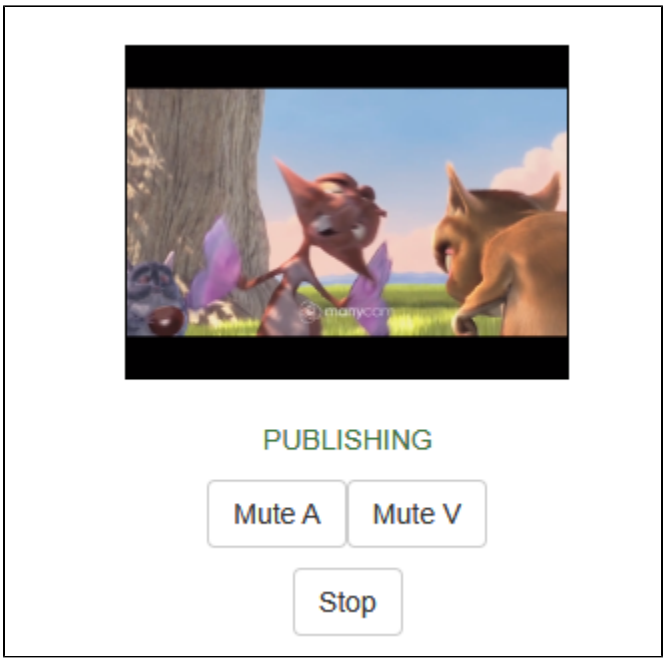
1. Для теста используем:

- ваш WCS сервер, например test1.flashphoner.com;
- веб-приложениеConference

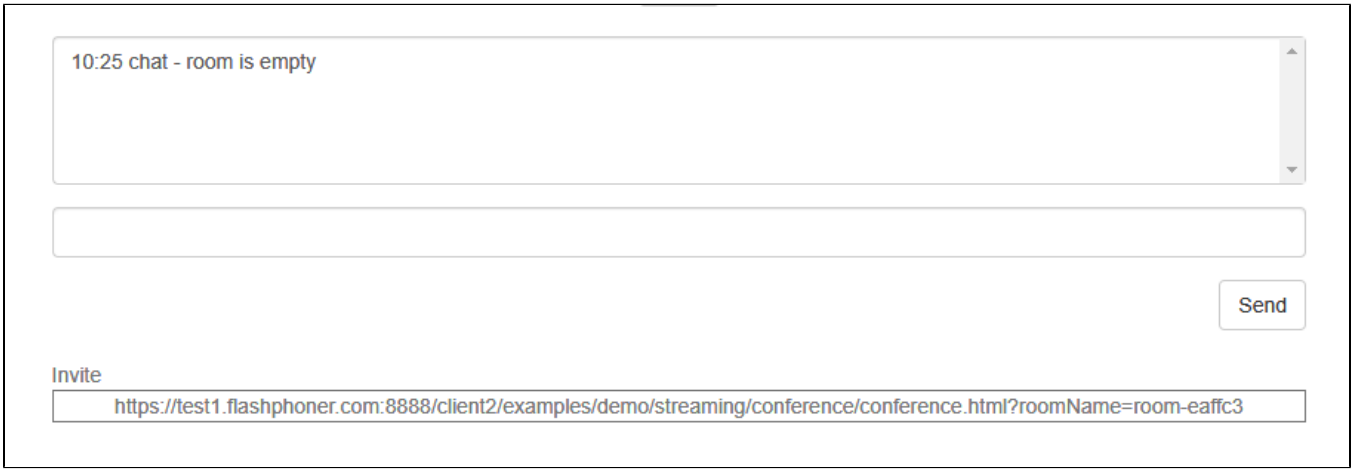
2. Откройте пример Conference в браузере, введите имя участника Alice и взведите переключатель Record



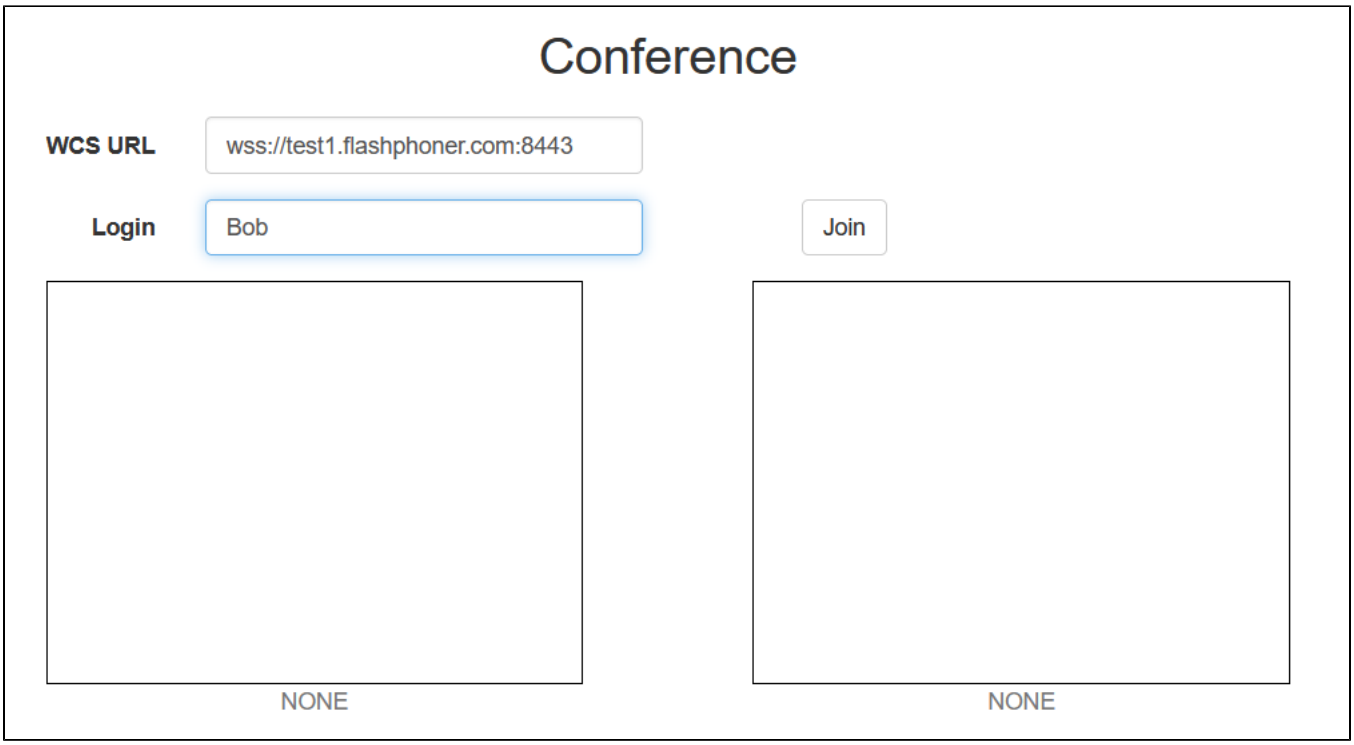
3. Нажмите Join. Начнется публикация потока



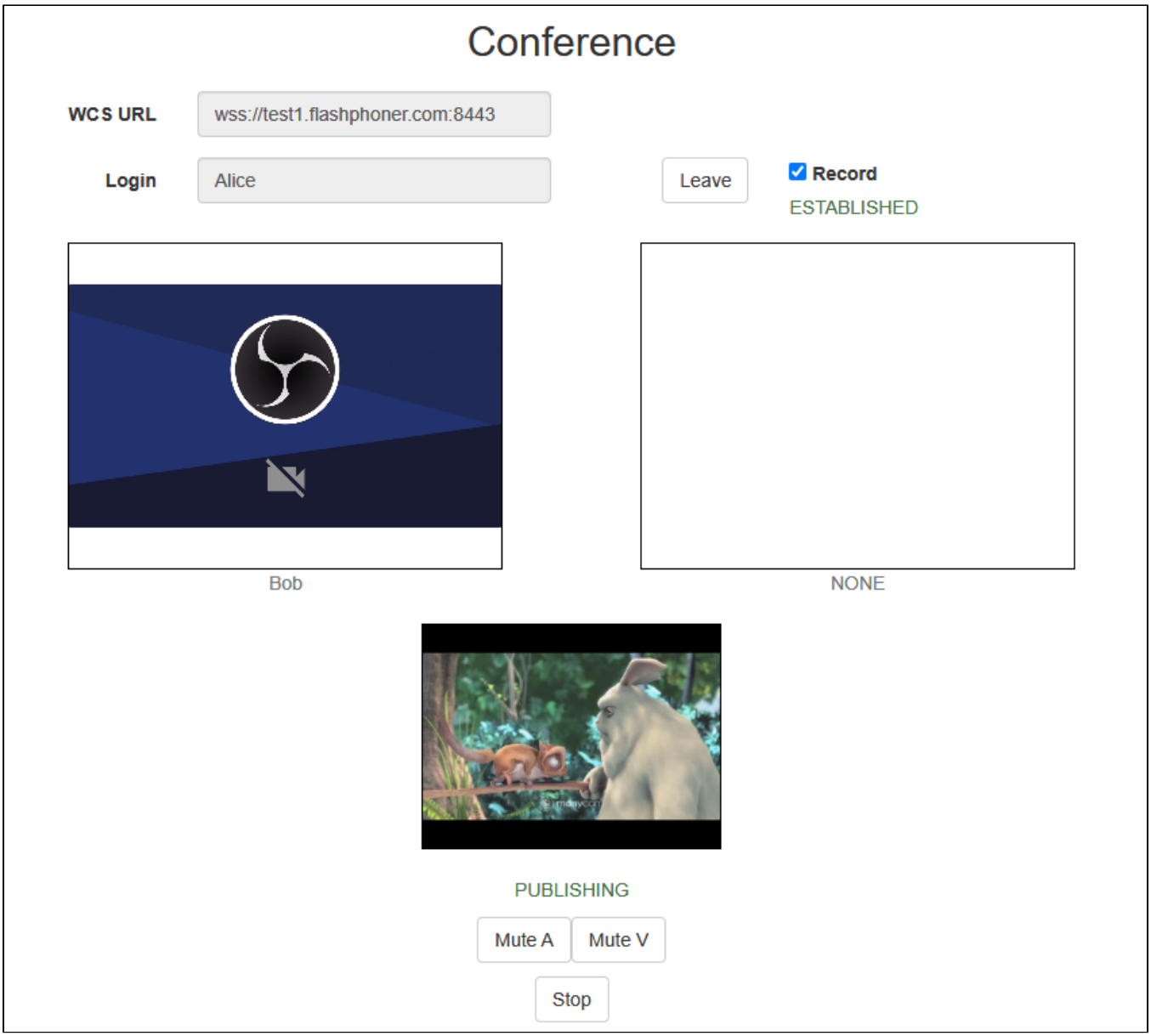
4. В другом окна браузера откройте ссылку из поля Invite



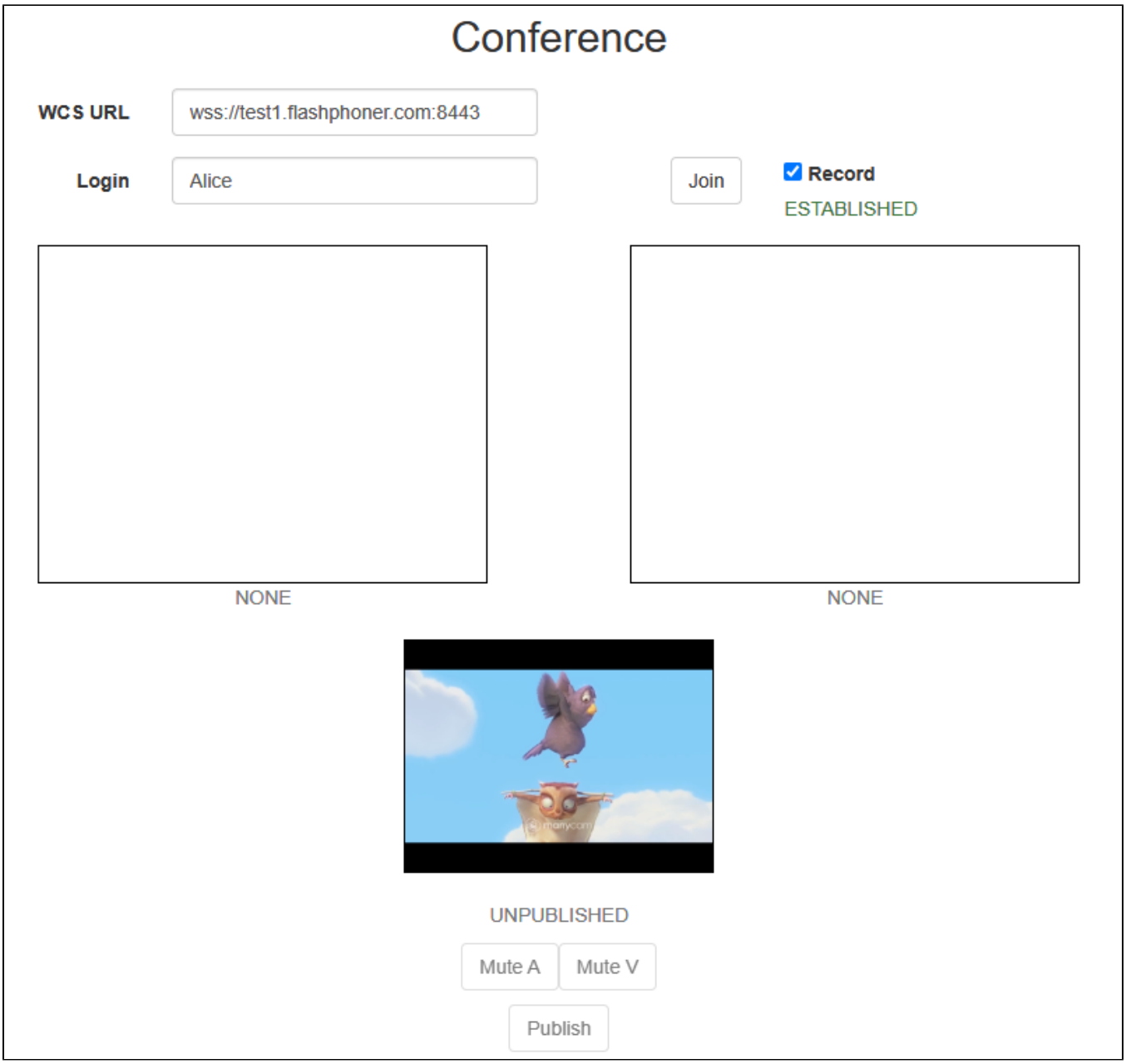
5. Введите имя пользователя Bob и нажмите Join



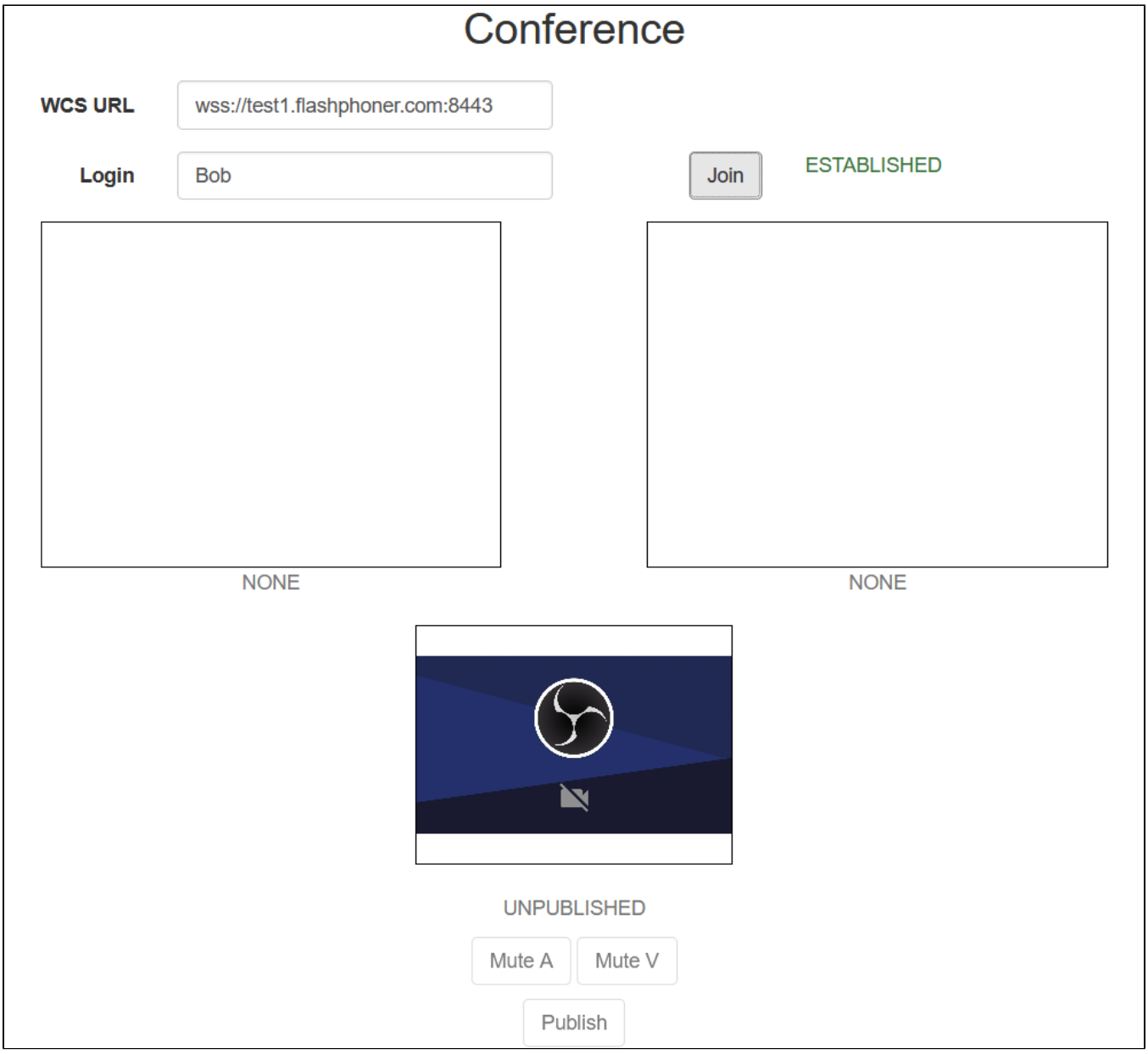
6. Bob присоединился к комнате



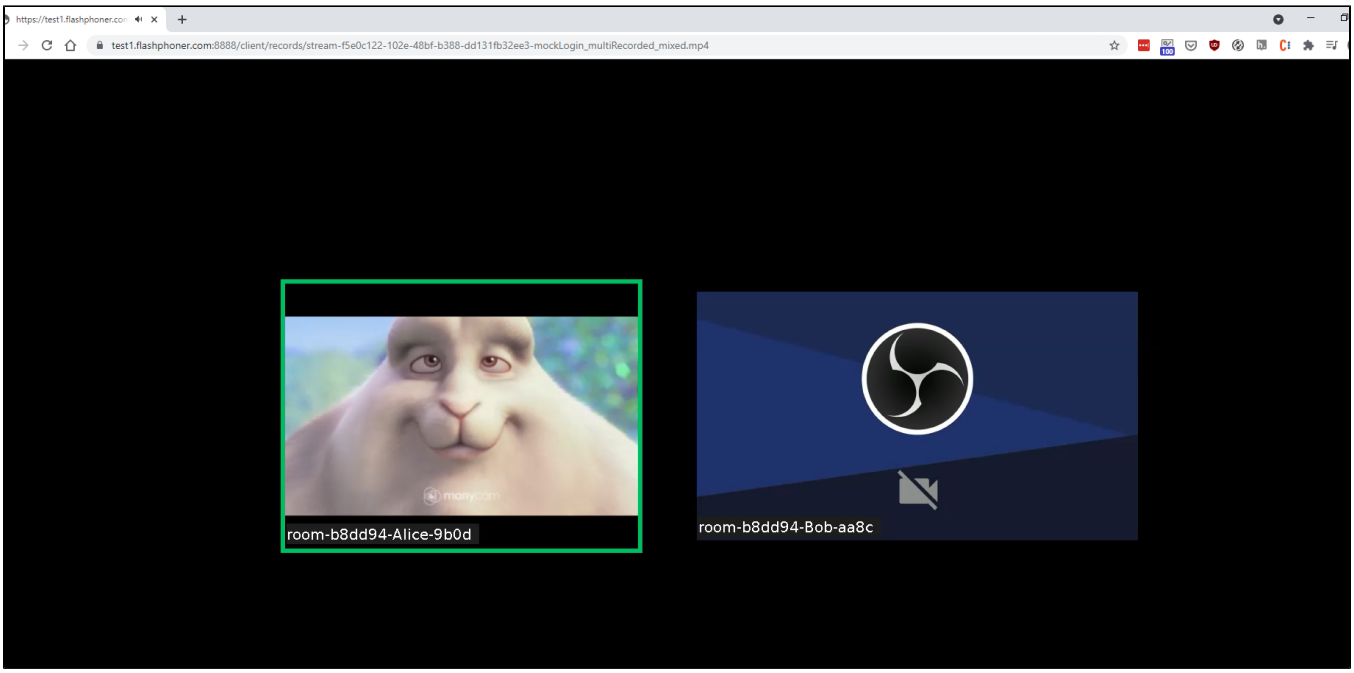
7. Нажмите Leave в окне пользователя Alice



и в окне пользователя Bob



8. 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 JavaScriptencodeURIComponent() при отправке сообщения

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

идecodeURIComponent() при его получении

```
    ...
    }).on(ROOM_EVENT.MESSAGE, function(message){
        addMessage(message.from.name(), decodeURIComponent(message.text));
    });
    ...
```

2. При быстром вызове connection.join() и затем room.leave() возможна отправка серверу команды join, в то время как сервер еще не обработал предыдущую команду leave для этого пользователя

Симптомы: при вызове connection.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 connection.join()