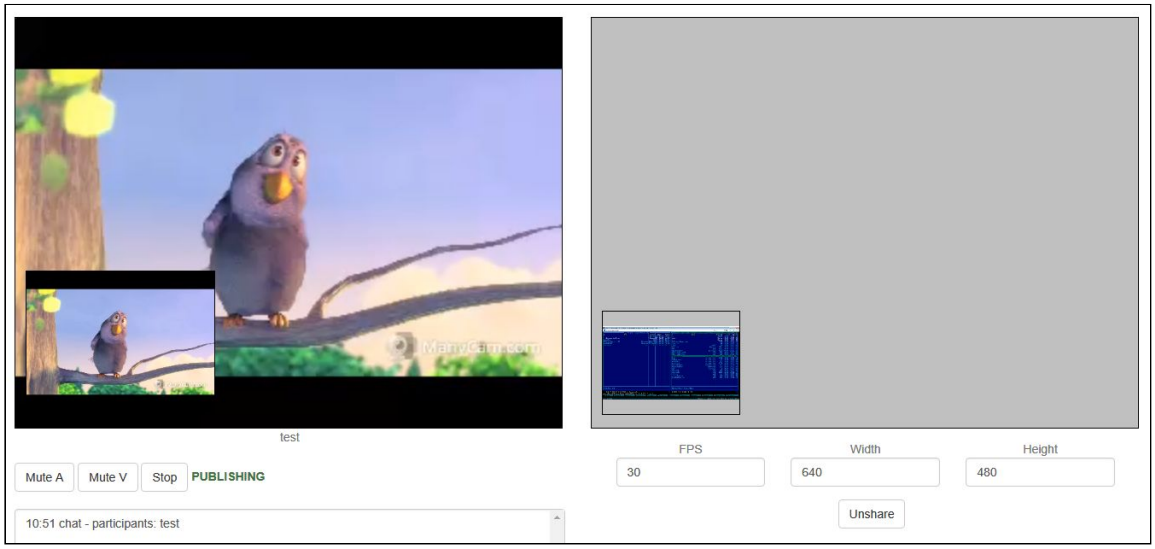
Video Chat and Screen Sharing

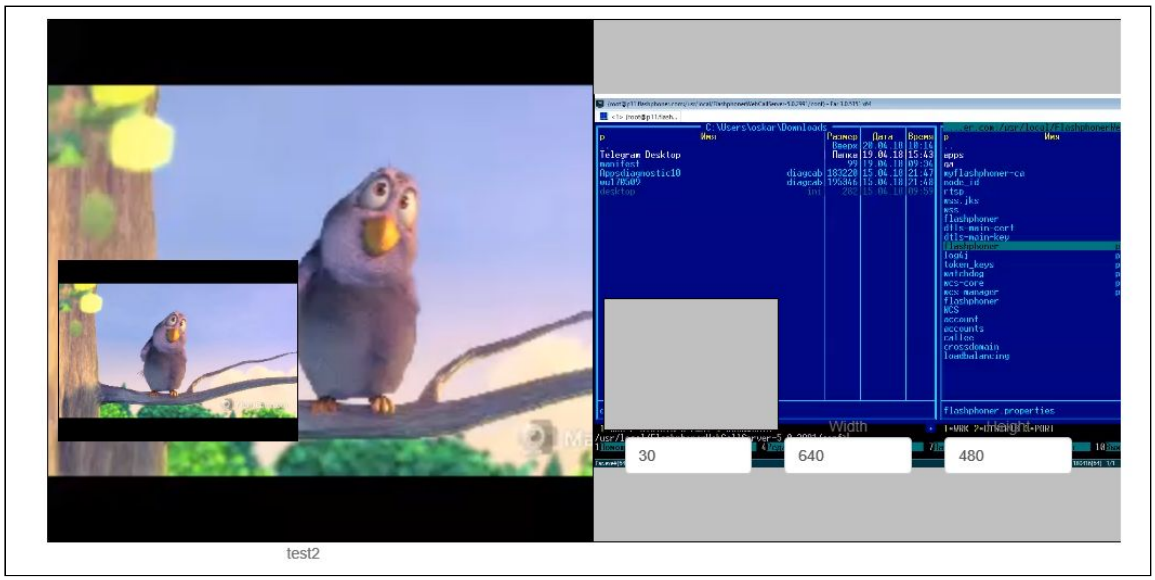
Пример видеочата с отображением экрана

Данный пример может использоваться для видеочата между двумя участниками на Web Call Server с отображением экрана одного из участников. Участник видеочата может публиковать WebRTC поток с веб-камеры и одновременно с этим WebRTC поток с экрана или из окна приложения

Пример окна клиента, публикующего свой экран в браузере Chrome:



Пример окна клиента, получающего поток с экрана в браузере Chrome



Код примера

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

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/video-chat-and-screen-sharing/

- video-chat.css - файл стилей
- video-chat-and-screen-sharing.html - страница участника чата
- video-chat-and-screen-sharing.js - скрипт, обеспечивающий работу чата

Тестировать данный пример можно по следующему адресу:

https://host:8888/client2/examples/demo/streaming/video-chat-and-screen-sharing/video-chat-and-screen-sharing.html

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла `video-chat-and-screen-sharing.js` с хешем `90771d4`, которая находится здесь и доступна для скачивания в соответствующей сборке 2.0.218.

1. Инициализация API

`Flashphoner.init() code

```
try {
    Flashphoner.init();
} catch(e) {
    $("#notifyFlash").text("Your browser doesn't support WebRTC technology
needed for this example");
    return;
}
```

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 комнате.

`Session.join()` code

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

```
connection.join({name: getRoomName()}).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()}).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;
    }
    room_ = room;
    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. Публикация видеопотока с экрана

`Room.publish()` code

При публикации передаются параметры:

- параметры видео: ширина, высота, количество кадров в секунду, источник (экран)

- элемент страницы для отображения превью

```
var constraints = {
    video: {
        width: parseInt($('#width').val()),
        height: parseInt($('#height').val()),
        frameRate: parseInt($('#fps').val()),
        withoutExtension: true
    },
    audio: $("#useMic").prop('checked')
};
constraints.video.type = "screen";
if (Browser.isFirefox()){
    constraints.video.mediaSource = "screen";
}
var options = {
    name: "screenShare",
    display: document.getElementById("preview"),
    constraints: constraints,
    cacheLocalResources: false
}
if (isSafariMacOS()) {
    options.disableConstraintsNormalization = true;
}
room.publish(options).on(STREAM_STATUS.FAILED, function (stream) {
    ...
});
```

8. Получение от сервера события, сигнализирующего о присоединении пользователя к комнате

`RoomStatusEvent 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){
    ...
});
```

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

`RoomStatusEvent PUBLISHED` code

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

10. Воспроизведение видеопотока от другого участника

`Participant.play()` code

Параметром передается `div` элемент, в котором будет отображаться видео, в зависимости от источника - веб-камера или экран

```
function playParticipantsStream(participant) {
    if (participant.getStreams().length > 0) {
        for (var i=0; i<participant.getStreams().length; i++) {
            $("[id$=Name]").each(function (index, value) {
                if ($(value).text() == participant.name()) {
                    var p = value.id.replace('Name', '');
                    var pDisplay = p + 'Display';
                    // check if we already play this stream
                    if (document.getElementById(participant.getStreams()
[i].id()) == null) {
                        // setup 1st stream to main div
                        if (participant.getStreams()
[i].streamName().indexOf("screenShare") == -1) {
                            participant.getStreams()
[i].play(document.getElementById(pDisplay)).on(STREAM_STATUS.PLAYING,
function (playingStream) {

document.getElementById(playingStream.id()).addEventListener('resize',
function (event) {
                                    resizeVideo(event.target);
                                });
                            });
                        } else {
                            participant.getStreams()
[i].play(document.getElementById("sharedDisplay")).on(STREAM_STATUS.PLAYING,
function (playingStream) {

document.getElementById(playingStream.id()).addEventListener('resize',
function (event) {
                                    resizeVideo(event.target);
                                });
                            });
                        }
```

```
                    }
                }
            });
        }
    }
}
```

11. Остановка публикации видеопотока с экрана

`Stream.stop()` code

```
room.publish(options).on(STREAM_STATUS.FAILED, function (stream) {
    ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
    /*
     * User can stop sharing screen capture using Chrome "stop" button.
     * Catch onended video track event and stop publishing.
     */
    document.getElementById(stream.id()).srcObject.getVideoTracks()
[0].onended = function (e) {
        stream.stop();
    };
    ...
}).on(STREAM_STATUS.UNPUBLISHED, function(stream) {
    ...
});
```

12. Получение от сервера события, подтверждающего остановку публикации

`StreamStatusEvent UNPUBLISHED` code

```
room.publish(options).on(STREAM_STATUS.FAILED, function (stream) {
    ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
    ...
}).on(STREAM_STATUS.UNPUBLISHED, function(stream) {
    onStopSharing();
});
```

13. Выход из комнаты чата

`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);
```

```
    ...
}
```