Flash Video Chat

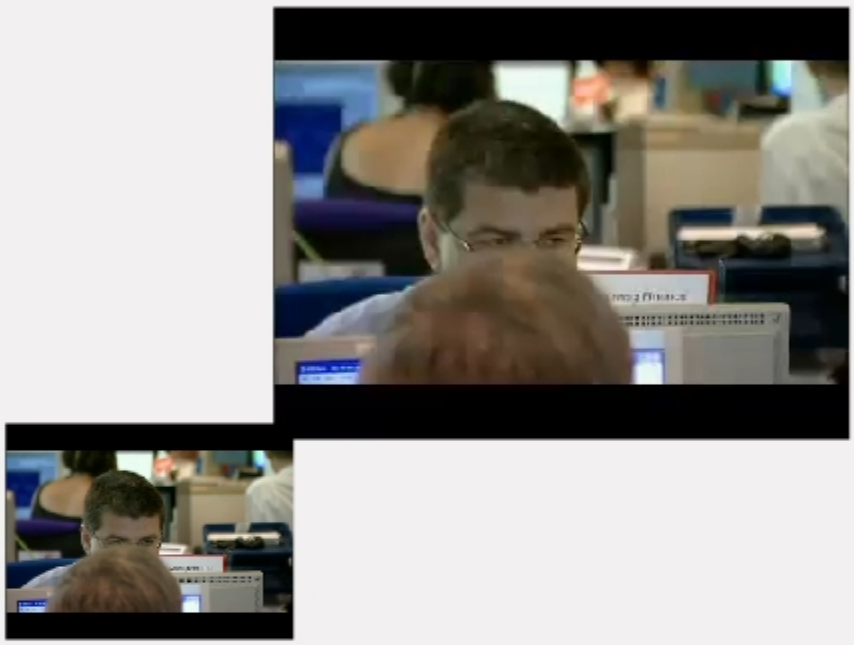
Пример двухстороннего видеочата в native Flash / Flex приложении

Данный пример представляет собой двухсторонний видеочат с использованием клиентского Flash приложения, которое может быть запущено простым swf-файлом.
Пример демонстрирует работу Flash видеочата, который позволяет установить двухстороннюю видеосвязь с таким же примером для Android или Web SDK.

На скриншоте показана работа Flash видеочата.

WCS URL rtmfp://192.168.1.59:1935

Login 22 Leave ESTABLISHED



222

Publish

UNPUBLISHED

11:56 chat - participants: 222

Send

Invite

http://192.168.1.59:9091/client2/examples/demo/streaming/flash_client/chat.html?roomName=53F89D

Интерфейс содержит поля для входа в видеочат:

- адрес WCS сервера
- имя пользователя (для тестирования можно использовать любое уникальное)

Под видео окнами находится простой текстовый чат для обмена сообщениями.

В поле 'Invite' выводится ссылка, по которой можно присоединить второго участника к этому чату.

Файлы примера

Пример представляет собой скомпилированный SWF-файл на HTML-странице, с использованием Flex / ActionScript3 и MXML и находится по следующему пути:

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/flash_client/chat.html

chat.html - страница примера
chat/bin-debug/chat.swf - файл приложения

Работа с исходным кодом примера

Для разбора кода возьмем версию файла chat.mxml с хешем 8b4bafe2766e0a1b485c41a8c64da80c74070ff1, который находится здесь. Результатом сборки streaming.mxml файла является приложение примера chat.swf. Скомпилированный swf и исходный код доступен для скачивания в соответствующей сборке 0.5.3.1894.

Основной файл примера chat.mxml опирается на несколько файлов, которые реализуют ROOM API, идентичное реализации модуля room-module.js для Web SDK.

com
 flashphoner
 room_api
 Participant.as
 RestAppCommunicator.as
 Room.as
 RoomStatus.as
 Session.as
 SessionStatus.as

Participant.as - объект, описывающий участника видеочата
RestAppCpmmunicator - объект, отвечающий за отправку sendData на WCS-сервер и получение входящих сообщений
Room.as - объект, описывающий "комнату", в которой находятся участники
RoomStatus.as - статусы комнат
Session.as - объект, описывающий соединение с сервером

1. В самом начале, при инициализации берется доступ к камере и микрофону.line 65

```
cam = Camera.getCamera();
localDisplay.attachCamera(cam);
mic = Microphone.getEnhancedMicrophone();
remoteDisplayHolder.addChild(remoteDisplay);
```

2. Далее создается объект Session с последующим коннектом к WCS-серверу. line 144

При успешном соединении с сервером будет вызван метод joinRoom() для присоединения к комнате.

```
session = new Session(url, username);
session.on(SessionStatus.FAILED, function():void{
        setStatus(sessionStatus, SessionStatus.FAILED);
        onLeft();
}).on(SessionStatus.DISCONNECTED, function():void {
        setStatus(sessionStatus, SessionStatus.DISCONNECTED);
        onLeft();
}).on(SessionStatus.ESTABLISHED, function():void {
        setStatus(sessionStatus, SessionStatus.ESTABLISHED);
        joinRoom();
});
session.connect();
```

3. Во время присоединения к комнате будут добавлены реакции на различные события, происходящие внутри этой комнаты. line 150

JOINED - к комнате присоединился новый участник
LEFT - участник покинул комнату
PUBLISHED - участник опубликовал видеопоток
FAILED - ошибка в коммуникации с комнатой
MESSAGE - входящее сообщение от участника внутри комнаты

```
session.join(this.roomName).on(RoomStatus.STATE, function(room:Room):void{
        var participants:Array = room.getParticipants();
        setInviteAddress(room);
        if (participants.length > 0) {
                var chatState:String = "participants: ";
                for (var i:Number = 0; i < participants.length; i++) {
                        installParticipant(participants[i]);
                        chatState += participants[i].getName();
                        if (i != participants.length - 1) {
                                chatState += ",";
                        }
                }
                addMessage("chat", chatState);
        } else {
                addMessage("chat", " room is empty");
        }
        publishLocalMedia(room);
        onJoined(room);
}).on(RoomStatus.JOINED, function(participant:Participant):void{
        installParticipant(participant);
        addMessage(participant.getName(), "joined");
}).on(RoomStatus.LEFT, function(participant:Participant):void{
        removeParticipant();
        addMessage(participant.getName(), "left");
}).on(RoomStatus.PUBLISHED, function(participant:Participant):void{
        playParticipantsStream(participant);
}).on(RoomStatus.FAILED, function(room:Room, info:Object):void{
        failedInfo.text = info.info;
        session.disconnect();
}).on(RoomStatus.MESSAGE, function(message:Object):void{
        addMessage(message.from.getName(), message.text);
});
```

4. Публиковать видеопоток с веб-камеры на WCS-сервер. line 232

```
private function publishLocalMedia(room:Room):void {
    var stream:NetStream = room.publish(mic, cam);
    stream.addEventListener(NetStatusEvent.NET_STATUS, function(event:NetStatusEvent):void{
        Logger.info("handlePublishStreamStatus: "+event.info.code);
        switch (event.info.code) {
            case "NetStream.Publish.BadName":
                setStatus(streamStatus, "FAILED");
                onMediaStopped(room);
            break;
            case "NetStream.Unpublish.Success":
                setStatus(streamStatus, "UNPUBLISHED");
                onMediaStopped(room);
            break;
            case "NetStream.Publish.Start":
                setStatus(streamStatus, "PUBLISHING");
                onMediaPublished(stream);
            break;
        }
    });
}
```

5. Воспроизвести поток другого участника. line 207

```
private function playParticipantsStream(p:Participant):void
{
    var stream:NetStream = p.play();
    if (stream != null) {
        remoteDisplay.attachNetStream(stream);
        stream.addEventListener(NetStatusEvent.NET_STATUS, function(event:NetStatusEvent):void{
            Logger.info("handlePlayStreamStatus: "+event.info.code);
            switch (event.info.code) {
                case "NetStream.Video.DimensionChange":
                    var res:Object = downScaleToFitSize(remoteDisplay.videoWidth, remoteDisplay.videoHeight,
display.width, display.height);
                    remoteDisplay.width = res.w;
                    remoteDisplay.height = res.h;
                    remoteDisplayHolder.width = res.w;
                    remoteDisplayHolder.height = res.h;
                break;
                case "NetStream.Play.UnpublishNotify":
                case "NetStream.Play.Stop":
                    remoteDisplay.clear();
                break;
            }
        });
    }
}
```