


Firewall Streaming

Пример использования TURN-сервера

Пример публикации и воспроизведения потока через firewall с использованием [TURN-сервера](#). Данный способ работает во всех браузерах, кроме Microsoft Legacy Edge, поскольку последний не поддерживает TURN по TCP.


Firewall Traversal Streaming

Local



testStop

Player



testStop

PUBLISHING

WCS Server

turn:turn.flashphoner.com:4

Turn Server

turn:turn.flashphoner.com:4

Username of turn server

flashphoner

Credential of turn server

coM77EMrV7Cwhyan

Force relay

☒

Disconnect

ESTABLISHED

PLAYING

Код примера

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

`/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/firewall-traversal-streaming/`

- firewall-traversal-streaming.css - файл стилей
- firewall-traversal-streaming.html - страница примера
- firewall-traversal-streaming.js - скрипт, обеспечивающий работу примера

Тестировать данный пример можно по следующему адресу:

<https://host:8888/client2/examples/demo/streaming/firewall-traversal-streaming/firewall-traversal-streaming.html>

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла firewall-traversal-streaming.js с хешем ecbadc3, которая находится [здесь](#) и доступна для скачивания в соответствующей сборке [2.0.212](#).

1. Инициализация API

`Flashphoner.init()` [code](#)

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

2. Подключение к серверу

`Flashphoner.createSession()` [code](#)

Методу `createSession()` передаются параметры:

- URL WCS-сервера
- URL TURN-сервера
- параметры авторизации пользователя на TURN-сервере

```
var options = {
  urlServer: url,
  mediaOptions: {
    "iceServers": [
      {
        'url': $('#urlTurnServer').val(),
        'username': $('#usernameTurnServer').val(),
        'credential': $('#credentialTurnServer').val()
      }
    ]
  }
};
if ($("#forceRelay").is(':checked')) {
  options.mediaOptions.iceTransportPolicy = "relay";
}
```

```

}
Flashphoner.createSession(options).on(SESSION_STATUS.ESTABLISHED, function
(session) {
    ...
});

```

3. Получение от сервера события, подтверждающего успешное соединение

`ConnectionStatusEvent ESTABLISHED` [code](#)

```

Flashphoner.createSession(options).on(SESSION_STATUS.ESTABLISHED, function
(session) {
    setStatus("#connectStatus", session.status());
    onConnected(session);
}).on(SESSION_STATUS.DISCONNECTED, function () {
    ...
}).on(SESSION_STATUS.FAILED, function () {
    ...
});

```

4. Публикация видеопотока

`Session.createStream()`, `Stream.publish()` [code](#)

Методы `createStream()` передаются:

- `streamName` - имя видеопотока
- `localVideo` - div-элемент, в котором будет отображаться видео с камеры.

```

session.createStream({
    name: streamName,
    display: localVideo,
    cacheLocalResources: true,
    receiveVideo: false,
    receiveAudio: false
    ...
}).publish();

```

5. Получение от сервера события, подтверждающего успешную публикацию потока

`StreamStatusEvent PUBLISHING` [code](#)

```

session.createStream({
    ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
    setStatus("#publishStatus", STREAM_STATUS.PUBLISHING);
    onPublishing(stream);
});

```

```

}).on(STREAM_STATUS.UNPUBLISHED, function () {
    ...
}).on(STREAM_STATUS.FAILED, function () {
    ...
}).publish();

```

6. Воспроизведение видеопотока

`Session.createStream()`, `Stream.play()` [code](#)

Методы `createStream()` передаются:

- `streamName` - имя видеопотока (в том числе, это может быть имя потока, опубликованного выше)
- `remoteVideo` - div-элемент, в котором будет отображаться видео.

```

session.createStream({
    name: streamName,
    display: remoteVideo
    ...
}).play();

```

7. Получение от сервера события, подтверждающего успешное воспроизведение потока

`StreamStatusEvent PLAYING` [code](#)

```

session.createStream({
    ...
}).on(STREAM_STATUS.PLAYING, function (stream) {
    document.getElementById(stream.id()).addEventListener('resize', function
(event) {
        resizeVideo(event.target);
    });
    setStatus("#playStatus", stream.status());
    onPlaying(stream);
}).on(STREAM_STATUS.STOPPED, function () {
    ...
}).on(STREAM_STATUS.FAILED, function () {
    ...
}).play();

```

8. Остановка воспроизведения видеопотока

`Stream.stop()` [code](#)

```

function onPlaying(stream) {
    $("#playBtn").text("Stop").off('click').click(function () {
        $(this).prop('disabled', true);
    });
}

```

```

        stream.stop();
    }).prop('disabled', false);
}

```

9. Получение от сервера события, подтверждающего успешную остановку воспроизведения потока

StreamStatusEvent STOPPED [code](#)

```

session.createStream({
    name: streamName,
    display: remoteVideo
}).on(STREAM_STATUS.PLAYING, function (stream) {
    ...
}).on(STREAM_STATUS.STOPPED, function () {
    setStatus("#playStatus", STREAM_STATUS.STOPPED);
    onStoped();
}).on(STREAM_STATUS.FAILED, function () {
    ...
}).play();

```

10. Остановка публикации видеопотока

Stream.stop() [code](#)

```

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

```

11. Получение от сервера события, подтверждающего успешную остановку публикации потока

StreamStatusEvent UNPUBLISHED [code](#)

```

session.createStream({
    ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
    ...
}).on(STREAM_STATUS.UNPUBLISHED, function () {
    setStatus("#publishStatus", STREAM_STATUS.UNPUBLISHED);
    onUnpublished();
}).on(STREAM_STATUS.FAILED, function () {
    ...
}).publish();

```