

# Embed Player

- [Пример плеера, встраиваемого на веб-страницу](#)
- [Код примера](#)
- [Работа с кодом примера](#)

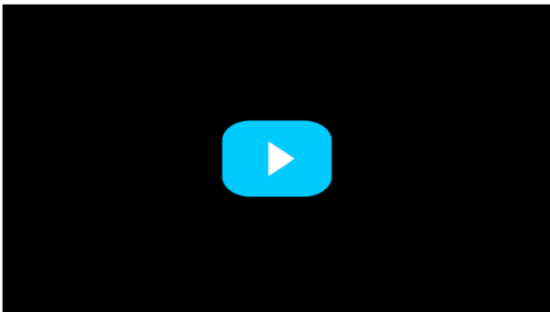
## Пример плеера, встраиваемого на веб-страницу

Данный пример может использоваться для внедрения на веб-страницу плеера для воспроизведения живых (Live) потоков с веб-камер и IP камер. Поддерживаются следующие технологии воспроизведения:

- WebRTC
- MSE

Интерфейс страницы внедрения:

### Embed player



Server

Stream

Tech

☒ WebRTC

☒ MSE

Test now

Code

```
<iframe id='fp_embed_player' src='http://localhost:9091/embed_player?urlServer=ws://localhost:8080&streamName=test&mediaProviders=WebRTC,MSE' marginwidth='0' marginheight='0' frameborder='0' width='100%' height='100%' scrolling='no' allowfullscreen='allowfullscreen'></iframe>
```

Copy to clipboard

## Код примера

Код данного примера находится на сервере по следующему пути:

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/embed\_player

player.css - файл стилей  
player.html - страница с плеером  
player.js - скрипт, обеспечивающий работу плеера  
sample.css - файл стилей для страницы интерфейса внедрения  
sample.html - страница интерфейса внедрения  
sample.js - скрипт, обеспечивающий формирование кода внедрения

Тестировать данный пример можно по следующему адресу:

https://host:8888/client2/examples/demo/streaming/embed\_player/sample.html

Здесь host - адрес вашего WCS-сервера.

## Работа с кодом примера

Для разбора кода возьмем версию файла `player.js` с хешем `24a69e1`, которая находится [здесь](#) и доступна для скачивания в соответствующей сборке [2.0.225](#).

#### 1. Инициализация API.

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

```
Flashphoner.init({ preferredMediaProviders: mediaProviders && mediaProviders !== "" ? mediaProviders.split(
','): [] });
```

#### 2. Подключение к серверу.

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

Методу `createSession()` передаются параметры:

- `urlServer` - URL WCS-сервера
- `mediaOptions` - параметры подключения к серверу через [TURN-сервер](#)

```
let mediaOptions = {"iceServers": [{ 'url': 'turn:turn.flashphoner.com:443?transport=tcp', 'username':
'flashphoner', 'credential': 'com77EMrV7Cwhyant' }]};
Flashphoner.createSession({urlServer: urlServer, mediaOptions: mediaOptions}).on(SESSION_STATUS.ESTABLISHED,
function (session) {
    ...
});
```

#### 3. Получение от сервера события, подтверждающего успешное соединение.

`SESSION_STATUS.ESTABLISHED` [code](#)

```
Flashphoner.createSession({urlServer: urlServer, mediaOptions: mediaOptions}).on(SESSION_STATUS.ESTABLISHED,
function (session) {
    setStatus(session.status());
    //session connected, start playback
    playStream(session);
}).on(SESSION_STATUS.DISCONNECTED, function () {
    ...
}).on(SESSION_STATUS.FAILED, function () {
    ...
});
```

#### 4. Воспроизведение видеопотока.

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

В метод `createStream` передаются:

- имя видеопотока `streamName`
- `remoteVideo` - div элемент, в котором будет отображаться видео
- разрешение, с которым поток будет проигрываться (на сервере включается транскодинг)
- `useControls` - включает использование стандартных элементов управления проигрыванием потока
- `unmutePlayOnStart: false` - отключает автоматическое воспроизведение звука при автозапуске воспроизведения для соблюдения требований браузеров

```

let useVideoControls = true;
...
let options = {
  name: streamName,
  display: remoteVideo,
  useControls: useVideoControls
};
if (resolution) {
  playWidth = resolution.split("x")[0];
  playHeight = resolution.split("x")[1];
  options.constraints = {
    video: {
      width: playWidth,
      height: playHeight
    },
    audio: true
  };
}
if (autoplay) {
  options.unmutePlayOnStart = false;
}
playingStream = session.createStream(options).on(STREAM_STATUS.PENDING, function (stream) {
  ...
});
playingStream.play();

```

5. Получение от сервера события, подтверждающего готовность потока к воспроизведению

STREAM\_STATUS.PENDING [code](#)

По данному событию:

- в браузере Chrome отключается собственный прелоадер, поскольку есть стандартный при включенных стандартных элементах управления
- настраивается обработчик изменения размера видео элемента по событию `resize`
- настраиваются обработчики событий в браузере Safari и в остальных браузерах

```

playingStream = session.createStream(options).on(STREAM_STATUS.PENDING, function (stream) {
  if (Browser.isChrome()) {
    // Hide a custom preloader in Chrome because there is a standard one with standard controls
    hideItem('preloader');
  }
  let video = document.getElementById(stream.id());
  if (!video.hasListeners) {
    video.hasListeners = true;
    setResizeHandler(video, stream, playWidth);
    if (Browser.isSafariWebRTC()) {
      setWebkitEventHandlers(video);
    } else {
      setEventHandlers(video);
    }
  }
}).on(STREAM_STATUS.PLAYING, function (stream) {
  ...
}).on(STREAM_STATUS.STOPPED, function () {
  ...
}).on(STREAM_STATUS.FAILED, function (stream) {
  ...
}).on(STREAM_EVENT, function (streamEvent) {
  ...
});
playingStream.play();

```

6. Получение от сервера события, подтверждающего успешное воспроизведение потока

STREAM\_STATUS.PLAYING [code](#)

По этому событию снимается с паузы проигрывание потока по MSE в браузере Android Firefox

```

playingStream = session.createStream(options).on(STREAM_STATUS.PENDING, function (stream) {
    ...
}).on(STREAM_STATUS.PLAYING, function (stream) {
    // Android Firefox may pause stream playback via MSE even if video element is muted
    if (Flashphoner.getMediaProviders()[0] == "MSE" && autoplay && Browser.isAndroidFirefox()) {
        let video = document.getElementById(stream.id());
        if (video && video.paused) {
            video.play();
        }
    }
    setStatus(STREAM_STATUS.PLAYING);
    onStart();
}).on(STREAM_STATUS.STOPPED, function () {
    ...
}).on(STREAM_STATUS.FAILED, function(stream) {
    ...
}).on(STREAM_EVENT, function(streamEvent){
    ...
});
playingStream.play();

```

#### 7. Остановка воспроизведения видеопотока.

Stream.stop() [code](#)

```

playingStream.stop();

```

#### 8. Получение от сервера события, подтверждающего успешную остановку воспроизведения потока

STREAM\_STATUS.STOPPED [code](#)

```

playingStream = session.createStream(options).on(STREAM_STATUS.PENDING, function (stream) {
    ...
}).on(STREAM_STATUS.PLAYING, function (stream) {
    ...
}).on(STREAM_STATUS.STOPPED, function () {
    setStatus(STREAM_STATUS.STOPPED);
    onStop();
}).on(STREAM_STATUS.FAILED, function(stream) {
    ...
}).on(STREAM_EVENT, function(streamEvent){
    ...
});
playingStream.play();

```

#### 9. Автозапуск воспроизведения при загрузке страницы

[code](#)

```

if (autoplay) {
    centralButton.click();
}

```

#### 10. Настройка обработчика события resize

[code](#)

По данному событию меняется размер контейнера для video элемента

```
function setResizeHandler(video, stream, playWidth) {
  video.addEventListener('resize', function (event) {
    let streamResolution = stream.videoResolution();
    if (Object.keys(streamResolution).length === 0) {
      resizeVideo(event.target);
    } else {
      // Change aspect ratio to prevent video stretching
      let ratio = streamResolution.width / streamResolution.height;
      let newHeight = Math.floor(playWidth / ratio);
      resizeVideo(event.target, playWidth, newHeight);
    }
  });
}
```

## 11. Настройка обработчиков событий видео в браузере Safari

[code](#)

Обрабатываются следующие события:

- `playing` - скрывается собственный прелоадер при проигрывании потока
- `webkitbeginfullscreen`, `webkitendfullscreen` - определяется полноэкранный режим для снятия потока с паузы при выходе из него в iOS Safari
- `pause` - проигрывание снимается с паузы при выходе из полноэкранного режима; при нажатии на стандартную кнопку паузы в оконном режиме останавливается проигрывание потока

```
function setWebkitEventHandlers(video) {
  let needRestart = false;
  let isFullscreen = false;
  // Hide custom preloader
  video.addEventListener('playing', function () {
    hideItem('preloader');
  });
  // Use webkitbeginfullscreen event to detect full screen mode in iOS Safari
  video.addEventListener("webkitbeginfullscreen", function () {
    isFullscreen = true;
  });
  video.addEventListener("pause", function () {
    if (needRestart) {
      console.log("Video paused after fullscreen, continue...");
      video.play();
      needRestart = false;
    } else if (!(isFullscreen || document.webkitFullscreenElement)) {
      // Stop stream by standard play/pause control
      playingStream.stop();
    }
  });
  video.addEventListener("webkitendfullscreen", function () {
    video.play();
    needRestart = true;
    isFullscreen = false;
  });
}
```

## 12. Настройка обработчиков событий видео в других браузерах

[code](#)

Обрабатываются следующие события:

- `playing` - скрывается собственный прелоадер при проигрывании потока
- `pause` - при нажатии на стандартную кнопку паузы в оконном режиме останавливается проигрывание потока

```
function setEventHandlers(video) {  
  // Hide custom preloader  
  video.addEventListener('playing', function () {  
    hideItem('preloader');  
  });  
  // Use standard pause control to stop playback  
  video.addEventListener("pause", function () {  
    if (!(document.fullscreenElement || document.mozFullscreenElement)) {  
      // Stop stream by standard play/pause control if we're not in fullscreen  
      playingStream.stop();  
    }  
  });  
}
```