

С экрана компьютера (screen sharing) в браузере по WebRTC

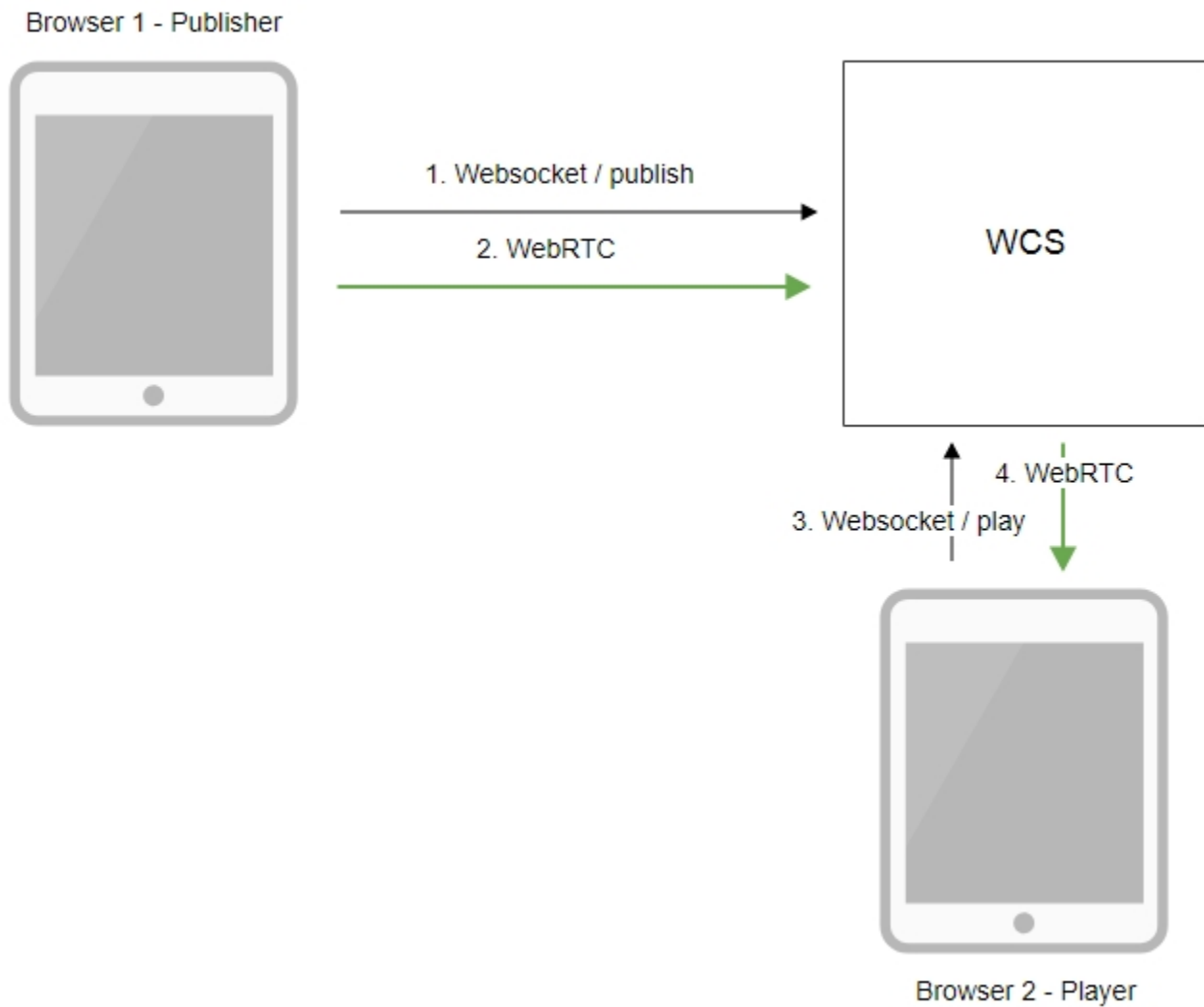
- [Описание](#)
 - [Поддерживаемые платформы и браузеры](#)
 - [Схема работы](#)
- [Краткое руководство по тестированию](#)
 - [Захват видеопотока с экрана компьютера и подготовка к его трансляции](#)
- [Последовательность выполнения операций \(Call Flow\)](#)
- [Разработчику](#)
 - [Расширение для Google Chrome с публикацией в Chrome Store](#)
 - [Создайте аккаунт Google](#)
 - [Зарегистрируйтесь как разработчик Chrome Web Store](#)
 - [Опубликуйте расширение в Chrome Web Store](#)
 - [Встроенная установка расширения](#)
 - [Настройка клиента](#)
 - [Параметры источника медиа](#)
 - [Список параметров](#)
 - [Захват системного звука в браузере Chrome](#)
 - [Управление источником захвата \(экран или окно\) в браузере Firefox](#)
 - [Демонстрация экрана без использования расширения](#)
 - [Браузер Firefox](#)
 - [Браузеры на основе Chrome](#)
 - [Браузер Safari для MacOS](#)
 - [Известные ограничения](#)
- [Известные проблемы](#)

Описание

Поддерживаемые платформы и браузеры

	Chrome	Firefox	Safari
Windows	+	+	
Mac OS	+	+	+

Схема работы



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду publish.
2. Браузер захватывает экран и отправляет WebRTC поток на сервер.
3. Второй браузер устанавливает соединение также по Websocket и отправляет команду play.
4. Второй браузер получает WebRTC поток и воспроизводит этот поток на странице.

Краткое руководство по тестированию

Захват видеопотока с экрана компьютера и подготовка к его трансляции

1. Для теста используем демо-сервер demo.flashphoner.com и веб-приложение Screen Sharing в браузере Chrome

<https://demo.flashphoner.com/client2/examples/demo/streaming/screen-sharing/screen-sharing.html>

Screen Sharing

Chrome extension
Install Now

FPS

Width

Height

W/o extension?
☒

Use Mic?
☐

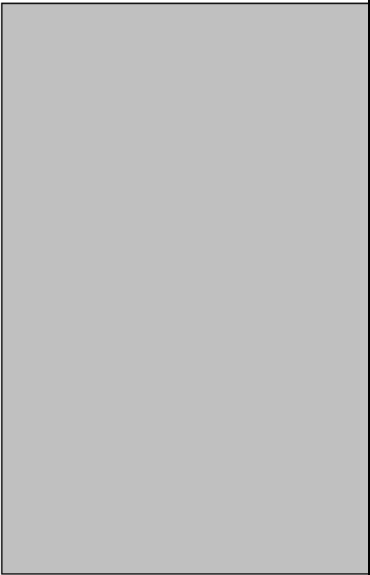
Mic

Default - Microphone ▼



My Screen

➔



Preview

wss://demo.flashphoner.com:8443/3a437ca9

Start

2. Нажмите кнопку "Start". Браузер запросит доступ к экрану, и начнется захват экрана и трансляция видеопотока:

Screen Sharing

Chrome extension
Install Now

FPS

Width

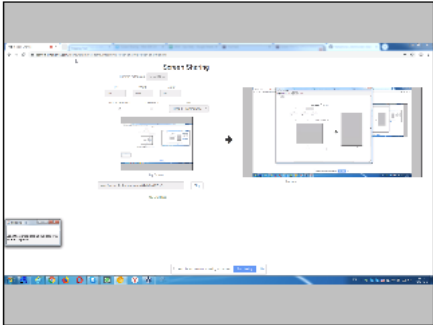
Height

W/o extension?
☒

Use Mic?
☐

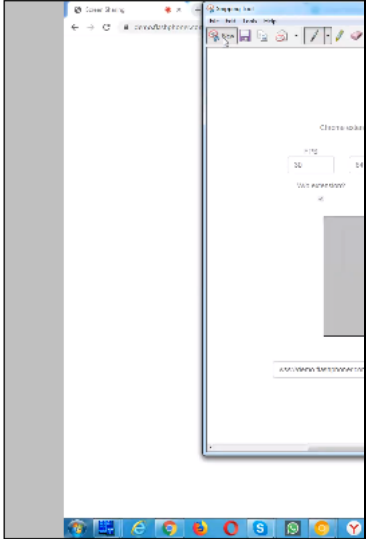
Mic

Default - Microphone ▼



My Screen

➔



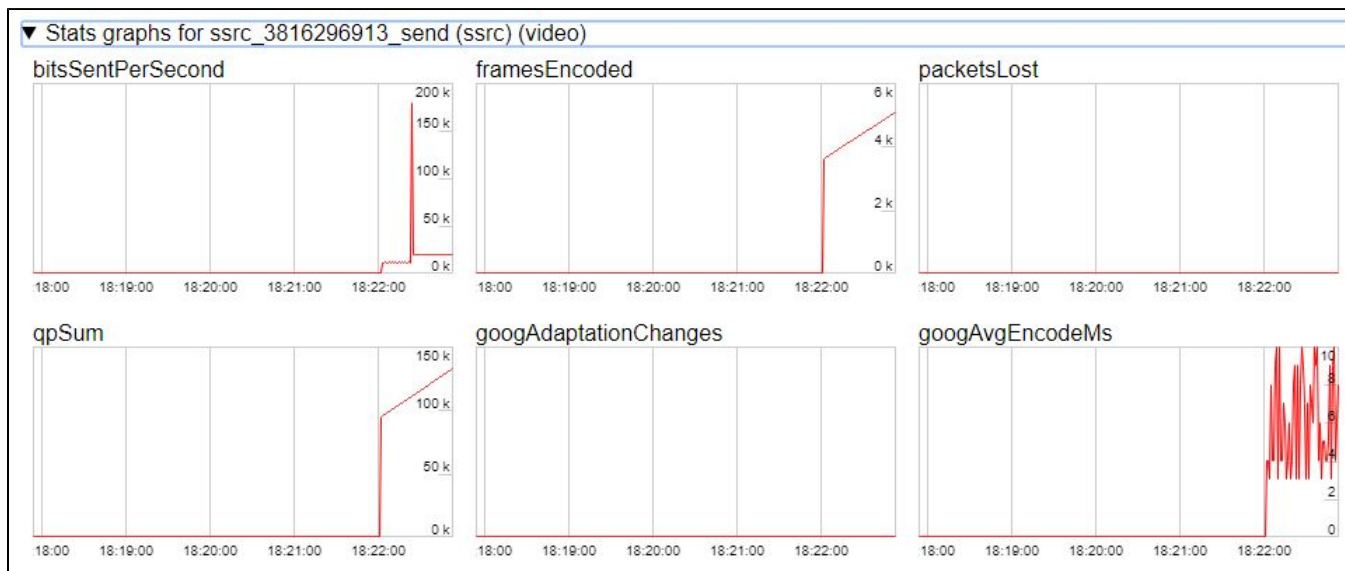
Preview

wss://demo.flashphoner.com:8443/3a437ca9

Stop

PUBLISHING

3. Убедитесь, что поток отправляется на сервер и система работает нормально, откройте <chrome://webrtc-internals>



4. Откройте [Two Way Streaming](#) в отдельном окне, нажмите Connect и укажите идентификатор потока, затем нажмите Play.

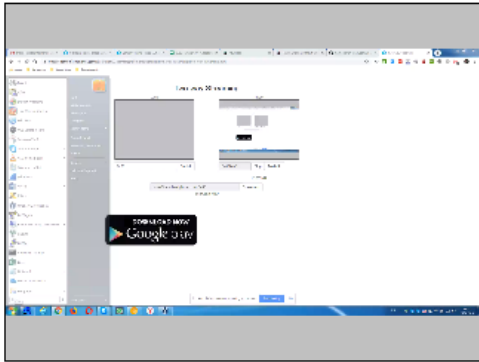
Two-way Streaming

Local



5c6f Publish

Player



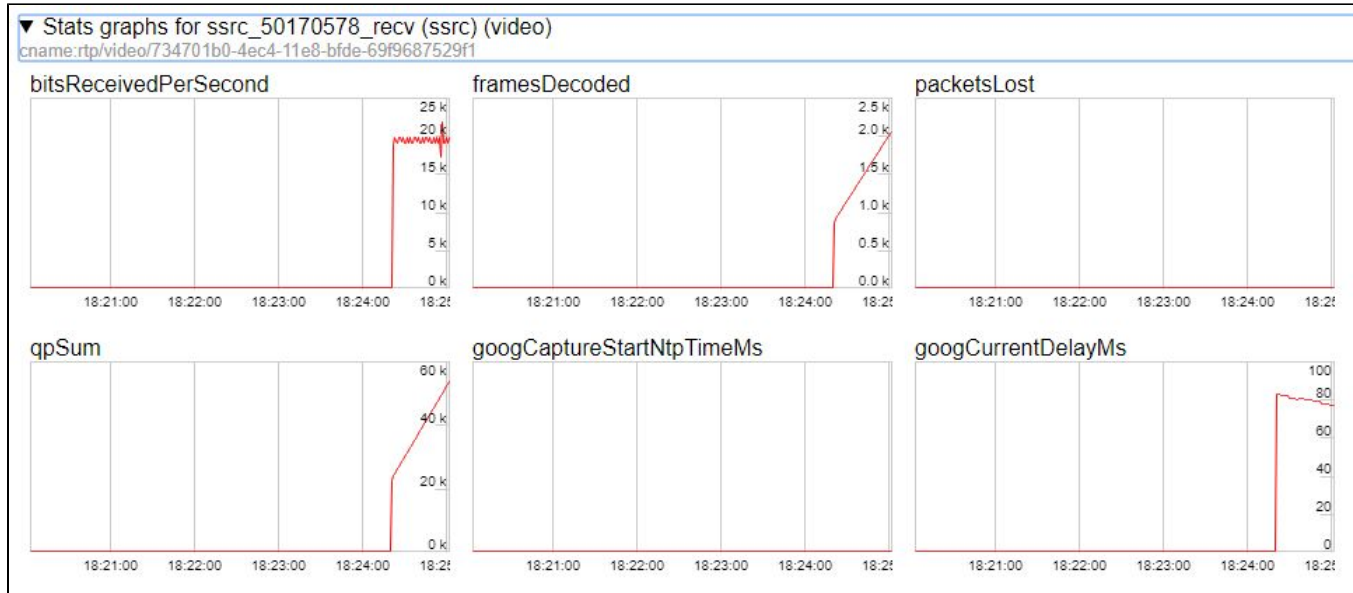
3a437ca9 Stop Available

PLAYING

wss://demo.flashphoner.com:8443 Disconnect

ESTABLISHED

5. Графики воспроизведения <chrome://webrtc-internals>

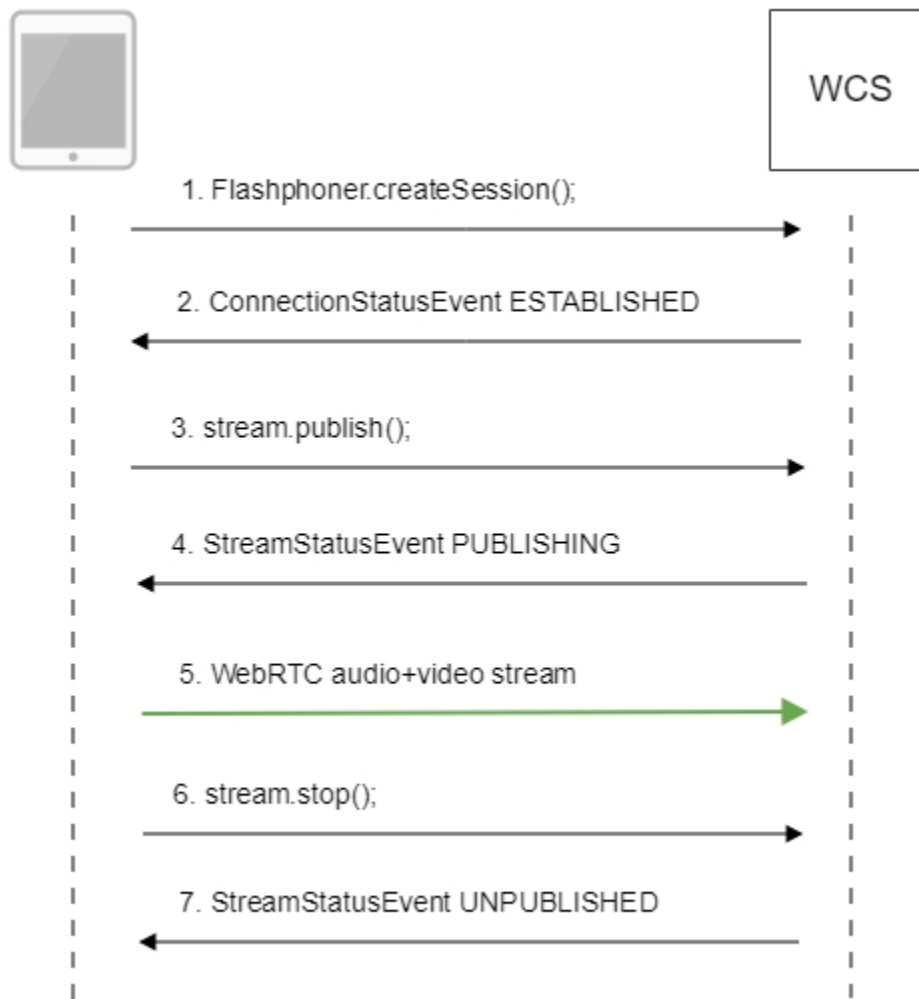


Последовательность выполнения операций (Call Flow)

Ниже описана последовательность вызовов при использовании примера Screen Sharing

[screen-sharing.html](#)

[screen-sharing.js](#)



1. Проверка необходимости установки расширения

Browser.isFirefox(), Browser.isChrome();[code](#)

```

if (Browser.isFirefox()) {
    $("#installExtensionButton").show();
    ...
} else if (Browser.isChrome()) {
    $('#mediaSourceForm').hide();
    interval = setInterval(function() {
        chrome.runtime.sendMessage(extensionId, {type: "isInstalled"}, function (response) {
            if (response) {
                $("#extension").hide();
                clearInterval(interval);
                onExtensionAvailable();
            } else {
                (inIframe()) ? $("#installFromMarket").show() : $("#installExtensionButton").show();
            }
        });
    }, 500);
} else {
    $("#notify").modal('show');
    return false;
}

```

2. Установка соединения с сервером.

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

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    //session connected, start streaming
    startStreaming(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
    setStatus(SESSION_STATUS.DISCONNECTED);
    onStopped();
}).on(SESSION_STATUS.FAILED, function(){
    setStatus(SESSION_STATUS.FAILED);
    onStopped();
});
```

3. Получение от сервера события, подтверждающего успешное соединение.

ConnectionStatusEvent ESTABLISHED[code](#)

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    //session connected, start streaming
    startStreaming(session);
    ...
});
```

4. Публикация потока.

stream.publish();[code](#)

```
session.createStream({
    name: streamName,
    display: localVideo,
    constraints: constraints
}).on(STREAM_STATUS.PUBLISHING, function(publishStream){
    ...
}).on(STREAM_STATUS.UNPUBLISHED, function(){
    ...
}).on(STREAM_STATUS.FAILED, function(stream){
    ...
}).publish();
```

5. Получение от сервера события, подтверждающего успешную публикацию потока.

StreamStatusEvent, статус PUBLISHING[code](#)

```

session.createStream({
  name: streamName,
  display: localVideo,
  constraints: constraints
}).on(STREAM_STATUS.PUBLISHING, function(publishStream){
  ...
  setStatus(STREAM_STATUS.PUBLISHING);
  //play preview
  session.createStream({
    name: streamName,
    display: remoteVideo
  }).on(STREAM_STATUS.PLAYING, function(previewStream){
    document.getElementById(previewStream.id()).addEventListener('resize', function(event){
      resizeVideo(event.target);
    });
    //enable stop button
    onStart(publishStream, previewStream);
  }).on(STREAM_STATUS.STOPPED, function(){
    publishStream.stop();
  }).on(STREAM_STATUS.FAILED, function(stream){
    //preview failed, stop publishStream
    if (publishStream.status() == STREAM_STATUS.PUBLISHING) {
      setStatus(STREAM_STATUS.FAILED, stream);
      publishStream.stop();
    }
  }).play();
}).on(STREAM_STATUS.UNPUBLISHED, function(){
  ...
}).on(STREAM_STATUS.FAILED, function(stream){
  ...
}).publish();

```

6. Отправка аудио-видео потока по WebRTC

7. Остановка публикации потока.

stream.stop();[code](#)


```

session.createStream({
  name: streamName,
  display: localVideo,
  constraints: constraints
}).on(STREAM_STATUS.PUBLISHING, function(publishStream){
  /*
   * User can stop sharing screen capture using Chrome "stop" button.
   * Catch onended video track event and stop publishing.
   */
  document.getElementById(publishStream.id()).srcObject.getVideoTracks()[0].onended = function (e) {
    publishStream.stop();
  };
  ...
  setStatus(STREAM_STATUS.PUBLISHING);
  //play preview
  session.createStream({
    name: streamName,
    display: remoteVideo
  }).on(STREAM_STATUS.PLAYING, function(previewStream){
    ...
  }).on(STREAM_STATUS.STOPPED, function(){
    publishStream.stop();
  }).on(STREAM_STATUS.FAILED, function(stream){
    //preview failed, stop publishStream
    if (publishStream.status() == STREAM_STATUS.PUBLISHING) {
      setStatus(STREAM_STATUS.FAILED, stream);
      publishStream.stop();
    }
  }).play();
  ...
}).publish();

```

8. Получение от сервера события, подтверждающего остановку публикации потока.

StreamStatusEvent, статус UNPUBLISHED [code](#)

```

session.createStream({
  name: streamName,
  display: localVideo,
  constraints: constraints
}).on(STREAM_STATUS.PUBLISHING, function(publishStream){
  ...
}).on(STREAM_STATUS.UNPUBLISHED, function(){
  setStatus(STREAM_STATUS.UNPUBLISHED);
  //enable start button
  onStopped();
}).on(STREAM_STATUS.FAILED, function(stream){
  ...
}).publish();

```

Разработчику

Функция демонстрации экрана может быть использована для публикации видеопотока, демонстрирующего рабочий стол или окно приложения. Для демонстрации экрана WCS API использует расширение для браузера Chrome. Браузер Firefox, начиная с версии 52, не нуждается в расширении.

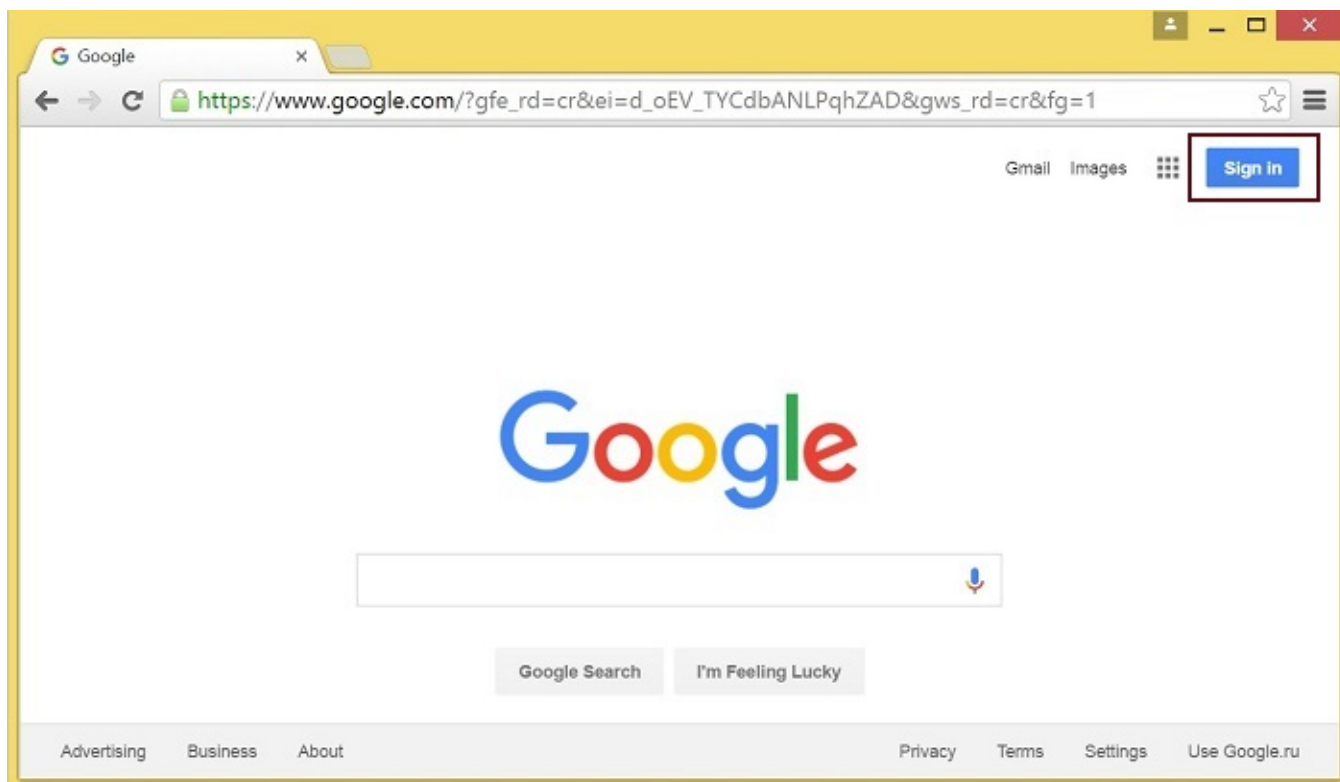
Расширение для Google Chrome с публикацией в Chrome Store

Исходный код расширения для сборки доступен по ссылке ниже:

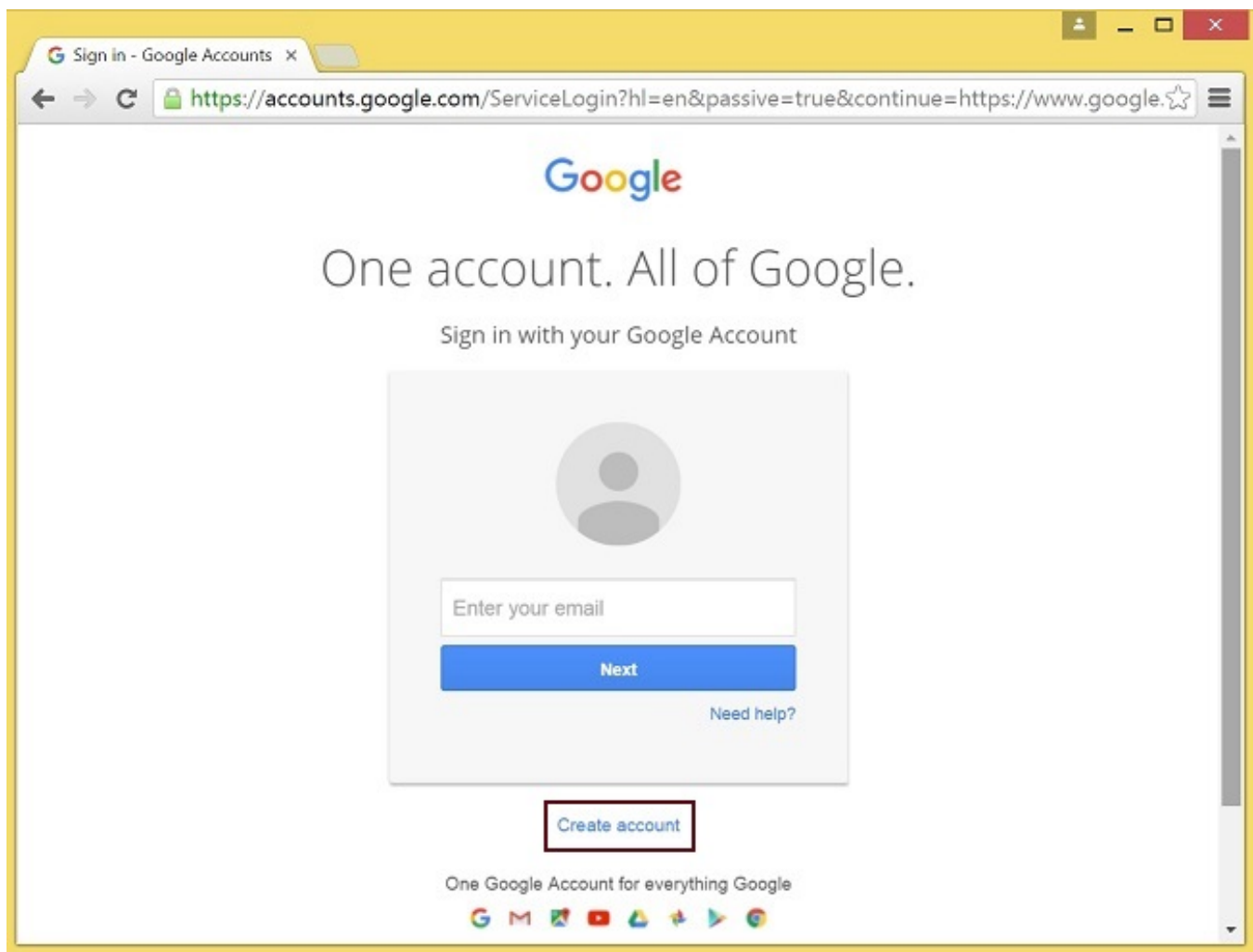
[Chrome Screen Sharing Extension](#)

Создайте аккаунт Google

1. Зайдите на [google.com](https://www.google.com) и кликните кнопку 'Sign in'



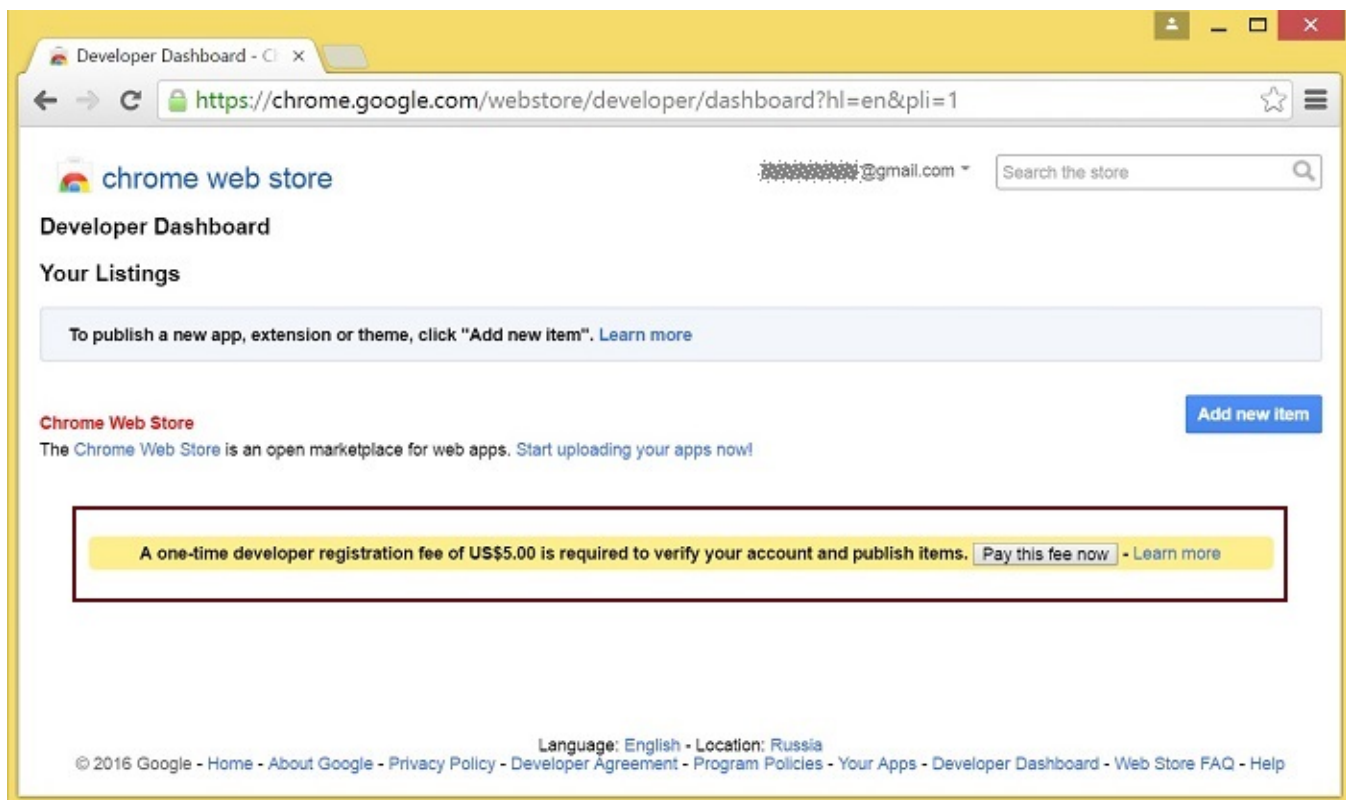
2. Кликните ссылку 'Create account' на странице 'Sign in'



3. Будет открыта страница 'Create your Google Account'
Заполните необходимые поля и кликните кнопку 'Next step', чтобы создать аккаунт.

Зарегистрируйтесь как разработчик Chrome Web Store

1. Войдите в [Chrome Developer Dashboard](#), используя созданный аккаунт Google
2. Внесите единовременную плату в размере US\$5, чтобы подтвердить аккаунт



Адаптируйте расширение к вашему домену

Выполните действия, описанные ниже, чтобы использовать расширения со своим доменом. Отредактируйте файл манифеста `manifest.json` расширения для Chrome.

Измените:

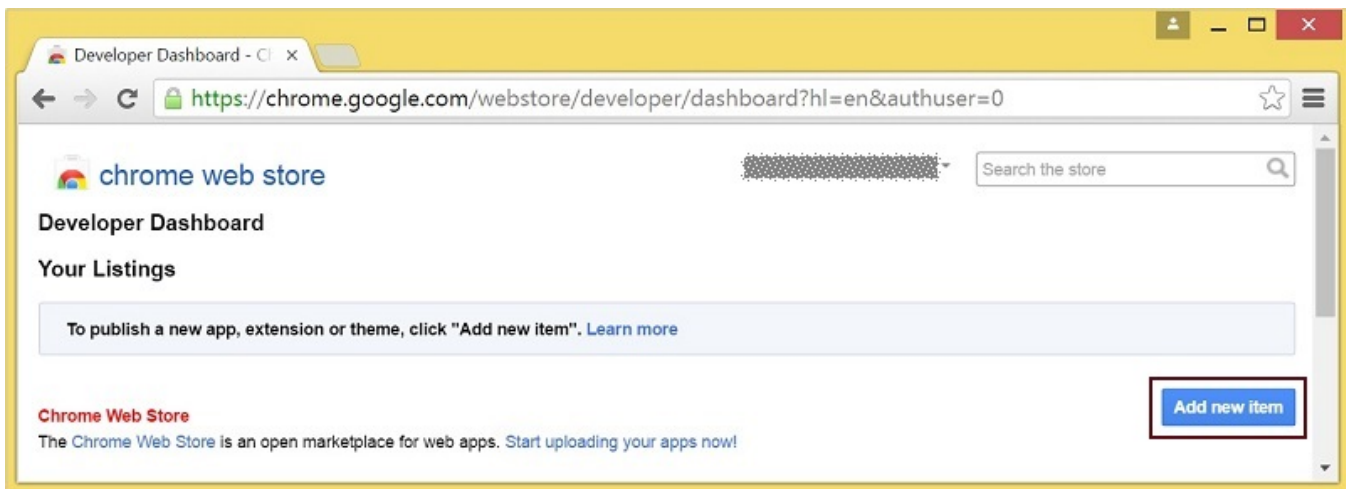
- имя (name)
- автора (author)
- описание (description)
- адрес домашней страницы (homepage_url)
- в "externally_connectable": "matches" замените flashphoner.com на свой домен

Добавьте свои иконки в директорию `chrome-extension` и отредактируйте имена файлов в "icons" и "web_accessible_resources". (Для дополнительной информации см. [Manifest - Icons](#) и [Supplying Images](#).)

Опубликуйте расширение в Chrome Web Store

Упакуйте файлы из директории `chrome-extension` в zip-архив.

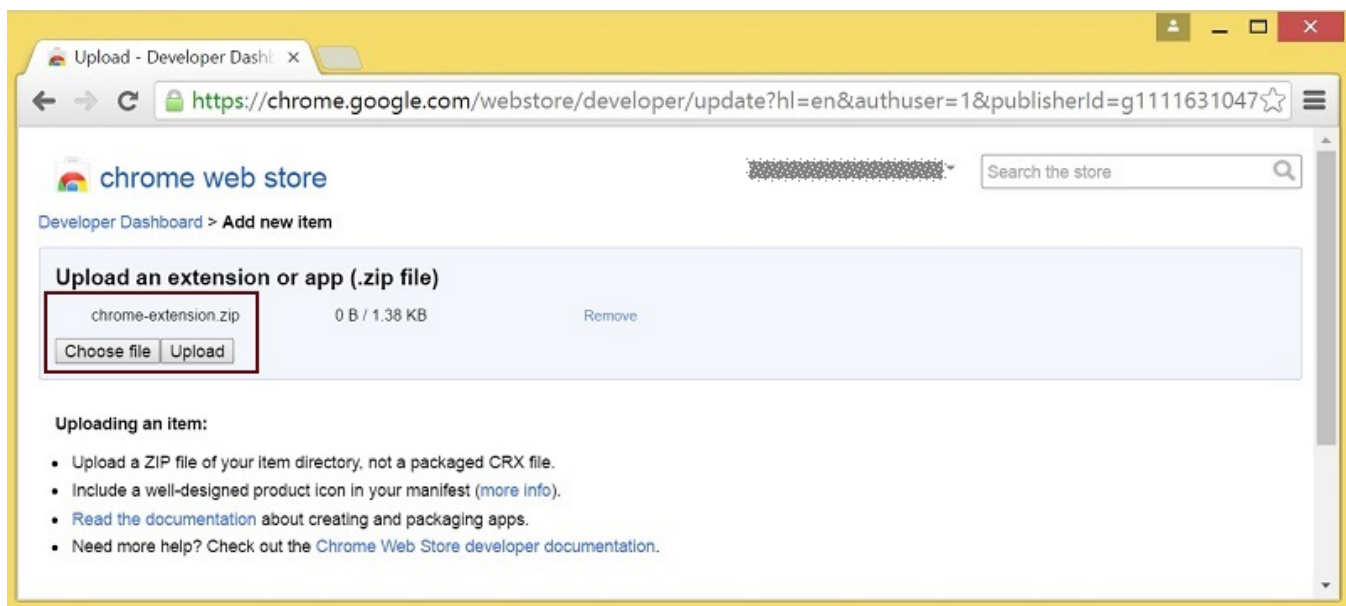
1. Войдите в Chrome Developer Dashboard
2. Кликните кнопку 'Add new item'



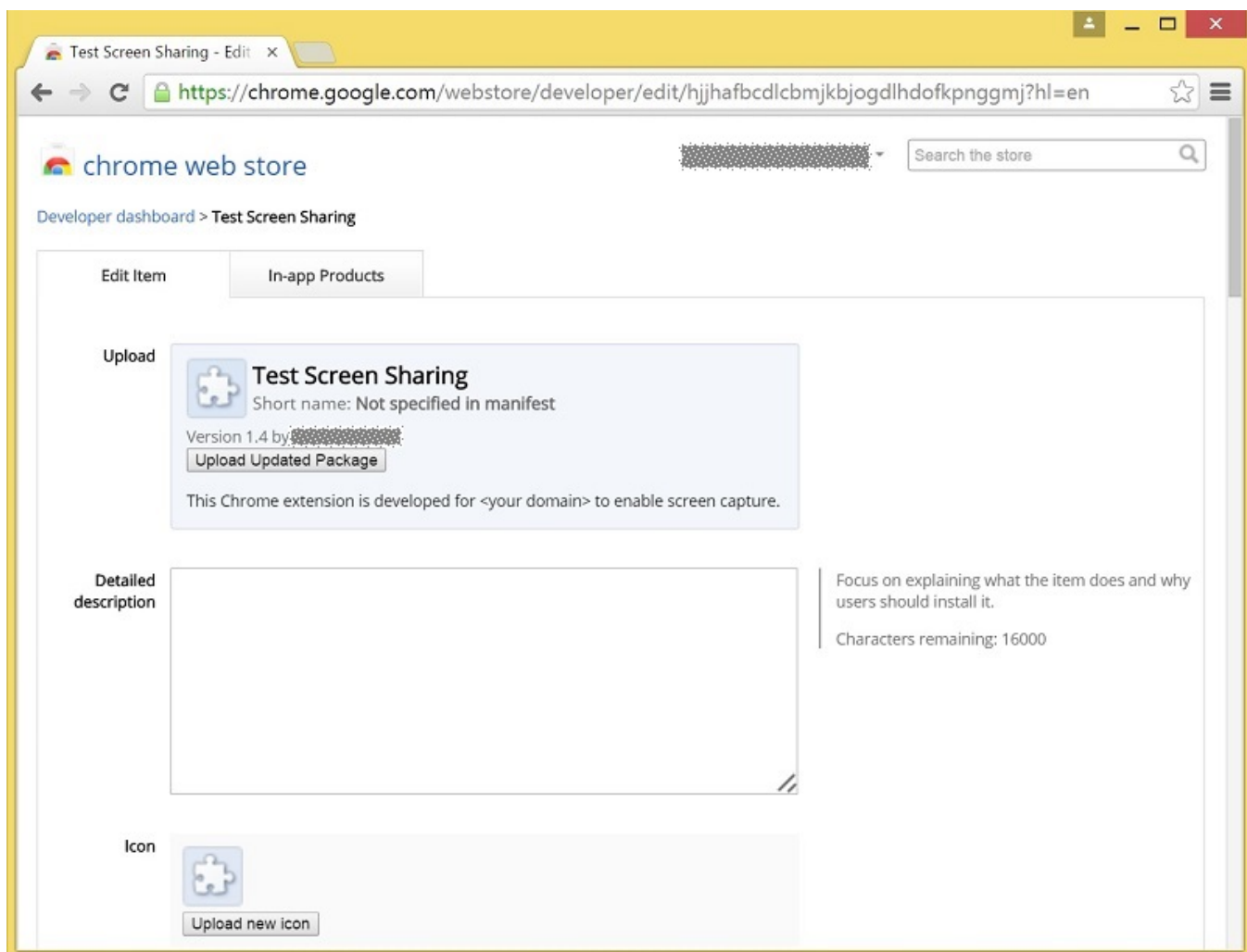
3. Примите соглашение разработчика



4. Выберите файл chrome-extension.zip и кликните кнопку 'Upload' на странице 'Upload'



5. После загрузки расширения будет открыта страница для редактирования опций. Выберите необходимые опции и кликните кнопку 'Save draft and return to dashboard' внизу страницы.

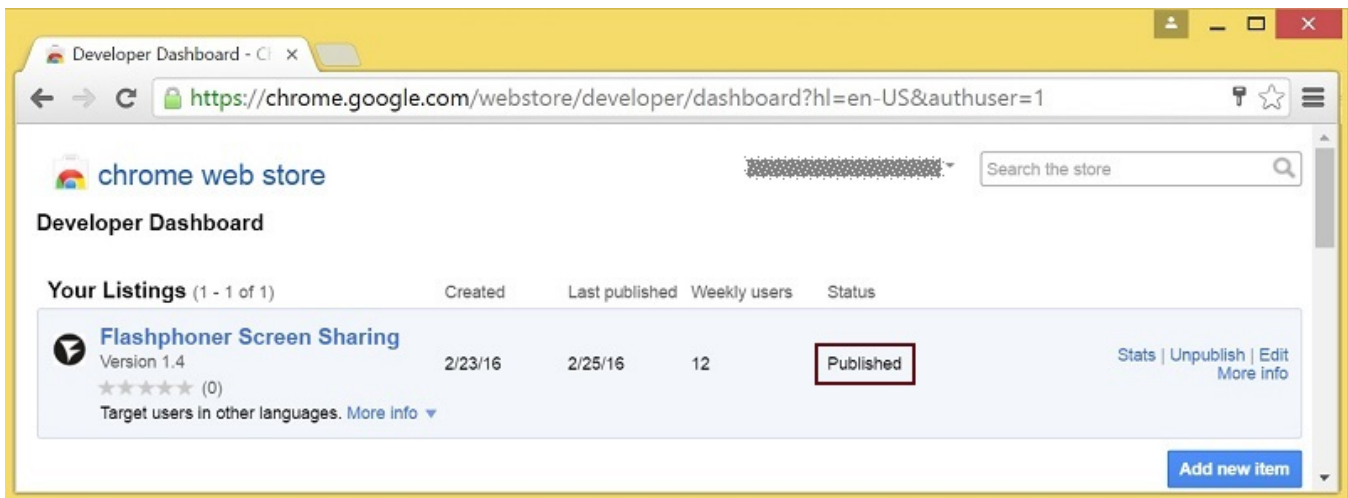


The screenshot shows the 'Edit' page for a Chrome Web Store extension. The URL is <https://chrome.google.com/webstore/developer/edit/mclajopigkalbkpcdemgdgdcdbmlibpch?hl=en&authuser=☆>. The page has several sections: 'Inline install' with a checked checkbox 'This item uses inline install.' and a note to select this checkbox to allow inline install; 'Analytics' with a text input for 'Google Analytics ID UA-' and a placeholder '(example: 1234567-1)'; 'Requirements' with the text 'No requirements'; and 'Visibility options' with three radio buttons: 'Public' (selected), 'Unlisted', and 'Private'. At the bottom, there are three buttons: 'Discard draft', 'Save draft and return to dashboard' (highlighted with a red box), 'Preview changes', and 'Publish changes'.

6. Расширение появится в панели разработчика
Для публикации расширения кликните ссылку 'Publish'.

The screenshot shows the 'Developer Dashboard' page for the Chrome Web Store. The URL is <https://chrome.google.com/webstore/developer/dashboard/g15527730490755806720?hl=en>. The page features the 'chrome web store' logo, a search bar, and a yellow banner about the 'Multiple Platform feature'. Below this is a table titled 'Your Listings (1 - 1 of 1)' with columns: 'Created', 'Last published', 'Weekly users', and 'Status'. The table contains one entry for 'Test Screen Sharing' (Version 1.4, 4/6/16, 0 weekly users, Draft status). To the right of the entry are links for 'Publish' (highlighted with a red box), 'Edit', and 'More info'. At the bottom right, there is a blue button labeled 'Add new item'.

Опубликованное расширение будет иметь статус 'Published' как на изображении ниже.



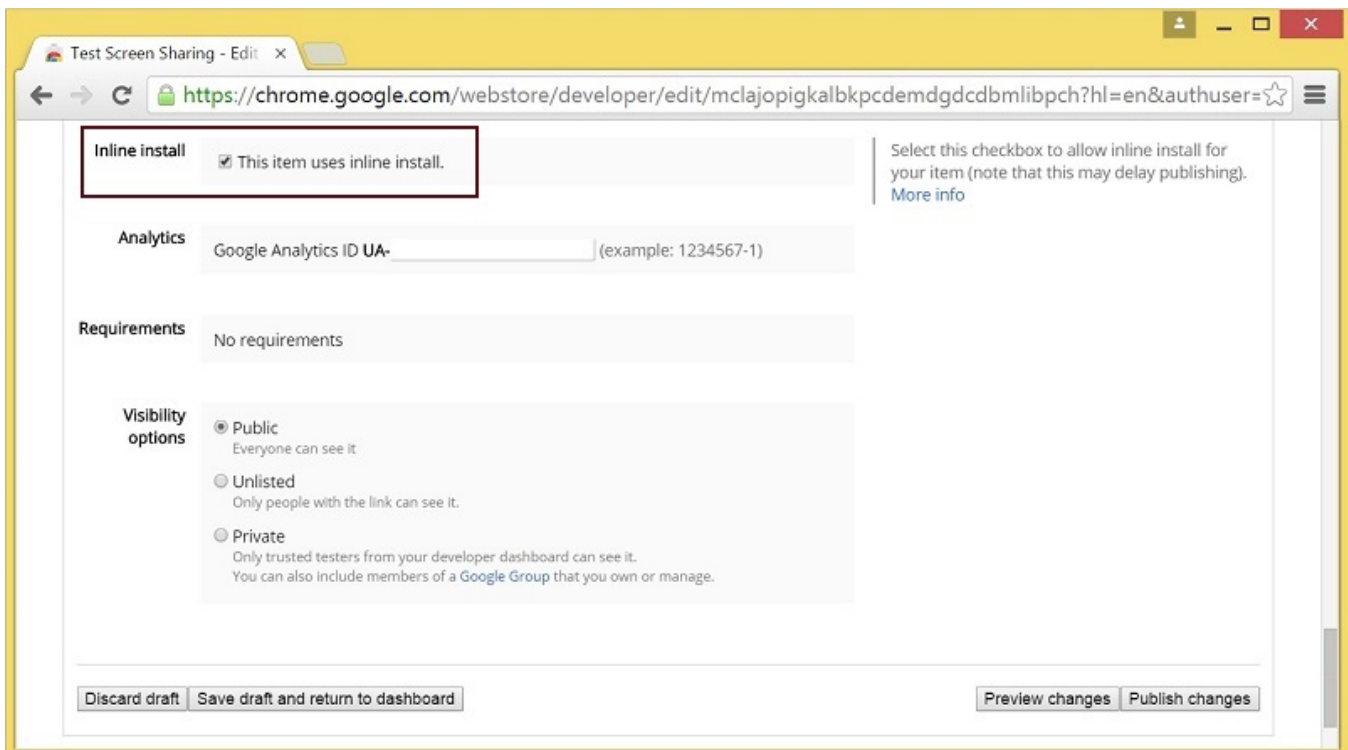
Для дополнительной информации см. [Chrome Web Store Publishing Tutorial](#).

Встроенная установка расширения

Встроенная установка позволяет инициировать установку расширения для демонстрации экрана кликом на ссылке на странице клиента. Для работы встроенной установки расширение должно быть адаптировано, опубликовано и одобрено.

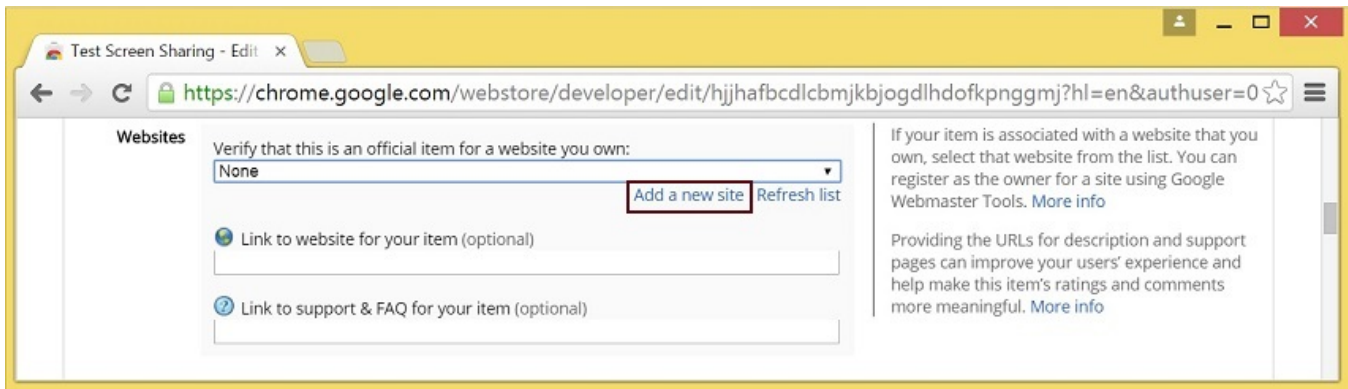
Выполните действия, описанные ниже, чтобы использовать клиент со своими расширениями.

1. При публикации выберите опцию 'Inline Install'

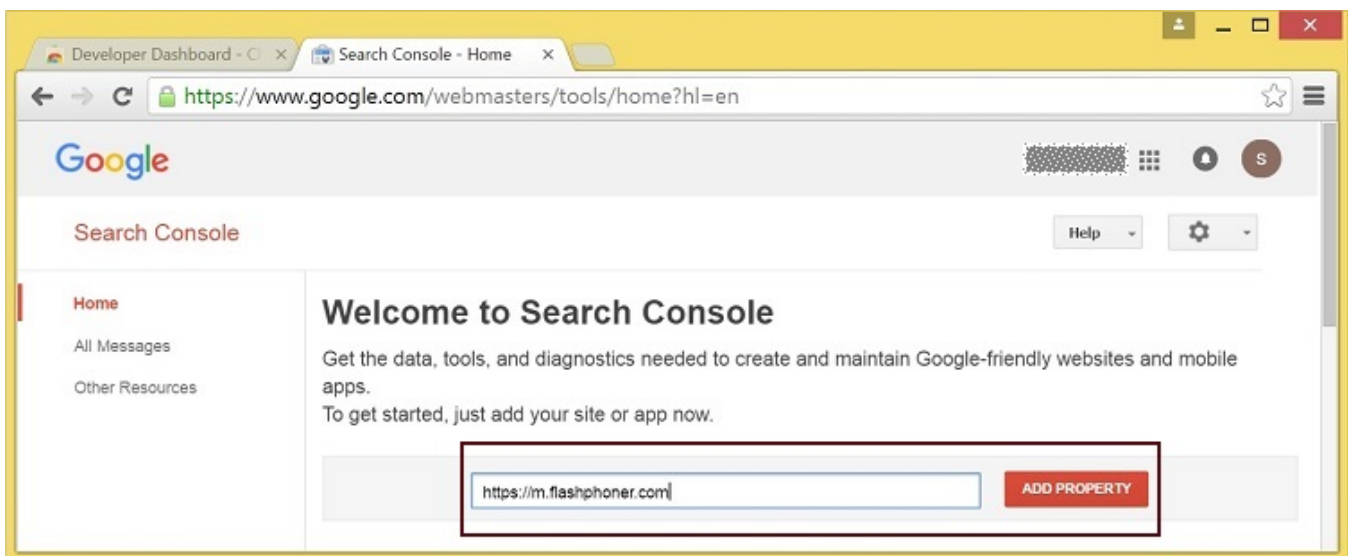


2. Подтвердите и привяжите к расширению сайт со своим доменом

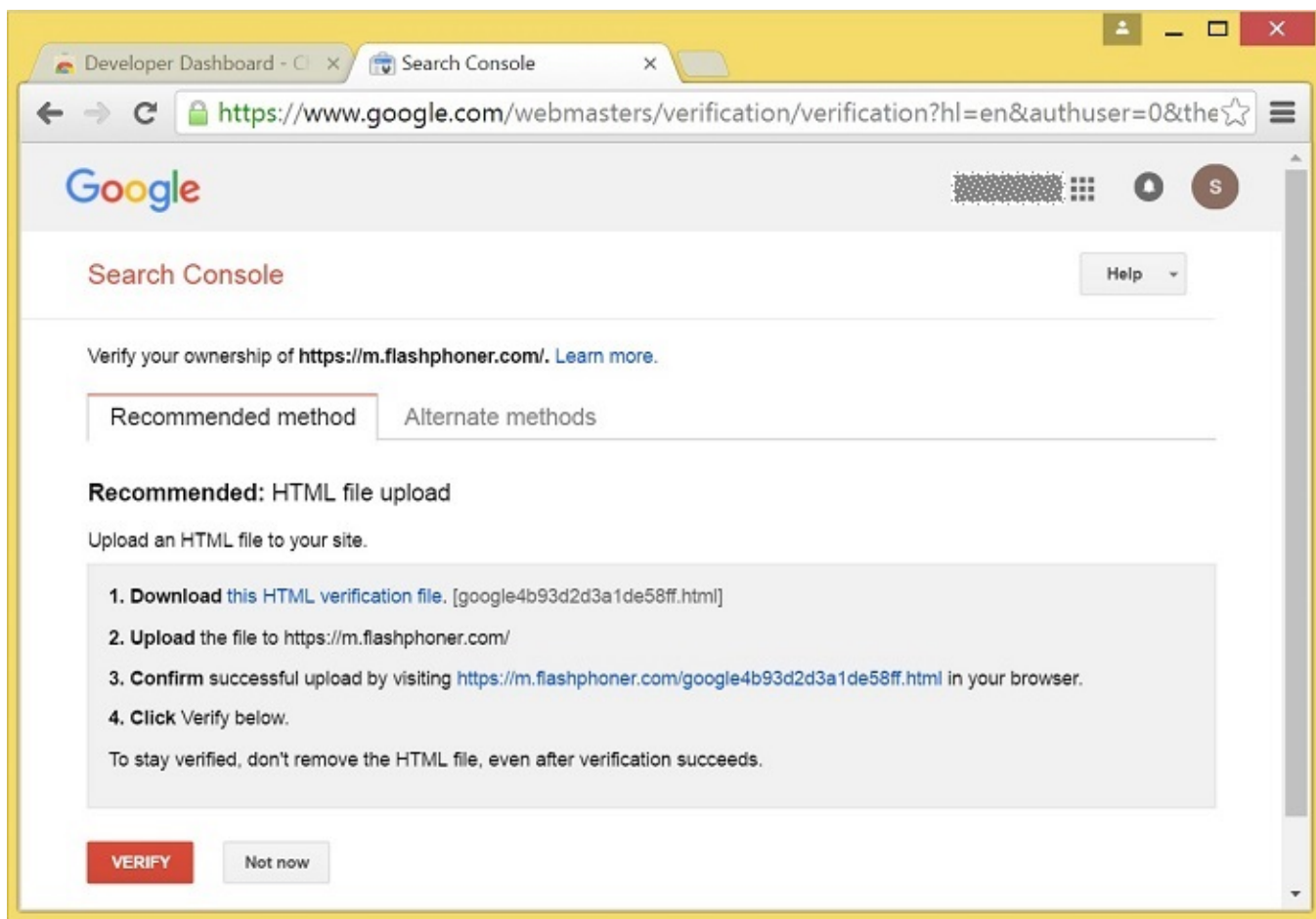
- Кликните 'Add a New Site'



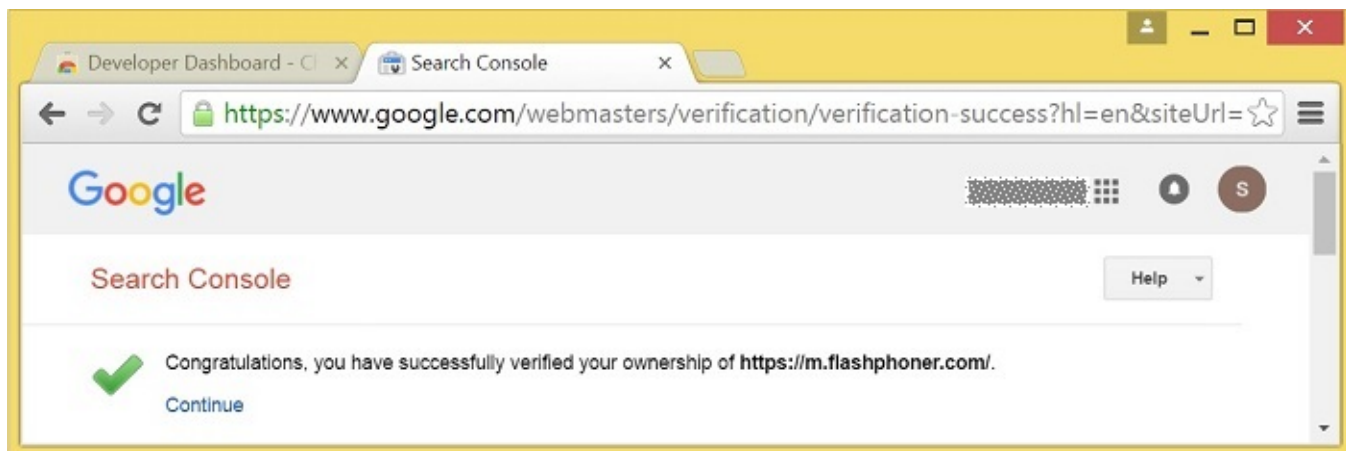
- В новой вкладке браузера будет открыта страница Google Search Console
Введите URL со своим доменом.



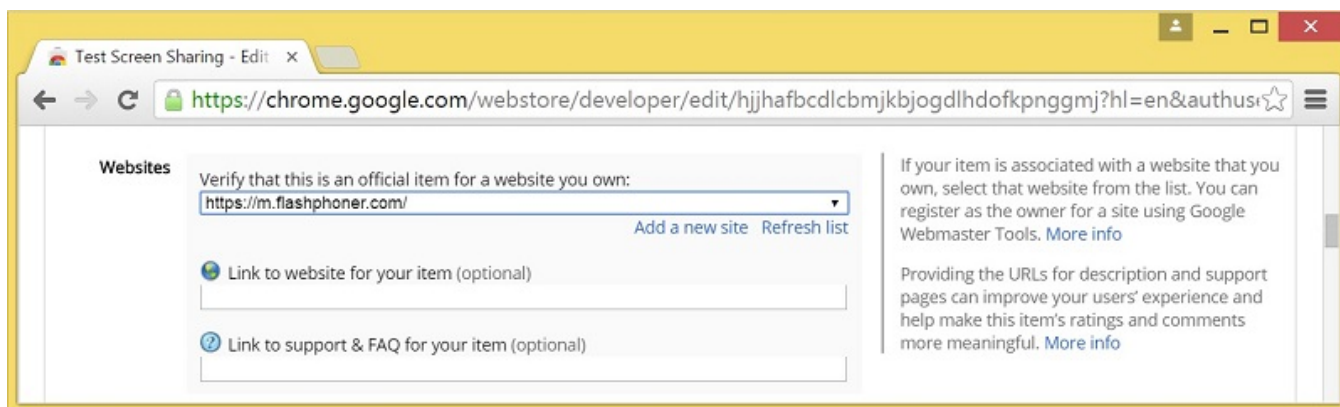
- Будет открыта страница с инструкцией для подтверждения сайта
Выполните требуемые шаги и кликните кнопку 'Verify'.



- Если проверка пройдена, будет открыта страница с подтверждением успешной проверки



- Подтвержденный сайт появится в списке в опциях расширения; далее расширение может быть проассоциировано с этим сайтом

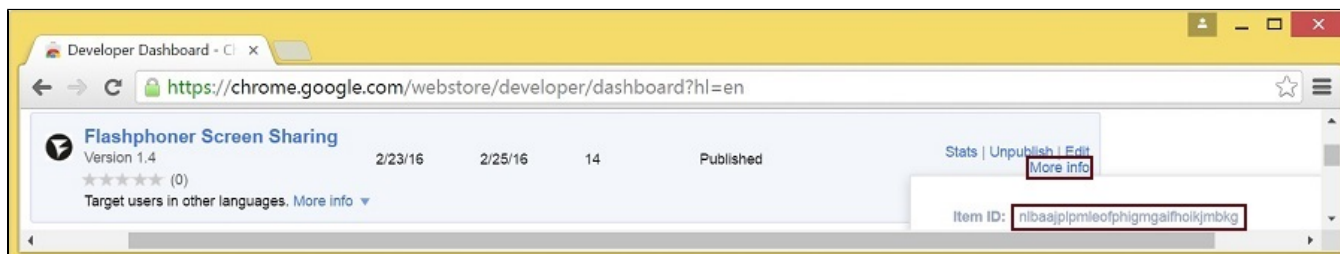


Настройка клиента

Отредактируйте Screen-sharing.html и Screen-sharing.js

- В Screen-sharing.html chrome-webstore-item должен указывать на ваше расширение в интернет-магазине Chrome
- В Screen-sharing.js замените значение переменной chromeScreenSharingExtensionId на ID вашего расширения

Чтобы узнать ID расширения, кликните 'More info' для этого расширения в [Chrome Developer Dashboard](#).



Параметры источника медиа

Для конфигурации параметров источника медиа экрана могут быть использованы параметры объекта Configuration, передаваемые методу init() при инициализации экземпляра Flashphoner API.

```
var f = Flashphoner.getInstance();
var configuration = new Configuration();
....
configuration.screenSharingVideoWidth = 1920;
configuration.screenSharingVideoHeight = 1080;
configuration.screenSharingVideoFps = 10;
f.init(configuration);
```

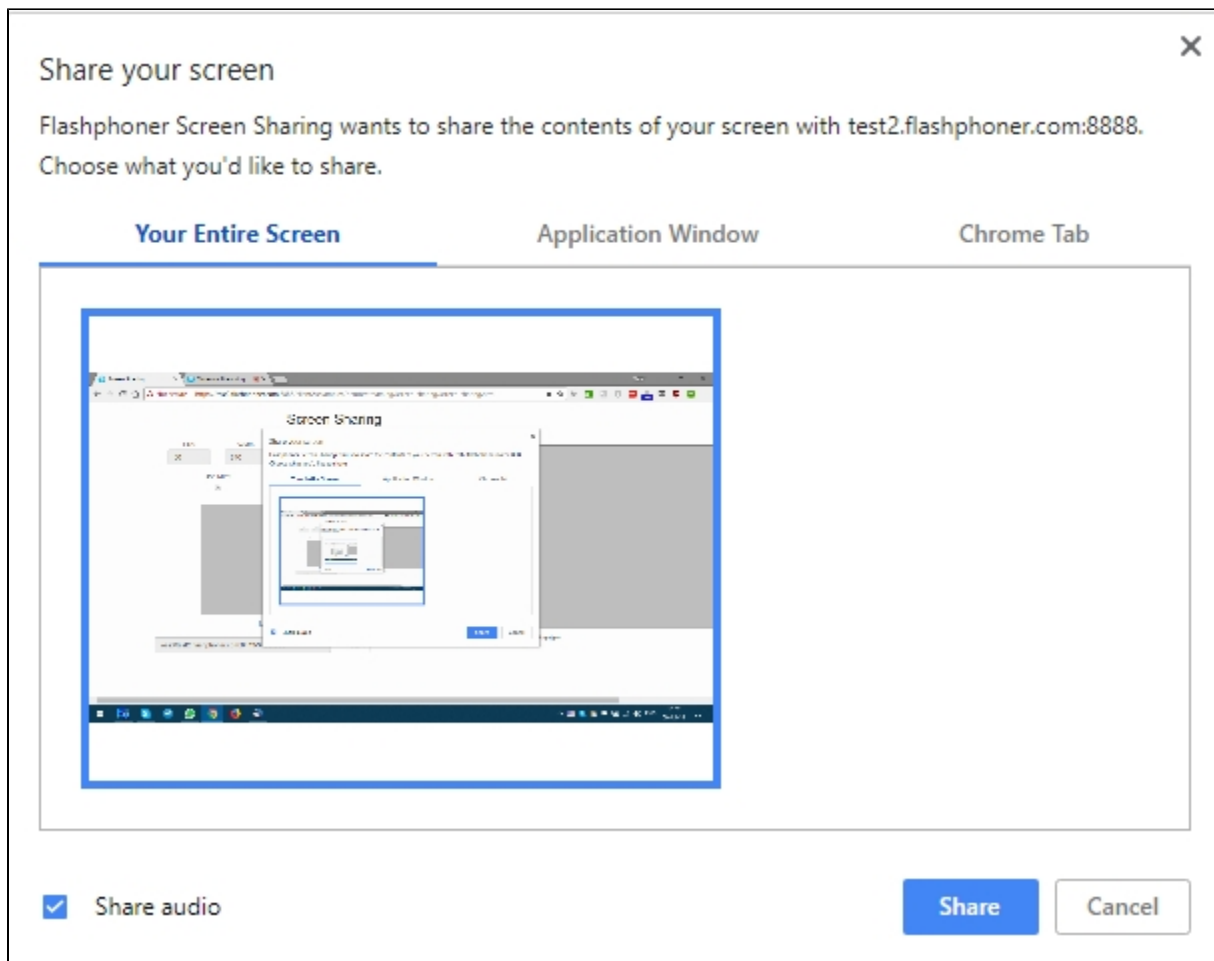
Список параметров

Параметр	Описание
screenSharingVideoWidth	Ширина источника медиа экрана
screenSharingVideoHeight	Высота источника медиа экрана
screenSharingVideoFps	Частота кадров источника медиа экран

Данные параметры задают предельные значения разрешения и количество кадров в секунду (FPS). Например, screenSharingVideoWidth = 1080 означает, что ширина исходного видео не может быть более 1080 пикселей, но может быть меньше (напр., в случае передачи потока с изображением окна приложения с шириной 720 пикселей.)

Захват системного звука в браузере Chrome

В браузере Chrome существует возможность при захвате экрана транслировать аудиопоток из системного источника звука. Такая возможность полезна, например, при скринкастинге. Для захвата системного звука при выборе окна или вкладки в диалоговом окне расширения Chrome установите опцию "Share audio":



Код расширения [code](#):

```
callback({sourceId: sourceId, systemSoundAccess: opts.canRequestAudioTrack});
```

Управление источником захвата (экран или окно) в браузере Firefox

В браузере Firefox можно выбрать экран или окно программы в качестве источника видео при помощи параметра `constraints.video.mediaSource`

[code](#):

```
constraints.video.type = "screen";
if (Browser.isFirefox()){
    constraints.video.mediaSource = $('#mediaSource').val();
}
session.createStream({
    name: streamName,
    display: localVideo,
    constraints: constraints
})
```

Пример интерфейса для выбора источника

FPS	Width	Height	Source
30	640	480	Screen
Use Mic?		Mic	Screen
☐		default: Microphone (Realtek)	Window
			
My Screen			
wss://test2.flashphoner.com:8443/f5b78c77			Start

Захват окна программы

Screen Sharing

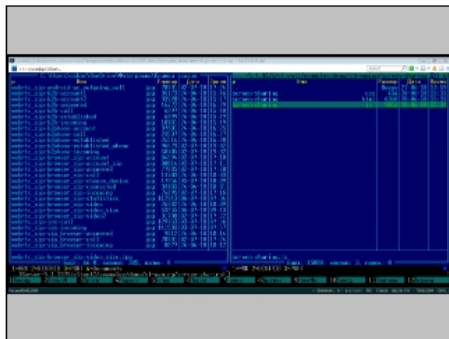
FPS Width Height Source
30 640 480 Window

Use Mic?

☐

Mic

default: Microphone (Realtek)

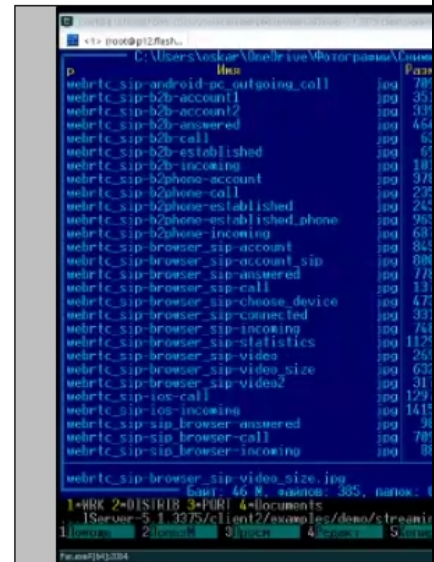


My Screen

wss://test2.flashphoner.com:8443/f5b78c77

Stop

PUBLISHING



Preview

Захват экрана

Screen Sharing

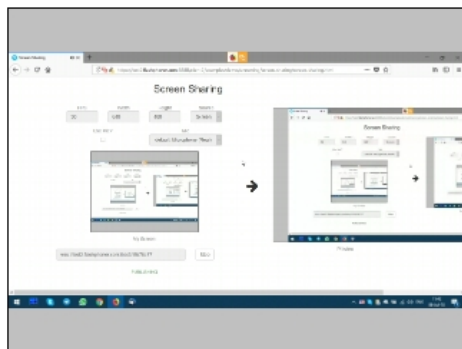
FPS Width Height Source
30 640 480 Screen

Use Mic?

☐

Mic

default: Microphone (Realtek)



My Screen

wss://test2.flashphoner.com:8443/f5b78c77

Stop

PUBLISHING



Preview

Демонстрация экрана без использования расширения

Браузер Firefox

Браузер Firefox не использует расширение для захвата экрана

Браузеры на основе Chrome

Начиная с версии Chrome 73 и версии Flashphoner WebSDK [0.5.28.2753.86](#) возможна демонстрация экрана без использования расширения. Для этого необходимо при создании потока передать параметр `constraints.video.withoutExtension`

[code](#)

```
if ($("#woChromeExtension").prop('checked')) {  
    constraints.video.withoutExtension = true;  
}
```

Браузер Safari для MacOS

Начиная с версии Safari 13 и версии Flashphoner WebSDK [0.5.28.2753.152](#) возможна демонстрация экрана без использования расширения. Для этого необходимо при создании потока передать параметр `constraints.video.withoutExtension`

[code](#)

```
if ($("#woChromeExtension").prop('checked') || Browser.isSafari()) {  
    constraints.video.withoutExtension = true;  
}
```

Известные ограничения

1. При публикации из Chrome, разрешение и FPS устанавливаются не в соответствии с заданными в `constraints` при создании потока, а по размерам источника (экрана, окна или вкладки браузера) и по реальной частоте изменения картинки. Эта проблема исправлена, начиная со сборки Web SDK [0.5.28.2753.152](#)
2. Захват системного звука работает начиная с версии Chrome 74

Известные проблемы

- 1) Если веб-приложение расположено внутри `iframe` элемента, публикация видеопотока может не пройти.

Симптомы: ошибки `IceServer error` в консоли браузера.

Решение: вынести приложение из `iframe` на отдельную страницу.

- 2) Если публикация потока идет с Windows 10 или Windows 8 и в браузере Google Chrome включено аппаратное ускорение, могут быть проблемы с битрейтом.

Симптомы: качество видео плохое, мутное, битрейт в <chrome://webrtc-internals> показывает меньше 100 kbps.

Решение: отключите аппаратное ускорение в браузере, переключите браузер или сервер на использование кодека VP8.

- 3) Остановка всех потоков, захваченных с экрана, при остановке одного из них

Симптомы: при создании нескольких потоков, захваченных с экрана, из одной вкладки браузера Chrome, и последующей остановке одного из них, останавливаются все потоки.

Решение: кешировать дорожки по источнику видео, и останавливать их вместе с последним потоком, использующим этот источник, например

```
var handleUnpublished = function(stream) {
    console.log("Stream unpublished with status " + stream.status());
    //get track label
    var video = document.getElementById(stream.id() + LOCAL_CACHED_VIDEO);
    var track = video.srcObject.getVideoTracks()[0];
    var label = track.label;
    //see if someone using this source
    if (countDisplaysWithVideoLabel(label) > 1) {
        //remove srcObject but don't stop tracks
        pushTrack(track);
        video.srcObject = null;
    } else {
        var tracks = popTracks(track);
        for (var i = 0; i < tracks.length; i++) {
            tracks[i].stop();
        }
    }
    //release resources
    Flashphoner.releaseLocalMedia(streamVideoDisplay);
    //remove stream display
    display.removeChild(streamDisplay);
    session.disconnect();
};
```