

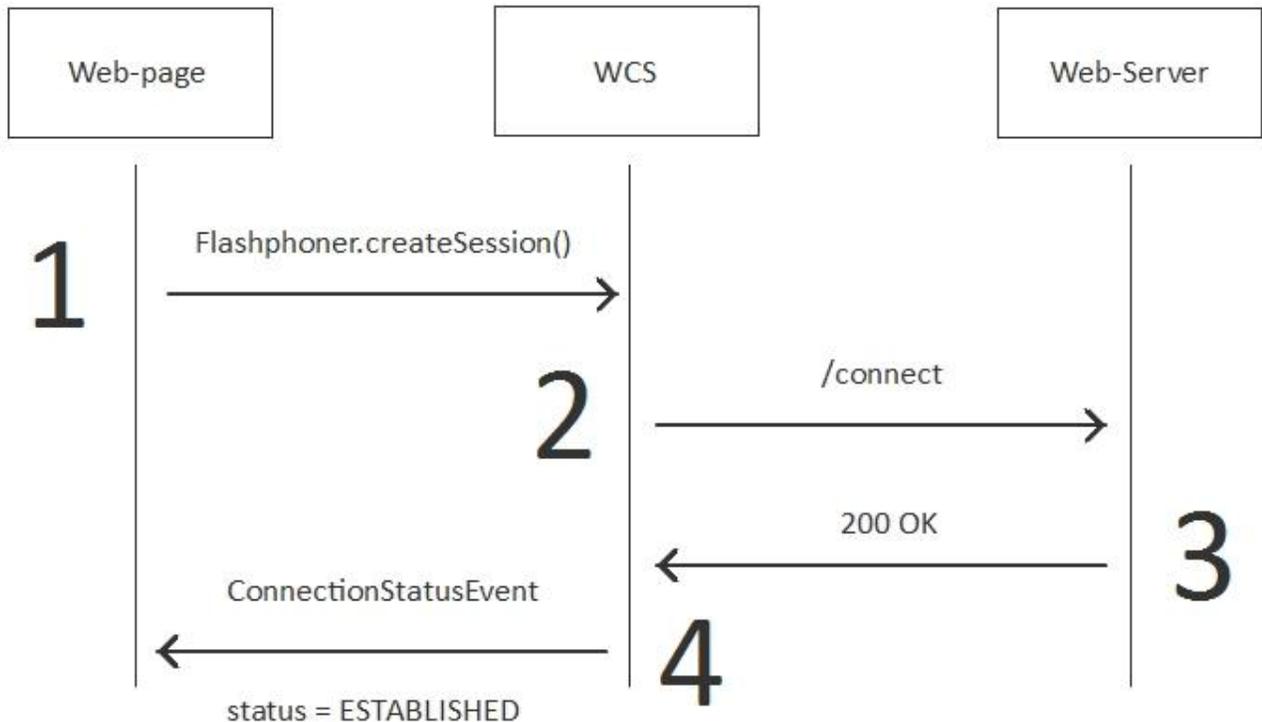
# Тип 1 - connect

- [Описание работы](#)
- [Требования к реализации ответа на бэкенде](#)
- [Аутентификация](#)
- [Настройка других REST-методов](#)

## Описание работы

Этот метод вызывается только один раз при подключении страницы браузера к WCS-серверу по протоколу WebSocket.

Выполнение метода connect является критичным, т.к. этот метод отвечает за аутентификацию подключения к серверу, и если он не пройдет, или будет завершен с ошибкой, то WCS-сервер отклонит данную попытку коннекта.



1. На стороне браузера вызывается `Flashphoner.createSession()`, после чего браузер пытается установить соединение с WCS
2. WCS вызывает REST-метод **connect**
3. WCS получает ответ 200 OK и на основании этого ответа, авторизует эту попытку соединения.
4. WCS отправляет браузеру подтверждение в виде события `ConnectionStatusEvent` со статусом `ESTABLISHED`.

Пример:

2	3
---	---

```
POST /EchoApp/connect HTTP/1.1
Accept: application/json, application/*+json
Content-Type: application/json;charset=UTF-8
User-Agent: Java/1.8.0_45
Host: localhost:8081
Connection: keep-alive
Content-Length: 537

{
  "nodeId": "H4gfHeULTx6ddGGUWwZxhUNyqZHUFH8j@192.168.1.59",
  "appKey": "defaultApp",
  "sessionId": "/192.168.1.38:64604/192.168.1.59:8443",
  "useWsTunnel": false,
  "useWsTunnelPacketization2": false,
  "useBase64BinaryEncoding": false,
  "mediaProviders": [
    "WebRTC",
    "WSPlayer"
  ],
  "clientVersion": "0.5.16",
  "clientOSVersion": "5.0 (Windows NT 6.3; WOW64)",
  "clientBrowserVersion": "Mozilla/5.0 (Windows NT 6.3; WOW64)"
}
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sun, 26 Feb 2017 23:54:06 GMT
Content-Length: 537

{
  "nodeId": "H4gfHeULTx6ddGGUWwZxhUNyqZHUFH8j@192.168.1.59",
  "appKey": "defaultApp",
  "sessionId": "/192.168.1.38:64604/192.168.1.59:8443",
  "useWsTunnel": false,
  "useWsTunnelPacketization2": false,
  "useBase64BinaryEncoding": false,
  "mediaProviders": [
    "WebRTC",
    "WSPlayer"
  ],
  "clientVersion": "0.5.16",
  "clientOSVersion": "5.0 (Windows NT 6.3; WOW64)",
  "clientBrowserVersion": "Mozilla/5.0 (Windows NT 6.3; WOW64)"
}
```

## Требования к реализации ответа на бэкенде

Бэкенд сервер в ответе 200 OK на /connect должен вернуть все поля, полученные в запросе от WCS сервера (зеркальный ответ). Также ответ бэкенд-сервера может содержать дополнительные поля, например restClientConfig (см ниже)

```
{
  "nodeId": "H4gfHeULTx6ddGGUWwZxhUNyqZHUFH8j@192.168.1.59",
  "appKey": "defaultApp",
  "sessionId": "/192.168.1.38:64604/192.168.1.59:8443",
  "useWsTunnel": false,
  "useWsTunnelPacketization2": false,
  "useBase64BinaryEncoding": false,
  "mediaProviders": [
    "WebRTC",
    "WSPlayer"
  ],
  "clientVersion": "0.5.16",
  "clientOSVersion": "5.0 (Windows NT 6.3; WOW64)",
  "clientBrowserVersion": "Mozilla/5.0 (Windows NT 6.3; WOW64)",
  "restClientConfig": {
    ...
  }
}
```

Также возможен вариант пустого ответа без тела, с Content-Length: 0

```
HTTP/1.1 200 OK
Date: Fri, 20 Nov 2020 03:23:57 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Type: application/json
Content-Length: 0
```

При необходимости, значения полей могут быть модифицированы. При этом модификация или удаление содержимого следующих полей не допускается:

```

...
"clientVersion" : "0.5.28",
"clientOSVersion" : "5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36",
"clientBrowserVersion" : "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36",
...

```

В частности, поле `clientBrowserVersion` используется в дальнейшем при установке WebRTC соединения для определения версии DTLS, поддерживаемой браузером. В случае, если это поле отсутствует в ответе бэкенда, или является пустым, будет использоваться DTLS 1.0, и WebRTC не будет работать в последних версиях Chrome и Safari.

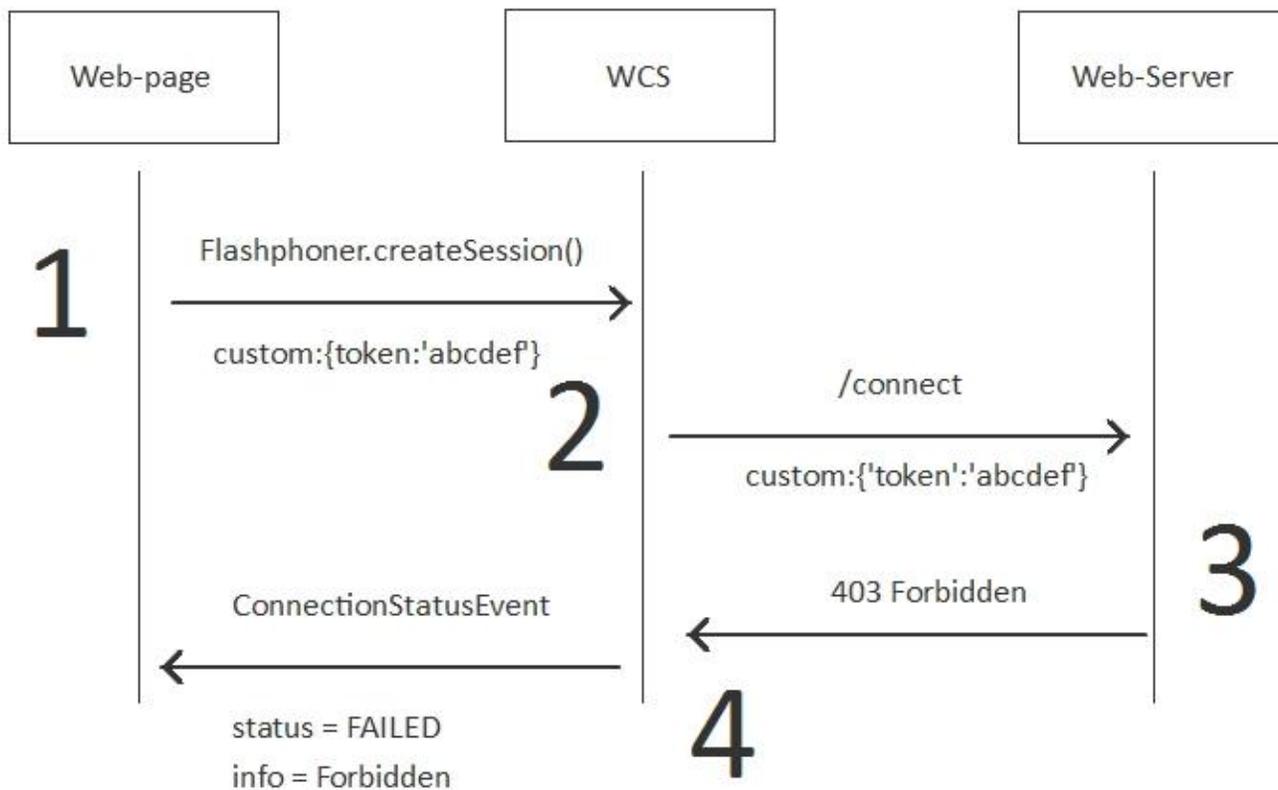
## Аутентификация

По-умолчанию, WCS аутентифицирует успешно все Websocket - коннекты. Т.е. встроенный web-сервер по адресу `http://localhost:8081/apps/EchoApp/connect` всегда возвращает 200 OK.

Вы можете переопределить это поведение и вернуть на вашем web-сервере 403 Forbidden, в этом случае WCS сбросит входящее соединение.

С веб-страницы может быть передан token, пароль или любая другая информация в поле `custom`.

WCS передает это поле web-серверу в теле JSON и web-сервер может принимать решение об аутентификации на основе этих данных.



Пример:

2	3
---	---

```
POST /rest/my_api/connect HTTP/1.1
Accept: application/json, application/*+json
Content-Type: application/json; charset=UTF-8
User-Agent: Java/1.8.0_111
Host: 192.168.1.101
Connection: keep-alive
Content-Length: 578
```

```
{
  "nodeId": "Hw47CFMBEchVOpBMDr29IxjudnJlsjOY@192.168.1.101",
  "appKey": "defaultApp",
  "sessionId": "/192.168.1.102:60204/192.168.1.101:8443",
  "useWsTunnel": false,
  "useWsTunnelPacketization2": false,
  "useBase64BinaryEncoding": false,
  "mediaProviders": [
    "WebRTC",
    "WSPlayer"
  ],
  "clientVersion": "0.5.16",
  "clientOSVersion": "5.0 (Windows NT 6.3; Win64; x64)",
  "clientBrowserVersion": "Mozilla/5.0 (Windows NT 6.3; Win64;
x64)",
  "custom": {
    "token": "abcdef"
  }
}
```

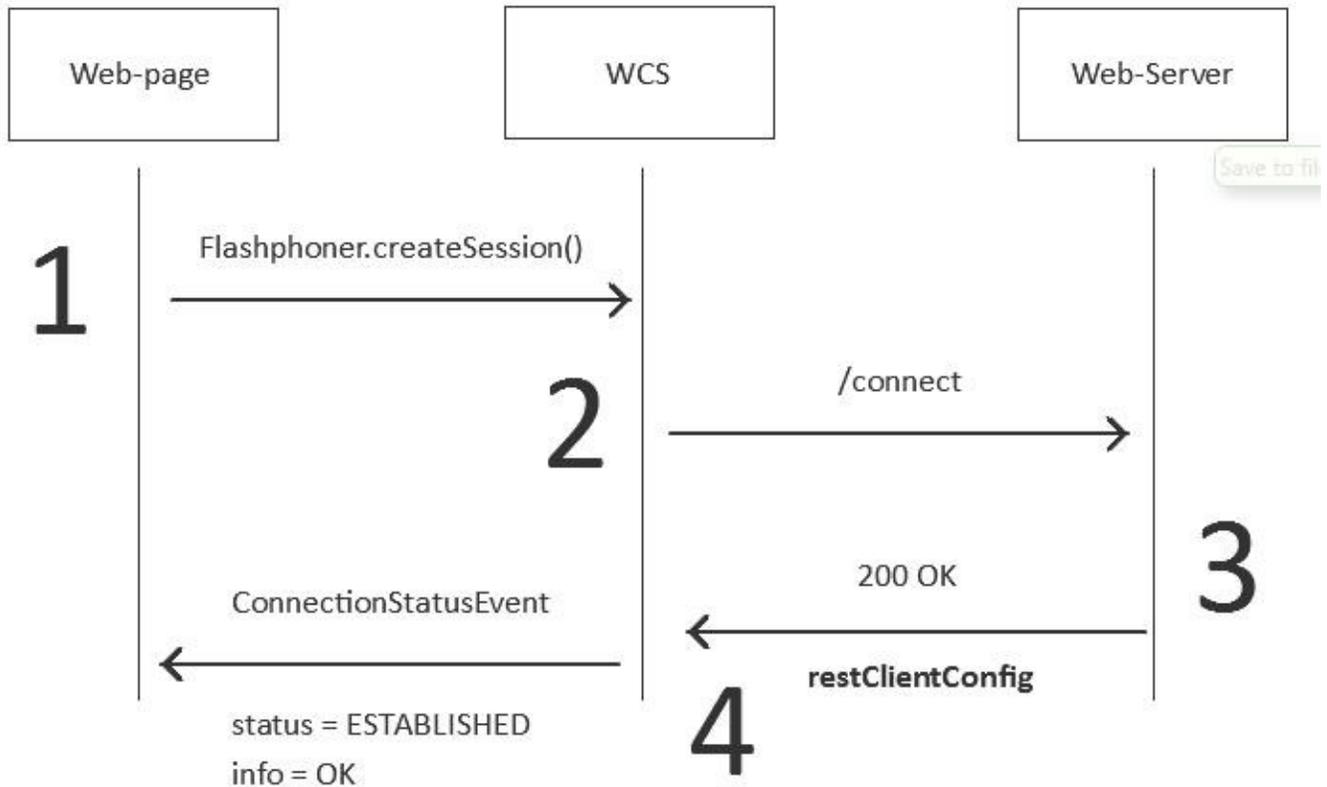
```
HTTP/1.1 403 Forbidden
Date: Tue, 28 Feb 2017 09:05:56 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

## Настройка других REST-методов

Во время аутентификации можно настроить работу всех остальных REST-методов, которые будут вызваны уже после установки соединения, например: `playStream`, `publishStream`, `StreamStatusEvent`, и т.д.

Для этого web-сервер должен вернуть поле `restClientConfig` в теле JSON ответа 200 OK.

`restClientConfig` представляет собой JSON - объект, в котором может быть прописан конфиг для любого из существующих REST-методов, кроме метода `connect`.



Пример тела JSON-ответа 200 OK с полем **restClientConfig**

```

{
  "nodeId" : "kPeSvMn1PFqIMCxZ1Ry6dJ0JNiZ9cqZw@89.179.119.95",
  "appKey" : "defaultApp",
  "sessionId" : "/172.16.16.139:49405/89.179.119.95:8443",
  "useWsTunnel" : false,
  "useWsTunnelPacketization2" : false,
  "useBase64BinaryEncoding" : false,
  "mediaProviders" : [ "WebRTC", "WSPlayer" ],
  "clientVersion" : "0.5.12",
  "clientOSVersion" : "5.0 (Android 5.1.1)",
  "clientBrowserVersion" : "Mozilla/5.0 (Android 5.1.1; Mobile; rv:50.0) Gecko/50.0 Firefox/50.0",
  "restClientConfig":
  {
    "publishStream":
    {
      "restExclude": "",
      "clientExclude": "",
      "restOnError": "FAIL",
      "restPolicy": "NOTIFY",
      "restOverwrite": ""
    }
  }
}
  
```

Этот ответ формируется по следующим правилам:

1. Возвращаем те данные, которые получили от WCS, без изменений.
2. Добавляем поле:

```
"restClientConfig":{
  "publishStream":{
    "restExclude":"","
    "clientExclude":"","
    "restOnError":"FAIL",
    "restPolicy":"NOTIFY",
    "restOverwrite":""
  }
}
```

Эта конфигурация говорит WCS-серверу, что вызов REST-метода **publishStream** будет проходить в соответствии со следующими правилами:

**1.restExclude - пустое значение.**

Это значит, что все поля в теле JSON запроса, такие как name, width, height, и т.д., будут отправлены с WCS на web-сервер, и ни одно из полей не будет исключено из этого запроса.

**2.clientExclude - пустое значение.**

Не используется в REST-методе publishStream.

**3.restOnError: FAIL**

Это значит, что если при обращении к web-серверу произошла ошибка либо если web-сервер вернул HTTP-статус 4xx, то WCS-сервер должен запретить и прервать операцию публикации потока.

По-умолчанию в этом поле используется restOnError: LOG. Т.е. По-умолчанию, WCS просто логирует все возникшие ошибки, не запрещает и не прерывает выполнение операций, таких как publishStream, даже если web-сервер вернул статус ошибки 403 Forbidden.

**4.restPolicy: NOTIFY**

Это значит, что если web-сервер вернет данные, отличные от тех, что принял, они не будут применены. Например, если WCS отправил поле name=stream1, а web-сервер вернул name=stream2 в теле ответа 200 OK, то новое значение не будет применено, т.к. установлена политика NOTIFY.

**5.restOverwrite - пустое значение.**

Эта настройка применяется только в том случае, когда предыдущая restPolicy установлена в значение OVERWRITE, т.е. позволяет переопределять поля тем, что пришло в теле JSON ответа 200 OK. Текущее значение пустое, тем самым ни одно поле не может быть переопределено. Для переопределения должен быть явно указан список полей, например: restOverwrite="name,width"

Пример:

2	3
---	---

```
POST /rest/my_api/connect HTTP/1.1
Accept: application/json, application/*+json
Content-Type: application/json;charset=UTF-8
User-Agent: Java/1.8.0_111
Host: 192.168.1.101
Connection: keep-alive
Content-Length: 550

{
  "nodeId": "Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.
168.1.101",
  "appKey": "defaultApp",
  "sessionId": "/192.168.1.102:26518/192.168.1.101:
8443",
  "useWsTunnel": false,
  "useWsTunnelPacketization2": false,
  "useBase64BinaryEncoding": false,
  "mediaProviders": [
    "WebRTC",
    "WSPlayer"
  ],
  "clientVersion": "0.5.16",
  "clientOSVersion": "5.0 (Windows NT 6.3; Win64;
x64)",
  "clientBrowserVersion": "Mozilla/5.0 (Windows NT
6.3; Win64; x64)"
}
```

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 10:11:03 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Content-Length: 833
Connection: close
Content-Type: application/json

{
  "nodeId": "Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.
168.1.101",
  "appKey": "defaultApp",
  "sessionId": "\/192.168.1.102:26518\/192.
168.1.101:8443",
  "useWsTunnel": false,
  "useWsTunnelPacketization2": false,
  "useBase64BinaryEncoding": false,
  "mediaProviders": [
    "WebRTC",
    "WSPlayer"
  ],
  "clientVersion": "0.5.16",
  "clientOSVersion": "5.0 (Windows NT 6.3; Win64;
x64)",
  "clientBrowserVersion": "Mozilla\/5.0 (Windows NT
6.3; Win64; x64)",
  "restClientConfig": {
    "publishStream": {
      "clientExclude": "",
      "restExclude": "recordName",
      "restOnError": "FAIL",
      "restPolicy": "NOTIFY",
      "restOverwrite": ""
    },
    "playStream": {
      "clientExclude": "",
      "restExclude": "",
      "restOnError": "LOG",
      "restPolicy": "OVERWRITE",
      "restOverwrite": "height,width"
    }
  }
}
```