# Stopping the video stream on the server side



## Overview

Any stream published or captured by WCS can be stopped on the server side.
To stop publishing or playing the stream on the side of the WCS server, use one of the following:

1. Return HTTP 403 FORBIDDEN state in response to a keep-alive query for the stream
2. Make a REST-query to the WCS server

## Keep-Alive of a video stream

Keep-alive messages can be used to terminate streams by the initiative of the WCS server.
To do this, the web server where REST Hooks are configured should return the HTTP 403 FORBIDDEN state in response to a keep-alive request for the stream.

This way you can stop publishing or playing the stream, or both..
To distinguish published and played streams, the script of the web server should handle the 'published' parameter of the StreamKeepAliveEvent method.

### REST API

Configure the web server to use REST Hooks. The server must handle keep-alive requests from the WCS server using, for example, a PHP script and define restClientConfig for the StreamKeepAliveEvent method.

```
"StreamKeepAliveEvent" : {
    "clientExclude" : "",
    "restExclude" : "sdp",
    "restOnError" : "FAIL",
    "restPolicy" : "NOTIFY",
    "restOverwrite" : ""
}
```

### Server side

To enable sending keep-alive messages for streams you need:

1) Enable the keep-alive setting inflashphoner.properties

```
keep_alive_streaming_sessions_enabled=true
```

2) Define the keep-alive interval and the application that will receive responses to keep-alive REST-queries

```
streaming_sessions_keep_alive_interval=10000
streaming_sessions_keep_alive_app_keys=defaultApp
```

You can use an application other than the 'defaultApp'.

UseCommand Line Interfaceand the 'show apps' command to see the list of applications and their keys.

3) Add the StreamKeepAliveEvent REST method to this application using the following command

```
add app-rest-method defaultApp StreamKeepAliveEvent
```

4) Add REST API to the application from the command line

```
update app defaultApp http://my-web-server.com/MyAPI
```

Here:

- my-web-server.com - is the address of the web server,
- MyAPI - is the path where REST API is located.

# REST queries to stop the stream

To stop a stream, use the/stream/terminateREST query.

A REST-query must be an HTTP/HTTPS POST query in the following form:

- HTTP:http://streaming.flashphoner.com:8081/rest-api/stream/terminate
- HTTPS:https://streaming.flashphoner.com:8444/rest-api/stream/terminate

Here:

- streaming.flashphoner.com- is the address of the WCS server
- 8081 - is the standard REST / HTTP port of the WCS server
- 8444- is the standard HTTPS port
- rest-api- is the required part of the URL
- /stream/terminate- is the REST method used

## REST-methods and response statuses

| REST-method | Example of REST query | Example of REST response | Response statuses |
|---|---|---|---|
| /stream/terminate | `{ "mediaSessionId" : "41c3f621-a847-4639" }` | | 200 - Stream termination request is received<br>404 - Stream not found |

## Parameters

| Parameter name | Description | Example |
|---|---|---|
| mediaSessionId | Media session identifier | 41c3f621-a847-4639 |
| name | Stream name | streamName |
| published | If true, the stream is published; if false, the stream is played | true |
| status | Current status of the stream | PUBLISHING |

## Stream filtering by parameters

A /stream/terminate query parameters considered as filters, all the steams that conforms to those fliters will be stopped. For example, we can stop all subscribers for all the streams published

```
{"published": false}
```

or all subsribers for certain stream

```
{"name": "streamName","published": false}
```

Streams published can be stopped by status

```
{"name": "streamName","status": "PUBLISHING"}
```

Also all streams in certain mediasessions can be stopped

```
{"mediaSessionIds":["41c3f621-a847-4639", "554916e0-931c-2479"]}
```

or in one mediasession

```
{"mediaSessionId":"41c3f621-a847-4639"}
```

### Sending the REST query to the WCS server

To send a REST query to the WCS server, you need a REST client, such as Advanced REST Console extension for the Chrome browser.

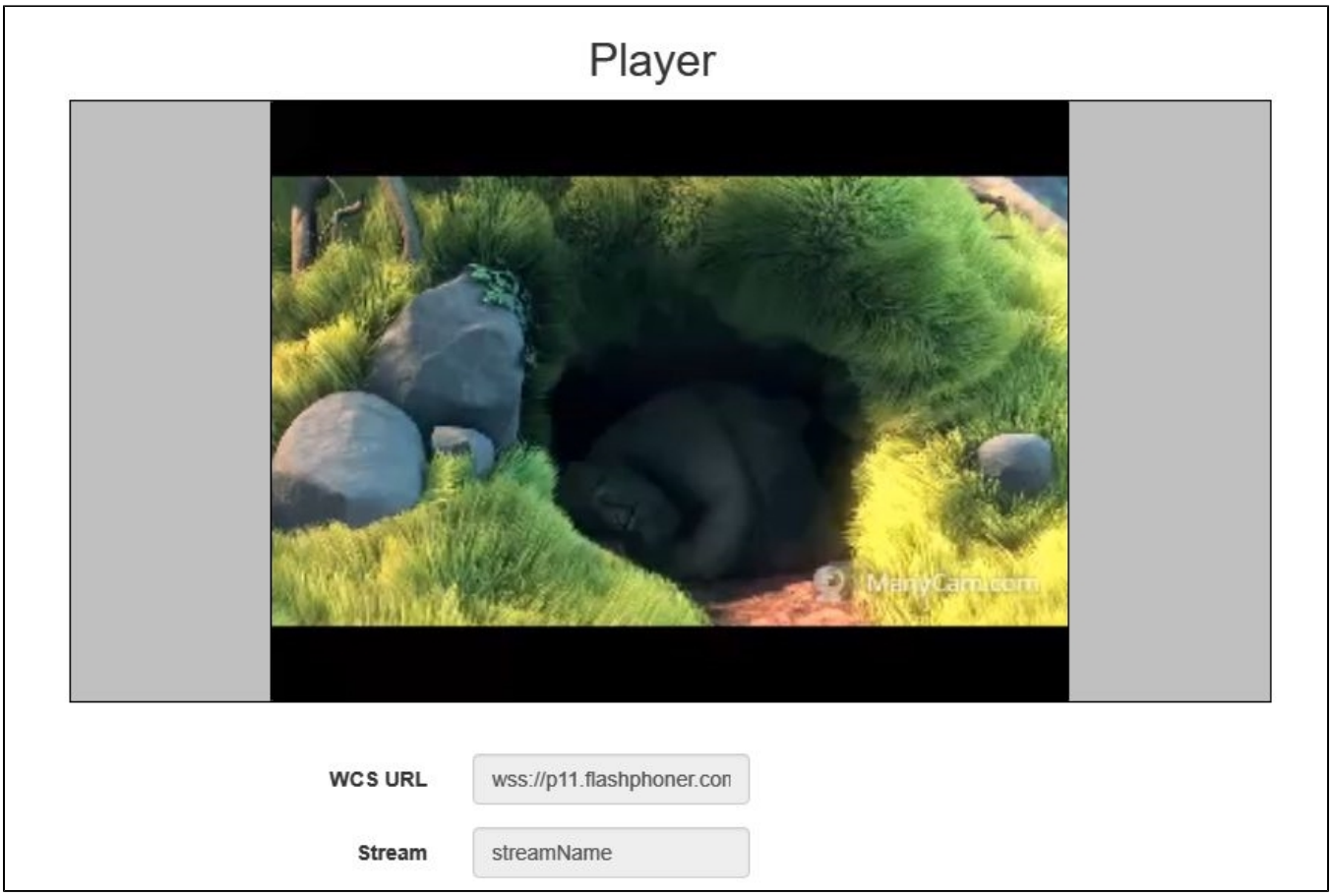
## Resuming the stream

After a stream was stopped from the server side, publishing or playing of the stream can be resumed.

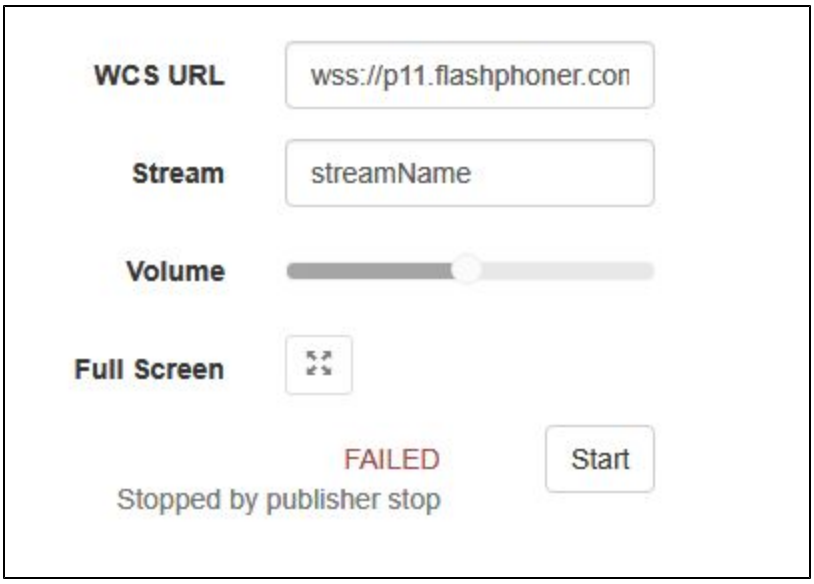
For example, if a WebRTC / WebSocket stream is published using the client2/examples/min/streaming.html client and played using the client2/examples /demo/streaming/player/player.html client:

1) The stream is published


2) The stream is playing




3) Stream playback is stopped by the initiative of the WCS server




Publishing of the stream will resume after the 'publish' button is clicked.
Playing of the stream will resume after the 'Start' button is clicked.

# Known issues

1) "Can't find mediasession" messages in flashphoner.log

Symptoms: If an RTMFP stream is stopped on the server side and if that stream was published using the client2/examples/demo/streaming/flash_client /chat.html client, publishing of the stream stops, but stream session is not terminated. In the flashphoner.log file you can see the following messages: "Can't find mediasession".
Solution: To terminate the session, click the 'Stop video' button on the client page.