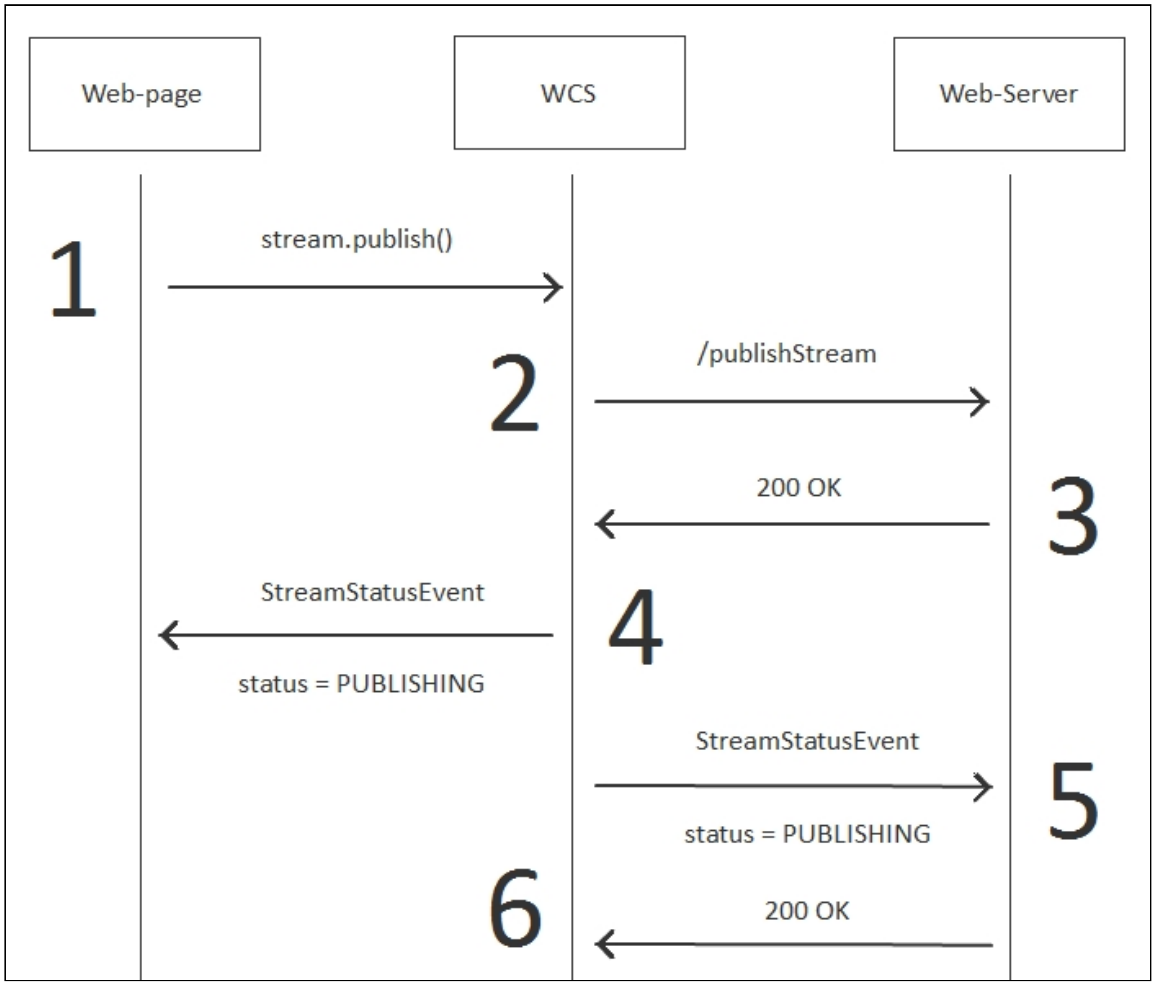
Type 3 - the event

Workflow example using `StreamStatusEvent` hook

The `StreamStatusEvent` REST method is an event. Events occur internally in the WCS server and are used to pass statuses of various operations.

For instance, the `StreamStatusEvent` event is used to pass statuses of video stream operations such as `Stream.play()` and `Stream.publish()`. Indeed, if we publish a video stream, play it, or stop it, we must know its status to manage this stream.

The backend server cannot authenticate (permit or forbid) an event and simply accepts it, for example to save information about the stream in the database.



On this diagram you can see that the `StreamStatusEvent` event goes in two directions:

1. To the client - step 4
2. To the backend server - step 5

Example:

REST hook

```
POST /rest/my_api/StreamStatusEvent HTTP/1.1
Accept: 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: 3614

{
    "nodeId":"Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.168.1.101",
    "appKey":"defaultApp",
    "sessionId":"/192.168.1.102:4388/192.168.1.101:8443",
    "mediaSessionId":"56141d10-fddc-11e6-ac3a-4d67d5b3360d",
    "name":"b4e7",
    "published":true,
    "hasVideo":true,
    "hasAudio":true,
    "status":"PUBLISHING",
    "sdp":".....",
    "record":false,
    "width":0,
    "height":0,
    "bitrate":0,
    "quality":0,
    "mediaProvider":"WebRTC"
}
```

Backend response

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 17:35:44 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Content-Length: 3656
Connection: close
Content-Type: application/json

{
   "nodeId":"Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.168.1.101",
    "appKey":"defaultApp",
    "sessionId":"/192.168.1.102:4388/192.168.1.101:8443",
    "mediaSessionId":"56141d10-fddc-11e6-ac3a-4d67d5b3360d",
    "name":"b4e7",
    "published":true,
    "hasVideo":true,
    "hasAudio":true,
```

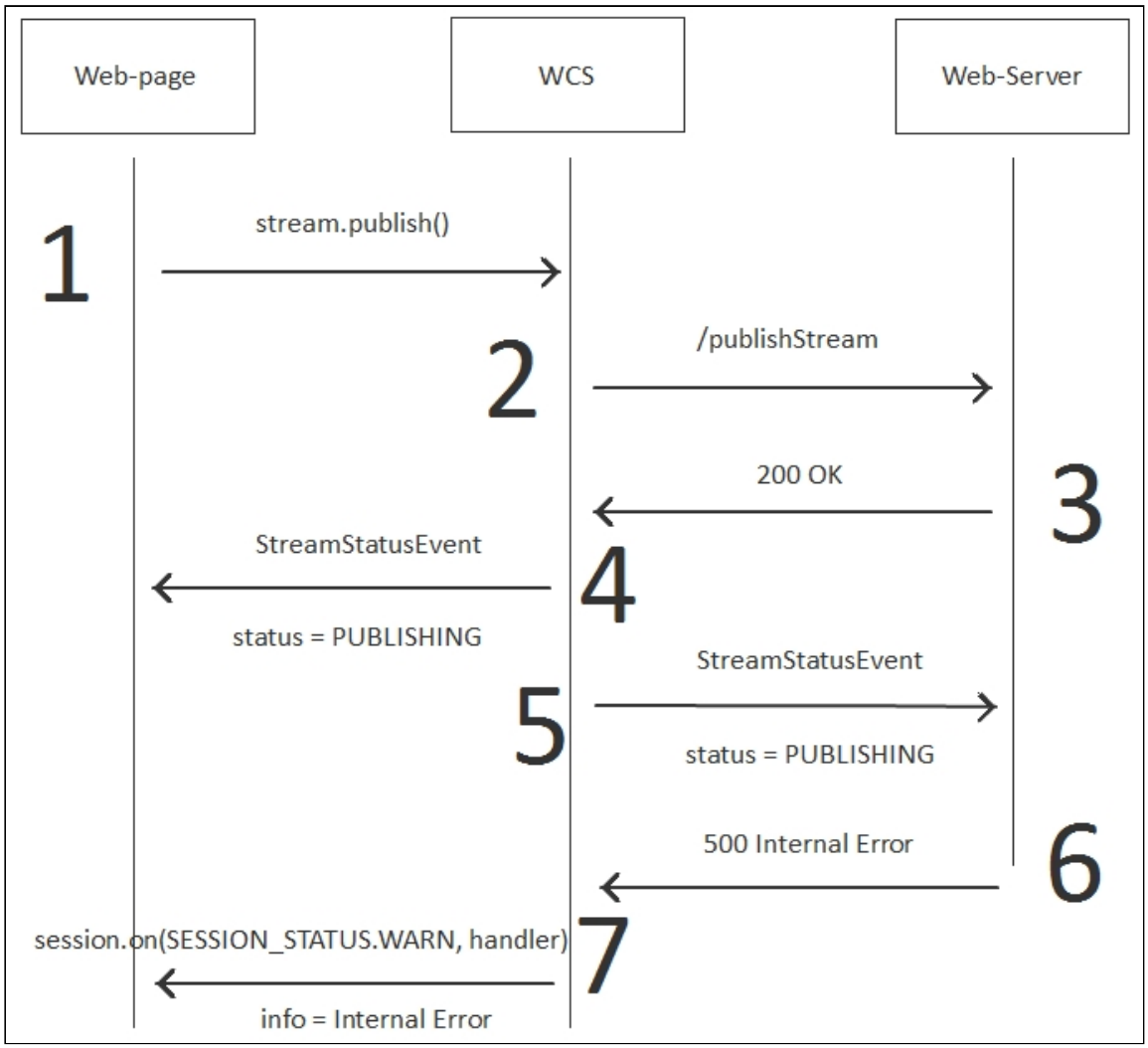
```
    "status":"PUBLISHING",
    "sdp":".....",
    "record":false,
    "width":0,
    "height":0,
    "bitrate":0,
    "quality":0,
    "mediaProvider":"WebRTC"
}
```

Error handling

By default, WCS does not check the status of the response to invocation of the `/StreamStatusEvent` REST method. That is, if the web server returns HTTP error status 403 or 500 or any other, the WCS simply ignores that.

To make WCS react to the error, set `restOnError: FAIL` in the settings of the `StreamStatusEvent` method, in restClientConfig when establishing the connection.

In this case, the client receives an additional event `ErrorEvent` and is notified about the error



On the step 6, the web server returns the HTTP status `500 Internal Error` in response to invocation of the `/StreamStatusEvent` method. The WCS server on the step 7 notifies the client of the error occured.

Example:

REST hook

```
POST /rest/my_api/StreamStatusEvent HTTP/1.1
Accept: 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: 3624

{
    "nodeId":"Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.168.1.101",
    "appKey":"defaultApp",
    "sessionId":"/192.168.1.102:25301/192.168.1.101:8443",
    "mediaSessionId":"e9c002d0-fde2-11e6-a2bf-c99492323844",
    "name":"dc6a",
    "published":true,
    "hasVideo":true,
    "hasAudio":true,
    "status":"PUBLISHING",
    "sdp":".....",
    "record":false,
    "width":0,
    "height":0,
    "bitrate":0,
    "quality":0,
    "mediaProvider":"WebRTC"
}
```

Backend response

```
HTTP/1.1 500 Internal Server Error
Date: Tue, 28 Feb 2017 18:22:49 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
```