Stream event passing to subscribers

Overview

Since build 5.2.935 it is possible to send an event from publishing client which is bound to the stream published, and to pass the event to all the stream subscribers. Today, the feature is used to notify subscribers when publisher mutes/unmutes audio/video.

Sending stream event from publiher

Audio/video state notification: muted/unmuted

Stream audio state notifications are sending when `Stream.muteAudio()` and `Stream.unmuteAudio()` are called as follows:

```
var muteAudio = function muteAudio() {
    if (mediaConnection) {
    mediaConnection.muteAudio();
    sendStreamEvent(STREAM_EVENT_TYPE.AUDIO_MUTED);
    }
};
...
var unmuteAudio = function unmuteAudio() {
    if (mediaConnection) {
    mediaConnection.unmuteAudio();
    sendStreamEvent(STREAM_EVENT_TYPE.AUDIO_UNMUTED);
    }
};
```

Stream video state notifications are sending when `Stream.muteVideo()` and `Stream.unmuteVideo()` are called:

```
var muteVideo = function muteVideo() {
    if (mediaConnection) {
    mediaConnection.muteVideo();
    sendStreamEvent(STREAM_EVENT_TYPE.VIDEO_MUTED);
    }
};
...
var unmuteVideo = function unmuteVideo() {
    if (mediaConnection) {
    mediaConnection.unmuteVideo();
    sendStreamEvent(STREAM_EVENT_TYPE.VIDEO_UNMUTED);
    }
};
```

Sending data to all the subscribers from publishing client

Since WCS build 5.2.942 and WebSDK build 2.0.168 it is possible to send any data in JSON format from publishing client to all the subscribers of stream published. To do this, `Stream.sendData()` method should be invoked, for example

```
stream.sendData({"number":33,"string":"hello",boolean:true});
```

Sending stream event from server

Since WCS build 5.2.944 it is possible to send an event with JSON data to all the stream subscribers from server using REST API.

REST query should be HTTP/HTTPS POST query like this:

- HTTP: `http://test.flashphoner.com:8081/rest-api/stream/event/send`
- HTTPS: `https://test.flashphoner.com:8444/rest-api/stream/event/send`

Where:

- `test.flashphoner.com` - WCS server address
- `8081` - standard REST / HTTP port
- `8444` - standard HTTPS port
- `rest-api` - mandatory part of the URL
- `/stream/event/send` - REST method used

REST queries and responses

/stream/event/send

Send data to all the stream subscribers

REQUEST EXAMPLE

```
POST /rest-api/stream/event/send HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
    "streamName":"test",
    "payload":{
        "number":33,
        "string":"hello",
        "boolean":true
    }
}
```

RESPONSE EXAMPLE

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
```

RETURN CODES

Code	Reason
200	OK
404	Not found
500	Internal error

Parameters

Parameter	Description	Example
streamName	Stream name	`test`
payload	JSON data	`{ "number":33, "string":"hello", "boolean":true }`

If the stream published on server has no subscribers, then the query will return `200 OK`, but the event will not be sent to anyone

Receiving stream event on subscribers side

When stream event is passed, all the subscribers receive the `STREAM_EVENT` event

code

```
previewStream = session.createStream({
    name: streamName,
    display: remoteVideo,
    ...
}).on(STREAM_EVENT, function(streamEvent) {
    switch (streamEvent.type) {
        case STREAM_EVENT_TYPE.AUDIO_MUTED:
```

```
            $("#audioMuted").text(true);
            break;
        case STREAM_EVENT_TYPE.AUDIO_UNMUTED:
            $("#audioMuted").text(false);
            break;
        case STREAM_EVENT_TYPE.VIDEO_MUTED:
            $("#videoMuted").text(true);
            break;
        case STREAM_EVENT_TYPE.VIDEO_UNMUTED:
            $("#videoMuted").text(false);
            break;

    }
    console.log("Received streamEvent ", streamEvent.type);
});
```

Since WCS build 5.2.942 and WebSDK build 2.0.168 the event type `STREAM_EVENT_TYPE.DATA` is added, to receive JSON data sent by `stream.sendData()` or by `/stream/event/send` REST query

```
session.createStream({
    name: streamName,
    display: remoteVideo
    ...
}).on(STREAM_EVENT, function(streamEvent) {
    switch (streamEvent.type) {
        case STREAM_EVENT_TYPE.DATA:
            console.log(JSON.stringify(streamEvent.payload));
            break;
    }
}).play();
```

Receiving mixer incoming stream event

Since build 5.2.966, a subscriber playing mixer output stream can receive incoming mixer streams events. In this case, `streamName` field is added to `payload` objects to show what stream publisher generated the event

```
session.createStream({
    name: streamName,
    display: remoteVideo,
    ...
}).on(STREAM_EVENT, function(streamEvent) {
    let mutedName="";
    if(streamEvent.payload !== undefined) {
        mutedName=streamEvent.payload.streamName;
    }
    switch (streamEvent.type) {
        case STREAM_EVENT_TYPE.AUDIO_MUTED:
            $("#audioMuted").text(true + " " + mutedName);
            break;
        case STREAM_EVENT_TYPE.AUDIO_UNMUTED:
            $("#audioMuted").text(false + " " + mutedName);
```

```
            break;
        case STREAM_EVENT_TYPE.VIDEO_MUTED:
            $("#videoMuted").text(true + " " + mutedName);
            break;
        case STREAM_EVENT_TYPE.VIDEO_UNMUTED:
            $("#videoMuted").text(false + " " + mutedName);
            break;

    }
    console.log("Received streamEvent ", streamEvent.type);
}).play();
```

Stream status detection while subscriber is connecting to a stream

A new subscriber can detect if audio/video track is muted in the stream while connecting to the stream using `Stream.getAudioState()` and `Stream.getVideoState()` methods in `STREAM_STATUS.PLAYING` handler:

```
session.createStream({
    name: streamName,
    display: remoteVideo,
    ...
}).on(STREAM_STATUS.PLAYING, function (stream) {
    if (stream.getAudioState()) {
        $("#audioMuted").text(stream.getAudioState().muted);
    }
    if (stream.getVideoState()) {
        $("#videoMuted").text(stream.getVideoState().muted);
    }
    ...
}).play;
```

Mixer incoming streams status detection while connecting to a mixer outgoing stream

Since WCS build 5.2.1011, when a new subscriber connects to a mixer outgoing stream, it will receive a set of `STREAM_EVENT` events per every mixer incoming stream, if audio or video track was muted at least once in any of them. In this case, `STREAM_EVENT` receiving order is not guaranteed and does not depend on streams addition order to the mixer.

Stream event processing on backend

To process stream events on backend server, `sendStreamEvent` and `StreamEvent` methods should be added to the REST hook application

```
add app-rest-method MyAppKey sendStreamEvent
add app-rest-method MyAppKey StreamEvent
```

Audio/video mute/unmute notification

Backend server will receive `sendStreamEvent`, if publisher mutes/unmutes audio/video in the stream

```
URL:http://localhost:8081/apps/EchoApp/sendStreamEvent
OBJECT:
{
  "nodeId" : "qg4BeHzYSAtkhUkXgnSMEUZpsshaLPL5@192.168.0.39",
  "appKey" : "defaultApp",
  "sessionId" : "/192.168.0.83:64573/192.168.0.39:8443-a98bb891-aeaf-46a8-
8fba-772e07ac035b",
  "mediaSessionId" : "9906b2b0-9c28-11eb-8d20-75f877676678",
  "type" : "audioMuted",
  "origin" : "https://wcs:8888"
}
```

Also, backend will receive `StreamEvent` per every the stream subscriber

```
URL:http://localhost:8081/apps/EchoApp/StreamEvent
OBJECT:
{
  "nodeId" : "qg4BeHzYSAtkhUkXgnSMEUZpsshaLPL5@192.168.0.39",
  "appKey" : "defaultApp",
  "sessionId" : "/192.168.0.83:64573/192.168.0.39:8443-a98bb891-aeaf-46a8-
8fba-772e07ac035b",
  "mediaSessionId" : "9fed5c50-9c28-11eb-8d20-75f877676678",
  "type" : "audioMuted"
}
```

Stream event with data passing notification

If stream event with data is passed from publisher to all the stream subscribers, then backend will receive `sendStreamEvent` with payload

```
URL:http://localhost:8081/apps/EchoApp/sendStreamEvent
OBJECT:
{
  "nodeId" : "qg4BeHzYSAtkhUkXgnSMEUZpsshaLPL5@192.168.0.39",
  "appKey" : "defaultApp",
  "sessionId" : "/192.168.0.83:64573/192.168.0.39:8443-a98bb891-aeaf-46a8-
8fba-772e07ac035b",
  "mediaSessionId" : "9906b2b0-9c28-11eb-8d20-75f877676678",
  "type" : "data",
  "payload" : {
    "count" : 23
  },
```

```
  "origin" : "https://wcs:8888"
}
```

Also, backend server wiil receive `StreamEvent` per every the stream subscriber

```
URL:http://localhost:8081/apps/EchoApp/StreamEvent
OBJECT:
{
  "nodeId" : "qg4BeHzYSAtkhUkXgnSMEUZpsshaLPL5@192.168.0.39",
  "appKey" : "defaultApp",
  "sessionId" : "/192.168.0.83:64573/192.168.0.39:8443-a98bb891-aeaf-46a8-
8fba-772e07ac035b",
  "mediaSessionId" : "9fed5c50-9c28-11eb-8d20-75f877676678",
  "type" : "data",
  "payload" : {
    "count" : 23
  }
}
```