Web Real-Time Communications Developer's Guide

Grtc.MediaSession

5/1/2025

Contents

Grtc.MediaSession

This class creates a peer connection. Once a user is signed in to the WebRTC Gateway, he or she can both initiate and receive calls by using the `Grtc.MediaSession` class.

Instantiation

To create a new `Grtc.MediaSession` object, pass the `Grtc.Client` object as an argument.

Example

```
var grtcClient = new Grtc.Client(configuration);
var grtcSession = new Grtc.MediaSession(grtcClient);
```

Instance Methods

acceptCall ([audioConstraints] [,videoConstraints])

Accepts the incoming call, with the possibility of accepting only audio or video mode by setting an optional parameter. By default, it is assumed that both audio and video should be enabled.

Arguments

- **audioConstraints (optional)**—boolean value indicating whether audio is enabled or not; default value is `true`.
- **videoConstraints (optional)**—boolean value indicating whether video is enabled or not; default value is `true`.

Throws: INVALID_STATE_ERROR

If there is no incoming call, an exception is thrown.

Example

```
grtcClient.onIncomingCall.add(function (data) {
    // create a session
    var grtcSession = new Grtc.MediaSession(grtcClient);
    // inform user of incoming call
    var userResponse = window.confirm("Accept call from " + data.peer + "?");
    if (userResponse === true) {
        // accept the call
        grtcSession.acceptCall();
        // process attached data if available
        var dataAttached = grtcSession.getData();
        if (dataAttached) {
            // dataAttached is an array of objects, each of which
            // has 2 properties "key" and "value"
            for (var i=0; i<dataAttached.length; ++i) {
                var obj = dataAttached[i];
                console.log("dataAttached[" + i + "]:" + obj.key + ", " + obj.value);
            }
        }
    } else {
        grtcSession.rejectCall();
    }
});
```

closeSession (sendBye)

Closes the ongoing media session on the client side, and also sends BYE to the remote peer if requested. The method terminateCall() is now obsolete, and instead developers should use closeSession (true) as a substitute for terminateCall().

Arguments

sendBye(optional)—boolean value which, if set to `true`, sends a BYE request to the remote peer, as well as closing the current media session on the client side. The default value is `false`.

Example

```
grtcSession.closeSession(true);
```

getData ( )

Get the JSON object attached to a `Grtc.MediaSession`.

The data retrieved is a JavaScript array with the same format as the data array passed to `setData(data)`. This means that the data in the array can be accessed using a loop.

Throws: GRTC_ERROR

If the data cannot be retrieved, an exception is thrown.

Example

```
var dataAttached = grtcSession.getData();
if (dataAttached) {
    // dataAttached is an array of objects, each of which
    // has 2 properties "key" and "value"
    for (var i=0; i<dataAttached.length; ++i) {
        var obj = dataAttached[i];
        console.log("dataAttached[" + i + "]:" + obj.key + ", " + obj.value);
    }
}
```

makeCall (remoteId, [audioConstraints] [,videoConstraints])

This method has been deprecated and is replaced with the `makeOffer` method.

Initiates an audio or video call to another user. The user can also make audio-only or video-only calls by setting an optional parameter.

Arguments

- **remoteId**—the ID of the other user (string).
- **audioConstraints (optional)**—boolean value indicating whether audio is enabled or not; default value is `true`.
- **videoConstraints (optional)**—boolean value indicating whether video is enabled or not; default value is `true`.

Example

```
// create a new session (passing the client object as argument)
var grtcSession = new Grtc.MediaSession(grtcClient);
// set callback to port remote media stream when it comes
grtcSession.onRemoteStream.add(function (data) {
    grtcClient.setViewFromStream(document.getElementById("remoteView"), data.stream;
});
// attach data if available: the data is an array of objects
// each object contains two properties named “key” and “value”
var dataToAttach = [
    {
        "key": "Name",
        "value": $('#login_name').val()
    },
    {
        "key": "Phone",
        "value": $('#phone_number').val()
    },
    {
        "key": "Email",
        "value": $('#email').val()
    }
];
grtcSession.setData(dataToAttach);
// preparation is ready, now make the call
grtcSession.makeCall(“1021”);
```

makeOffer(remoteId, [audioConstraints [,videoConstraints [,holdMedia]]])

Initiates an audio or video call to a peer, or update an existing call. The media types used and the directions of the media in the call depends on the local media stream(s) enabled, as well as the constraints used in this function call. Note that these optional constraints can be set ahead of time using the `Grtc.Client` method `setMediaConstraintsForOffer()`.

Note: This function replaces `makeCall`.

Arguments

- **remoteId**—the ID of the other user (string). If the value is set to `self` when a new call is made, the call becomes a loopback call with the WebRTC Gateway, which may be useful for testing purposes. In this case, no interaction takes place with the SIP-side, and the media sent by the browser is sent back to it. Note that the audio/video constraints specified here for the loopback call has to match the local media stream, since uni-directional media such as one with the `recvonly` attribute would not make sense.
- **audioConstraints (optional)**—boolean value indicating whether audio is enabled or not; default value is `true`.
- **videoConstraints (optional)**—boolean value indicating whether video is enabled or not; default value is `true`.
- **holdMedia** (optional)—if `true`, sets the media lines in the SDP to inactive in order to put the call on hold.

rejectCall ( )

Reject the incoming call.

Throws: INVALID_STATE_ERROR

If there is no incoming call, an exception is thrown.

Example

See the `acceptCall ( [audioConstraints] [,videoConstraints] )` example above.

sendDTMF ( )

Sends one or more DTMF tones from among the following possible values:

- [0-9]
- *
- #
- A, B, C, D

Important
For Firefox, supported in version 53 and higher only.

Arguments

- **tones**—String composed of one or multiple valid DTMF symbols

- **options**—Available options, including:
  - **duration**—The duration of the DTMF tones sent by this method. The default is 100 ms. The duration setting cannot be greater than 6000 ms or less than 70 ms.
  - **tonegap**—The gap between tones. It must be at least 50 ms. The default value is 50 ms.

Returns

- **0**—Success
- **-1**—Failure

Example

```
// This function uses our sendDTMF API to send DTMF tones
// for an already-created mediaSession object
function sendTone(tones){
  var options = {"duration": 500, "tonegap": 50};
  console.log("DTMF send result: [" + mediaSession.sendDTMF(tones, options) + "]");
}

// This function creates a simple DTMF Dial Pad for the HTML page
function createDialingPad() {
  var tones = '1234567890*#ABCD';
  var dialingPad = document.getElementById('dialingPad');
  for (var i = 0; i < tones.length; ++i) {
    var tone = tones.charAt(i);
    dialingPad.innerHTML += '<button id="' +
      tone + '" onclick="sendTone(\'' + tone +
      '\')" style="height:40px; width: 30px">' + tone + '</button>';
    if ((i + 1) % 4 == 0) {
      dialingPad.innerHTML += '<br>';
    }
  }
}
```

sendInfo (data)

Sends a mid-call ROAP INFO message to the WebRTC Gateway with the specified input data. The gateway, in turn, sends a SIP INFO message to the SIP Server with the data in the body of the message. The input data should be a simple object. A serialized representation of the data is created in URL query string format before the data is sent to the Gateway.

Arguments

- **data**—A simple object
- **mapData**—boolean value. When `mapData` is set to `true` or is undefined, the content-type of the SIP INFO message (from the Gateway to the SIP Server) is set to `application/x-www-form-urlencoded`. In this case, the SIP Server consumes the data and maps it to the corresponding T-Library events. When `mapData` is set to `false`, the content-type for the SIP INFO message is set to `application/octet-stream` and SIP Server simply passes the message to the remote end, which is another WebRTC or SIP EndPoint.

Throws: GRTC_ERROR

If the data is not well-formed, an exception is thrown.

Example

```
var data = {};
data.visitid = 'abc123';
data.token = '1234';
grtcSession.sendInfo(data, true);
```

setData (data)

Set a JSON object as the data to be attached to a `Grtc.MediaSession`.

Arguments

data—A well-formed JavaScript array of one or more objects, in which each object contains exactly two properties: key and `value`, where key is the identifier of the piece of data to be attached, and `value` is the value of the piece of data.

Throws: GRTC_ERROR

If the data is not well formed or cannot be attached, an exception is thrown.

Example

```
var dataToAttach = [
    {
        "key": "Name",
        "value": “My Name”
    },
    {
        "key": "Phone",
        "value": “800-555-1212”
    },
    {
        "key": "Email",
        "value": “my.address@somewhere.com”
    }
];
grtcSession.setData(dataToAttach);
```

terminateCall ( )

Terminates an existing media session.

Throws: INVALID_STATE_ERROR

If there is no active call, an exception is thrown.

Example

```
grtcSession.terminateCall();
```

updateCall (audioEnabled ,videoEnabled)

Updates the call in progress, muting or unmuting audio and/or video track, by setting the corresponding enabled flags.

Arguments

- **audioEnabled**—boolean value indicating whether audio is enabled; if not boolean or not specified, it will be ignored.
- **videoEnabled**—boolean value indicating whether video is enabled; if not boolean or not specified, it will be ignored.

Throws: INVALID_STATE_ERROR

If there is no active call, an exception is thrown.

holdCall()

Puts an active call on hold.

resumeCall()

Resumes a call from hold.

getServerStats()

Makes a request to the WebRTC Gateway for call statistics. When the statistics are received from the Gateway, JSAPI triggers the `Grtc.Client onStatsFromServer` event along with the data.

hasAudioEnabled()

Determines whether a call has audio enabled after an SDP negotiation with the peer.

Returns

Boolean (true/false)

hasVideoEnabled()

Determines whether a call has video enabled after an SDP negotiation with the peer.

Returns

Boolean (true/false)

isEstablished()

Returns true if a call session is established, and no renegotiation is occurring.

Events

onRemoteStream

This event is triggered when a remote stream is added to the peer connection. The user is expected to handle this event in order to deal with the received stream, for example, to embed the stream in an HTML5 <video> element.

Event data parameters

- stream—the remote stream

onSessionHold

This event is triggered when an active call goes into hold/inactive status. The application can use this event to update the call status. Note that, when the call becomes active again, the `onRemoteStream` event will be triggered.

Event data parameters

- isTrue—Set to true when the call is on hold. Currently, this event is not triggered with a value of "false" when the call becomes active again, although this can change in the future.