SocialVid

Video Platform API

Document Version 1.0

Democratizing Video Communication for Everyone to use

Table of Contents

Introduction

SocialVid Platform API is the interface for third-party web application developers to build or include custom Video Communication client applications that seamlessly integrate into the Web infrastructure.
Platform APIs allow video rendering using the webRTC engine built into popular browsers and control using JavaScript.
Rich set of Platform APIs, manage the overall configuration and control of the pre-call and in-call state machine and give programmers an easy and quick access to Video Communications and all necessary controls.
This documents attempts to list entire APIs, and also includes a sample application.

API Work Flow:

SocialVid Platform APIs are implemented in the WebRtcClient object. The WebRtcClient exposes interfaces to make peer to peer or multiparty audio/video calls to a conference.
It also has interfaces to enable desktop/application sharing, chat, file transfer, Conference Recording and whiteboard.

Developers using plain javascript will implement the code similar to the following code, and include in the main application.

1. Create a WebRtcClient object. It takes 2 arguments, the server to connect to and a callback in which events are posted.

```
var server = “demo.socialvid.in”;

var callback = function(msg) {
};

var client = new WebRtcClient(server, callback);
```

2. Login using an email id and password. The callback will provide the login status and configuration parameters of the server. A login status of 0 indicates that login was successful.

```
var email = user1@socialvid.in;
var password = “user1ispassword”; // For simplicity we shall go  with unsecure way

var loginCallback = function(loginResponse) {
        if (loginResponse.status === 0) {
        } else {
        }
};
```

```
client.login(email, password, loginCallback);
```

3. On successful login, fetch the contacts and conferences from the server.

```
var contactsCallback = function(contacts) {
        for (var j = 0; j < contacts.length; j++) {
                console.log(contacts[i].type);
                console.log(contacts[i].id);

        }
};
client.getContacts(contactsCallback);
```

4. Any conference can be joined. In the callback, the remoteStreams will be received along with active talker indication. The remote streams can be attached to the video elements and they can be shown based on the active talker indication.

In the example below, only one remote party is shown.

```
var callCallback = function(msg) {
        var element = document.getElementById("remoteVideo");
        switch(msg.type) {
                case "remoteStream":
                        if (msg.index === 1) {
                                attachMediaStream(element, msg.stream);
                        }
                        break;

                case "activeTalkerList":
                        element.style.display = "block";
                        break;
        }
};

client.joinVideoConference(confId, callCallback);
```

5. Disconnect the call.

```
client.disconnectCall();
element.style.display = "none";
```

List of APIs and its brief description:

WebRtcClient(server, callback)

- This is the constructor of the client object.
- Input
  - Server – the webrtc server to connect to
  - Callback – a callback that receives the following events
    - voiceCallRequest
    - videoCallRequest
    - contactUpdated
    - conferenceUpdated
    - userJoined
    - userLeft

login(email, passwd, cb)

- Login using email and password. The login callback cb will be invoked with the login status and configuration parameters.
- The user must be a provisioned and registered user

❖ Input:
  - Email – Email-Id of registered user.
  - Passwd : Password of the user

➢ Returns 0 is login is successful else check status field for the error status
  - userNotValid
  - licenseNotAvailable
  - licenseInvalid
  - licenseExpired

➢ Cb – provide a callback to handle the response

guestLogin(guestName, conferenceId, cb)

- Guest login with name 'guestName' for joining a conference with conference id 'conferenceId'. The login callback cb will be invoked the guest login status.
- No pre-registration is required to join the call/conference if he is a guest user.
- Is the user is a guest user, he will NOT be able to Record the conference

❖ Input:
  - guestName – Name of guest user.
  - ConferenceId – Enumerated conferenceId from guest link from getConferences

➢ Returns 0 is login is successful else an –ve error code

➢ Cb – provide a callback to handle the response
  - conferenceNotValid

- licenseNotAvailable
- licenseInvalid
- licenseExpired

getContacts(cb)

- Fetch all the contacts with their status. Contacts includes both users and conferences.
- This API is only applicable to Registered a.k.a. Logged-in user

➢ Returns Array of contacts/buddies in the same organization

Call APIs

makeVoiceCall(calleeId, cb) / makeVideoCall(calleeId, cb)

- Make voice or video call to user with userid as calleeId.
- makeVoiceCall/makeVideoCall creates a peer to peer call. So the media flows end-to-end.
- Recording, Call Transfer and Advanced Bandwidth management features will NOT be available for these calls

❖ Input:
- calleeId is obtained from enumeration getContacts function for a specific user in contact list

➢ The function provides following callbacks to developers to implement:
- userUnavailableResponse
- callProceeding
- rejectCallResponse
- acceptCallResponse
- remoteStream
- localStream
- userFilesTransferRequest
- userChatMessage
- remoteShareStream
- stopShare
- userAudioMuted
- userAudioUnmuted
- userVideoMuted
- userVideoUnmuted
- disconnectCallRequest
- userCaption

answerCall(callType, cb) / rejectCall() / ignoreCall()

- Answer, reject or ignore incoming call. The incoming call notification is given in the main callback which is registered when the WebRtcClient object is created.
- callType can be “voice” or “video”

disconnectCall(cb)

- Disconnect the current call.
- The callback is called on completion of the call disconnect

Conference APIs

addConference (conferenceObject)

- Adds a conference for a user using attributes specified in the conference object defined as input
- Input:
    - id: <some unique id>
        - -- Unique identifier to create identifier
    - name: <name of the conference>,
        - Reference name of conference for display purpose
    - autoRecord: true|false
        - Want auto recording of the created conference
    - maxBitrateKbps: "256|512|768|1024|auto"
        - Bitrate to be used per participant

modifyConference(conference object)

- The conference object with the matching id will be modified as new parameter defined above.

deleteConference(conference object)

- the conference object with the matching id will be deleted

joinVoiceConference(confIid, cb) / joinVideoConference(confId, cb)

- Make voice or video call to conference with conference id as 'id'.
- Advanced features like Recording, Call Transfer and Advanced Bandwidth management will be available.

❖ *Input:*
    - confId is obtained from enumeration getContacts function for a specific conference in contact list

➢ The conference callback cb will receive all the conferece related messages as following
    - callProceeding
    - videoConfResponse
    - licenseStatus
    - getUserMediaFailed
    - remoteAudioStream
    - remoteStream

- localStream
- remoteShareStream
- confFilesTransferRequest
- confChatMessage
- remoteShareStream
- confAudioMuted
- confAudioUnmuted
- confVideoMuted
- confVideoUnmuted
- confCaption
- activeTalkerList
- recordingStarted
- recordingStopped
- participantsUpdated

Recording APIs

startRecording (constraints)

- To start recording in a conference. Only allowed if you are the owner of the conference.

❖ *Input:*

constraints

```
{
        recordAudio : true/false,
        recordVideo: 0 to n indicating number of video participants to be recorded,
        recordShare: true/false
}
```

stopRecording(cb)

- To stop recording in a conference.
- cb – Callback is called After the action is completed.

getRecordings

- Will return the 10 most recent recordings.

```
Each recording object has the following
{
  userId: <unique user id>,
  userName: <the username>,
  confId: <the conference id>,
  confName: <the conference name>,
  path : <the path where the recording is stored>,
  startTime: <time in unix format when the recording was started>
}
```

- The recording updated event is called in the WebRtcClient callback. When the WebRtcClient object is created, a callback is passed to it. The 'recordingsUpdated' event will be sent to this callback.

```
{
    type: 'recordingsUpdated'
    }
```

On receiving this event, WebRtcClient.getRecordings can be invoked to get the recordings.

deleteRecording(recording.path)

- Will delete the recording, specified in the path returned from getRecording object

Feature APIs

uploadFiles(files, cb)

- Upload files in a call or conference

downloadFile(fileName)

- Download files in a call or conference. The indication that files are available are given in the call callback through userFilesTransferRequest.

sendChat(txt)

- Send chat message to a user or to all participants in the conference. The remote end will receive chat notification in the call / conference callback.
- Private chat in conference is not yet supported

startShare(cb)

- Start an application or screen share in call or conference

stopShare(cb)

- Stop the ongoing share.

muteVideo() / muteMicrophone() / muteSpeaker()

- Mute video / microphone / speaker.

unmuteVideo() / unmuteMicrophone() / unmuteSpeaker()

- Unmute video / microphone / speaker.

startCaptions()

- Start speech to text conversion and send the text to the remote end. The remote end will receive the text in the call / conference callback.

stopCaptions()

- Stop the speech to text conversion

setCanvases(c1, c2)

- Set HTML canvas elements on the client object for displaying whiteboard.

resizeCanvas(w, h)

- Called when the browser window is resized and the canvas needs to be refreshed.

setWhiteboardShape(shape, color, width)

- Sets the whiteboard drawing pattern. Shape can be line, pencil, rectangle, circle or eraser. The color can be RGB value of the color to be used for drawing. And the width can be of any value from 1 to 5.