ABTO Software
WebRTC VoIP SIP SDK Manual

I WORKFLOW

Simple video sample implementation.

1. SDK consists of 2 javascript files - ABTOPhoneUA.min.js and sip.min.js, so include it to your html page

```
<!DOCTYPE html>
<html>

    <head>
        <script src="<path>sip.min.js" type="text/javascript"></script>
        <script src="<path>ABTOPhoneUA.min.js" type="text/javascript"></script>

        ...

    </head>

    <body>

        ...

    </body>

</html>
```

2. In body section create 2 video objects to display incoming and outgoing video streams and two buttons to start a call and to end the call:

```
<body>
    <video id="video_remote" autoplay="autoplay" style="width:480px;
height:320px;"></video>
    <video id="video_local" muted autoplay="autoplay" style="width:180px;
height:120px;"></video>

    <br/>
    <input type="button" id="btnCall" value="Call"/>
    <input type="button" id="btnHangup" value="Hang Up"/>
</body>
```

3. In head section create a script node. All further logic will be implemented there.

```
<script type="text/javascript">

...

</script>
```

4. Inside the script node create a function createPhone and set it as onload listener and a function freePhone and set it as onbeforeunload listener. The former should be responsible for creating, initializing and registering a phone object, the former one - for unregistering. Also declare a global variable which will hold the phone object and another variable for current call id.

```
var ABTOPhone = null;
var callID = -1;

window.onload = createPhone;
window.onbeforeunload = freePhone;

function createPhone(){

}

function freePhone() {

}
```

5. Inside the createPhone function add the code that initializes the phone and registers a user.

a. create ABTOPhoneUA instance

```
ABTOPhone = new ABTOPhoneUA();
```

b. set listeners for phone's events you need

```
ABTOPhone.onConnected = function() { };

ABTOPhone.onDisconnected = function() { };

...

ABTOPhone.onRecordReady = function(id, url) { };
```

You don't need to implement all of them but you definitely need onInvited event which fires when an incoming call arrives.

```
ABTOPhone.onInvited = function(id, from) {
   if (confirm('Accept call from ' + from)) {
       callID = id;
       ABTOPhone.setRemoteMedia(id, document.getElementById("video_remote"));
       ABTOPhone.accept(id);
   }
   else {
       ABTOPhone.reject(id);
   }
};
```

This example of onInvited event implementation is not practically useful, as it cannot be canceled and simultaneous incoming calls cannot be processed. But it is very simple and shows the usage of accept and reject methods.
ABTOPhone sdk is a multicall system and can handle multiple calls simultaneously, however his simple sample stores only a single current call id.
In case a call is accepted, callID is used to bind a video object to this call using setRemoteMedia function. In audio only implementations there may be an audio object instead of video.

c. add listeners for call and hangup buttons

```
var btnCall = document.getElementById("btnCall");
btnCall.onclick=function(){
   callID = ABTOPhone.call('101@html5sdk.abtollc.com');
   if (callID >= 0  ) {
       ABTOPhone.setRemoteMedia(callID, document.getElementById("video_remote"));
   }
};
```

Function call creates a call, sends INVITE to the other party and returns callID. You use callID to bind the video control with id video_remote to this call. When callee accepts the invite, his video will be shown there.

```
var btnHangUp = document.getElementById("btnHangup");
btnHangUp.onclick=function(){
   ABTOPhone.bye(callID);
};
```

Function bye ends the call with id callID.

d. get access to hardware

```
ABTOPhone.startLocalMedia(document.getElementById("video_local"), null, null);
```

In this example browser will try to get access to both camera and microphone. Camera output instantly will be played on the video object with id video_local. This method returns promise.

e. set required account fields and register the account

```
ABTOPhone.setSipDomain('html5sdk.abtollc.com');
ABTOPhone.setWSPort('15063');
ABTOPhone.setSipUserName('100');
ABTOPhone.setSipLogin('100');
ABTOPhone.setSipPassword('100');
ABTOPhone.initAndRegister();
```

initAndRegister function will first connect to a server and instantly start the registration process. After successful registration application fires onRegistered event and from now is able to place and accept calls.

6. Inside freePhone function add the code that unregisters the account and closes connection

```
if(ABTOPhone)
   ABTOPhone.close();
```

This is needed not to leave parasite registrations after application is closed.

II SETTERS AND GETTERS

setters

clearStunServers = function()
removes all STUN servers and disables STUN

addStunServer = function(url)
url – string
adds a server to the list of STUN servers and enables STUN

clearTurnServer = function()
removes active TURN server and disables TURN

setTurnServer = function(url, username, credential)
url – string
username – string
credential – string
sets a TURN server and enables TURN

*setRemoteMedia = function(*callId*,remoteMedia)*
callId - integer; *remoteMedia* - Audio or Video object
associates Audio or Video object with a remote party.

setRecord = function(doRecord)
doRecord - boolean
defines whether to perform remote party's media (audio and/or video)
default value - false

setSipDomain = function(sipDomain)
sipDomain - string
defines the address of a sip server

setSipProxy = function(sipProxy)
sipProxy - string
defines the address of a proxy server. Optional – if not set, sipDomain is used instead.

setWSPort = function(WSPort)
WSPort - integer or string representing integer
defines websocket port of the sip server

setSecure = function(secure)
secure - boolean
defines whether websocket connection is performed via ws or wss
default value - true (wss)

setSipDisplayName = function(sipDisplayName)
sipDisplayName - string
optional element of sip account. Is used in FROM SIP header.

setSipUserName = function(sipUserName)
sipUserName - string
element of sip account. Is used in FROM SIP header.

setSipLogin = function(sipLogin)
sipLogin - string
element of sip account. Is used to authenticate a user.

setSipPassword = function(sipPassword)
sipPassword - string
element of sip account. Is used to authenticate a user.

setRegisterExpire = function(regExpire)
regExpire - integer or string representing integer
sets registration validity duration. SDK will automatically prolong registration before this timeout. If not set, the default value is used.

getters

getIsConference = function()
return value - boolean value which shows whether current calls are organized in a single conference

getToken = function()
return value - sip registration token (string)

getIsRegistered = function()
return value - boolean value which shows whether user is already registered with sip server

getIsLocalMediaStarted = function()
return value - boolean value which shows whether startLocalMedia was successful and local media stream is running

getActiveCallsCount= function()
return value - number of active calls

III EVENTS

connection and registration events

onConnected = function()
fires when connection to a sip server via websocket is established

onDisconnected = function()
fires when websocket connection is closed

onConnectionError = function(error)
fires when connection to a sip server via websocket was not successful. gives error string as a parameter

onRegistered = function()
fires when registration with sip credentials is successful

onUnregistered = function()
fires when unregistration is successful

onRegisterError = function(code,status)
fires when registration with sip credentials is not successful. gives error code and status text

onUnregisterError = function(code,status)
fires when unregistration was not successful/ gives error code and status text

onReconnected = function()
fires when registration was reestablished after connection loss

sip messaging event

onMessage = function(from, text)
fires when a registered user gets SIP message MESSAGE. gives the sender address as ‘from’ and message text.

media events

onRemoteMediaStarted = function(callId,stream)
fires when a media stream (stream parameter) of a remote party starts playing. callId parameter shows which remote party is it. Though this event gives a media stream, a developer should not bother handling it.

onRemoteMediaStopped = function(callId)
fires when a media stream of a remote party stops playing. callId parameter shows which remote party is it.

onLocalMediaStarted = function(stream)
fires when a browser has gained access to media resources (audio, video or both). Though this event gives a media stream, the developer should not bother handling it.

onLocalMediaStartFailed = function(error)
fires when a browser has failed to access media resources. Error is an object returned by getUserMedia function.

onLocalMediaStopped = function()
fires when the local media stream stops playing.

call events

All events in this group take place within a call. ABTO SDK supports multiple simultaneous calls so all events have a common parameter callId, which shows to which call belongs this event.

onInvited = function(callId,from)
fires when a call arrives from a remote party. 'from' parameter shows the telephone number of the remote party.

onRinging = function(callId,statusCode)
Can fire in case of outgoing call. This event fires when a SIP message with statusCode < 200 arrives, usually signaling that the call is placed on the remote party but the interlocutor hasn't accepted it yet (i.e. callee party is 'ringing').

onRingingTransfer = function(callId,to)
fires in the process of call transfer. Shows that the previous call is cleared and placing the call to the next interlocutor is in process (i.e. a transferee party is 'ringing'). The developer may use it to prepare the user interface as in case of an outgoing call. 'To' parameter shows transferee telephone number.

onEstablished = function(callId, from)
is fired when the call is successfully established after a callee accepts the call.

onEstablishError = function(callId,code,status)
fires when an error occurs in the process of establishing the call. Code is error code and status is reason phrase.

onHangUp = function(callId)
fires in those cases:
when a remote party rejects the call
when caller cancels his outgoing call
when the call ends normally

onCallCleared = function(callId)
fires when the call ends, no matter if the call ends normally, is cancelled or as a result of an error. In almost all cases onEstablishError and onHangUp are followed by onCallCleared.

onHold= function(callId, holdON, thisSideInitiated)
fires when one of the parties initiates HOLD or releases HOLD. holdON is true when HOLD gets initiated and is false otherwise (released). thisSideInitiated is true when HOLD is initiated by the local side, and is false if HOLD is initiated by an interlocutor.

onRecordReady = function(callId, bloburl)
fires when media stream recording is finished and record is ready. The recorded stuff is stored in memory and bloburl is a link to it. The developer may download it to a file, start playback or upload it to a storage. Recording shall not start if *setRecord* was called with a '*false'* parameter or if it was not called at all.

IV METHODS

function ABTOPhoneUA(userAgent)
This is the constructor of phone class. userAgent is an optional string parameter which sets a user agent for this object.

init = function()
Initializes phone object and starts connection to a SIP server via websocket. Connection parameters - address and port - must be set before this call by setters setSipDomain and setWSPort respectively. Depending on a result it will fire either onConnected or onConnectionError.

uninit = function()
closes websocket connection established by init().

register =function()
establishes SIP registration using credentials previously set by setters setSipUserName, setSipLogin, setSipPassword. Depending on a result it will fire either onRegistered or onRegisterError.

unRegister =function()
removes current registration. Depending on a result it will fire either onUnregistered or onUnregisterError.
.
initAndRegister =function()
this function first calls init() and if connection is established successfully it calls register().

reset = function()
resets inner state without reconnection

reconnect = function()
performs reconnection after a random timeout

recover = function(callId)
if a call is in progress, it finishes it and sends invite anew

startLocalMedia = function (localVideo, constraints, _stream)
returns promise that initiates access to microphone and camera. localVideo is an HTML video object. If it is null audio only access is assumed. The latter logic may be overridden by constraints. Constraints is an optional parameter that defines what kind of resource to get access to (microphone, camera or both). Function uses WebRTC getUserMedia to get media resources with the constraints passed as a parameter. If _stream parameter is not null startLocalMedia uses this stream instead, so constraints are ignored. Constraints may also be used to select one of microphones or one of cameras if there are several of them.

stopLocalMedia = function ()
stopps media stream previously accessed in startLocalMedia

call = function(to, addThisCaleeToConference, customHeaders)
initiates a new call with callee “to” by sending an INVITE message. Optionally the INVITE message may contain custom headers. If addThisCaleeToConference is true the callee will join a conference. SDK supports a single conference at a time, the number of participants is not limited. If there is an active conference session, false value of addThisCaleeToConference will be ignored. Conference participants may invite others, however it is impossible to join a conference from outside. Conferencing (many-to-many calls) is possible within ABTO WebRTC SIP SDK only – conference calls with other types of clients will be established as usual calls (as one-to-many).

hold = function(callId)
initiates call hold. If a call is on hold already, resumes the call

mute = function()
mutes microphone

unmute = function()
unmutes muted microphone

transfer = function(callId, transferTo)
initializes call transfer. In the process of transfer the current call gets disconnected and a new call to a third party gets placed.

sendDTMF = function(callId, DTMFCode)
sends sip DTMF code within established call

sendMessage =function(messageTo, text)
sends sip MESSAGE to a recipient registered as “messageTo” with content as “text”.

bye = function(callId)
ends a call; amid conference bye on any of active calls triggers bye on all calls

accept = function(callId)
accepts an incoming call

reject = function(callId)
rejects an incoming call

hangupAll = function()
hangs up all ongoing calls

close = function()
this function is opposite to initAndRegister - it hangs up all ongoing calls, removes registration and closes websocket connection.