Real-time Communication
Web-based Integration
Product Introduction

Contents

Web-based Integration Sample Code

Last updated：2018-09-28 16:42:44

Sample Codes

We provide some sample codes for different features to help you understand WebRTC.
For more information on APIs, see API Documentation.

Sample codes

- Detect whether WebRTC is supported
- LVB and Joint Broadcasting
- Viewer mode (non-push)

Detect whether WebRTC is supported

Whether WebRTC is supported

```
WebRTCAPI.fn.detectRTC( function(info ){
if( !info.support){
alert('Not SupportedWebRTC')
}
});
```

LVB and Joint Broadcasting

There are two ways of "LVB + Joint Broadcasting"

1. After a user joins a room, the camera and microphone data is collected automatically for joint broadcasting.

2. After a user joins a room, LVB video is played without push, and the push API is called at the right time for push.
   Choose either way based on your business needs, and two APIs are needed.
   The main difference is the default setting of closeLocalMedia in the initialization parameters.

   Automatic push

var RTC = new WebRTCAPI({

```
userId: userId,
sdkAppId: sdkappid,
accountType: accountType,
userSig: userSig
```

},function(){

```
//Callback of successful initialization

//Join a room
RTC.createRoom({
roomid : 12345,
role : "user",
privateMapKey: privateMapKey,
},function(error){
console.error( error )
} );
```

},function(error){

```
console.error( error )
```

});
Manual push

var RTC = new WebRTCAPI({

```
userId: userId,
sdkAppId: sdkappid,
accountType: accountType,
userSig: userSig,
closeLocalMedia: true
```

},function(){

```
//Callback of successful initialization

//Join a room
RTC.createRoom({
roomid : 12345,
role : "user",
privateMapKey: privateMapKey,
}, function(){
//Room created successfully

//Manual push
RTC.startRTC();
},function(error){
console.error( error )
} );
```

} ,function(error){

```
console.error( error )
```

});
Viewer mode (non-push)

var RTC = new WebRTCAPI({

```
"userId": userId,
"sdkAppId": sdkappid,
"accountType": accountType,
"userSig": userSig,
"privateMapKey": privateMapKey,
"closeLocalMedia": true
```

},function(){

```
//Join a room
RTC.createRoom({
roomid : 12345,
role : "user"
});
```

},function(error){

```
console.error( error )
```

});

Integrate SDK

Last updated：2018-10-08 17:54:52

Preliminary Information

Supported platforms for H5

OS Platform	Browser /webview	Required Version	Notes
iOS	Safari (Only)	11.1.2	
Android	Chrome	60+	Support for H264
Mac	Chrome	47+	
Mac	Safari	11+	
Windows(PC)	Chrome	52+	

Glossary

Tencent Cloud account information

Term	Meaning
appid	After a vendor is registered with Tencent Cloud, a unique ID (appId) on Tencent Cloud is generated.
sdkappid	Vendors can create multiple projects on the TRTC console to correspond to their own Apps. Each project is represented by a SdkAppid.
accounttype	Each SdkAppid has an AccountType parameter in the account system page, which is used during login.

"appId" is in the format of 125xxxxxxx and can be obtained at the top of the TRTC console.
"SdkAppid" is in the format of 14000xxxxx. It can be viewed in **App Basic Settings** after you create a project on the TRTC console.
You can see AccountType in **App Basic Settings** -> **Account Integration System** after creating a project on the TRTC console (if not, edit and save it)

User information

Term	Meaning
userId	Identifies users in the App, also known as user name.
userSig	Identity signature, which functions as a login password. Each userId has a corresponding signature with a certain validity period, which is added in the request to help Tencent Cloud identify users.

Video call information

Term	Meaning
Spear role	The name of a collection of a video user's configurations, such as resolution, bitrate, and frame rate, which can be maintained on the Spear engine page of the project.
roomId	Identifies a video chat room. Users with the same roomId can see each other.
privateMapKey	Room permission key, which functions as the key for joining a room with the specified roomId

Download sign_src.zip to obtain the calculation code for the server to issue UserSig and privateMapKey (the signature algorithm for generating UserSig and privateMapKey is ECDSA-SHA256).

Integration Preparations

1. Register a Tencent Cloud account and contact the Sales to activate the TRTC business.
2. Create a new project on the TRTC page. Get SdkAppid and AccountType.
3. See UserSig Calculation Documentation to calculate the UserSig of the test user name.
4. If you have configured a firewall for your network, make sure you open the following ports:

Protocol	Port
TCP	8687
UDP	8000, 8800 and 443

Introduce SDK using CDN.

Introduce WebRTCAPI.min.js on the page

```
<script src="https://sqimg.qq.com/expert_qq/webrtc/2.6/WebRTCAPI.min.js"></script>
```

Update Log

2.6

Added the SoundMeter API

Added a field for reporting log

Adjusted the initialization API

API Overview

Last updated：2018-09-28 16:50:12

TRTC Web APIs include the feature APIs for active calling and the event notification APIs with trigger setting, wherein the feature APIs include basic feature APIs and advanced feature APIs, and the event notifications include basic event notifications and advanced event notifications. Junior developers can complete basic development access through basic feature APIs and basic event notifications, and can experience the main features of TRTC. Senior developers can complete the development through advanced feature APIs and advanced event notifications, and can experience the advanced features of TRTC.

Basic feature APIs

API	Description
WebRTCAPI.fn.detectRTC	Detects whether it supports WebRTC
WebRTCAPI	Initialization
WebRTCAPI.enterRoom(WebRTCAPI.createRoom)	Creates or enters an audio/video room
WebRTCAPI.quit	Exits an audio/video room

Basic event notifications

Event	Description
onLocalStreamAdd	Addition/update of local video stream
onRemoteStreamUpdate	Addition/update of remote video stream
onRemoteStreamRemove	Disconnection of remote video stream
onWebSocketClose	Disconnection of websocket
onRelayTimeout	Disconnection of video stream server after timeout
onKickout	Forced logout (The same user logged in repeatedly)

Advanced feature APIs

API	Description
WebRTCAPI.startRTC	Only used for active push
WebRTCAPI.stopRtc	Stops the push
WebRTCAPI.getLocalStream	Gets local audio/video stream
WebRTCAPI.updateStream	Updates video stream
WebRTCAPI.openVideo	Turns on video collection again during the push process
WebRTCAPI.closeVideo	Temporarily turns off video collection during the push process
WebRTCAPI.openAudio	Turns on audio collection again during the push process
WebRTCAPI.closeAudio	Temporarily turns off audio collection during the push process
WebRTCAPI.changeSpearRole	Switches audio and video parameter settings
WebRTCAPI.getVideoDevices	Enumerates video collection devices
WebRTCAPI.getAudioDevices	Enumerates audio collection devices
WebRTCAPI.chooseVideoDevice	Chooses the video collection device
WebRTCAPI.chooseAudioDevice	Chooses the audio collection device
WebRTCAPI.SoundMeter	Sound input detection

Advanced event notifications

Event	Description
onPeerConnectionAdd	Notification of PeerConnection addition. Please make sure you understand the role and significance of peer connection notification

Update log

2.6.1

```
## WebRTCAPI.getSpeakerDevices
This API is used to enumerate audio output devices
## WebRTCAPI.chooseSpeakerDevice
```

This API is used to enumerate audio output devices

Basic Features

Last updated：2018-10-08 17:54:26

WebRTCAPI.fn.detectRTC

This API is used to detect whether WebRTC is supported.
Syntax example:
`WebRTCAPI.fn.detectRTC(options, callback);`

Parameter description:

Parameter	Type	Description
options	object	Parameter

options:

Parameter	Type	Description
screenshare	boolean	Whether to perform screen sharing detection. Defaults to true.

Callback's info field:

Field	Description	Note
support	Whether WebRTC is supported	
h264Support	Whether H.264 is supported	H.264 must be supported
screenshare	Whether screen sharing is supported	Plug-ins must be installed

Actual example:

```
WebRTCAPI.fn.detectRTC({
screenshare : false
}, function(info){
if( !info.support ) {
alert('WebRTC is not supported')
}
});
```

WebRTCAPI

This API is used to initialize WebRTC.

Syntax example:

```
var RTC = new WebRTCAPI( options , succ , error)
```

Parameter description:

Parameter	Type	Description
options	object	Parameter
succ	function	Callback successful
error	function	Callback failed

Options description:

Parameter	Type	Description	Note
sdkAppId	integer	sdkappid of application (if there is any doubt, see Integrating the SDK)	Required
accountType	integer	Account type (if there is any doubt, see Integrating the SDK)	Required
userId	string	The unique ID of the user, which is the user name we often say (if there is any doubt, see Integrating the SDK)	Required
userSig	string	Required. Identity signature, equivalent to the role of login password (if there is any doubt, see Integrating the SDK)	Required
closeLocalMedia	boolean	Whether to disable the automatic push (if set to true, the push at the local end will not be initiated after the entering/creating operation is completed; if the push is required, the push API needs to be actively called by the business)	Optional. Defaults to false.
audio	boolean	Whether to enable audio collection	Optional. Defaults to true.

Parameter	Type	Description	Note
video	boolean	Whether to enable video collection	Optional. Defaults to true.
debug	object	Debug mode (Prints the log on the console) {log:true, uploadLog:true, vconsole:true}	Optional
peerAddNotify	boolean	p2p connection notification. The business side determines whether a connection is required before establishing a p2p connection. Need to be used with [onPeerConnectionAdd] of [Advanced Event Notifications]	Optional. Defaults to false.

Actual example:

```
var RTC = new WebRTCAPI( {
"userId": userId,
"sdkAppId": sdkappid,
"accountType": accountType,
"userSig": userSig
} );

// Debug mode
var RTC = new WebRTCAPI( {
"userId": userId,
"sdkAppId": sdkappid,
"accountType": accountType,
"userSig": userSig,
"debug":{
"log": true, //Whether to print the debug log on the console. It is false by default
"vconsole": true //Whether the vconsole is shown (to facilitate viewing logs on the mobile end)
"uploadLog": true //Whether to report the log
}
} );
```

WebRTCAPI.enterRoom (WebRTCAPI.createRoom)

This API is used to create or enter an audio/video room.

Syntax example

```
var RTC = new WebRTCAPI( ... )
...
//Note: This must be called in the successful initialized callback of WebRTCAPI
RTC.enterRoom( options, succ , error );
```

Parameter description:

Parameter	Type	Required	Description
options	object	Yes	Parameter
succ	function	No	Callback successful
error	function	No	Callback failed

Options description:

Parameter	Type	Required	Description
roomid	integer	Yes	Room ID
privateMapKey	string	No	It refers to the key to enter the room with the specified roomID. If the business believes that there is no need to restrict the permission of users, you can leave it unset (if there is any doubt, see Integrating the SDK)
role	string	Yes	Switches the user role in the Screen Setting Console - SPEAR engine configuration
pureAudioPushMod	integer	No	Audio-only push mode. You need to use this parameter when you need to perform non-interactive broadcasting and recording 1 => This is an audio-only push, so there is no need to record mp3 files 2 => This is an audio-only push, and the recording file is mp3
recordId	integer	No	Custom business ID at automatic recording, int32, which is assigned to users at recording callback

Actual example:

```
var RTC = new WebRTCAPI({
"userId": "username",
"sdkAppId": 1400012345,
"accountType": 12345,
"userSig": "xxxxxxxxxxxxxxxxxxxxxxxxxxx",
}, function(data){
//Initialized successfully
RTC.createRoom( {
roomid : 123456,
role : "user",
}, function(){
//Entered the room successfully
} , function(data){
//Failed to enter the room
} );
}, function(data){
//Initialization failed
});
```

WebRTCAPI.quit

This API is used to exit the audio/video room.

Syntax example

```
var RTC = new WebRTCAPI( ... )
...
//Note: This must be called in the successful initialized callback of WebRTCAPI
RTC.quit( succ , error );
```

Parameter description:

Parameter	Type	Description	Note
succ	function	Callback successful	Optional
error	function	Callback failed	Optional

Actual example:

```
var RTC = new WebRTCAPI( ... )
...
//Note: This must be called in the successful initialized callback of WebRTCAPI
RTC.quit( function(){
//Exited successfully
} , function(){
//Failed to exit
} );
```

Basic Event Notifications

Last updated：2018-09-28 16:57:17

The basic event notifications include the addition/update of the local video stream, the addition/update of the remote video stream, the disconnection of the remote video stream, the disconnection of websocket, the disconnection of video stream server after timeout and the forced logout (the same user logged in repeatedly). The details are described as follows:

The basic syntax for event notification is as follows:

```
var RTC = new WebRTCAPI( { ... } );

......

RTC.on( 'EVENT_NAME' , function(data){

})
```

onLocalStreamAdd

Notification of the addition and update of the local video stream.

Sample codes

```
var RTC = new WebRTCAPI( { ... } );

RTC.on( 'onLocalStreamAdd' , function( data ){
if( data && data.stream){
var stream = data.stream
document.querySelector("#localVideo").srcObject = stream
}
})
```

data

Parameter	Type	Description
stream	Stream	Local video streamStream

onRemoteStreamUpdate

Notification of the addition and update of the remote video stream.

Sample codes

```
var RTC = new WebRTCAPI( { ... } );

RTC.on( 'onRemoteStreamUpdate' , function( data ){
if( data && data.stream){
var stream = data.stream
console.debug( data.userId + 'enter this room with unique videoId '+ data.videoId )
document.querySelector("#remoteVideo").srcObject = stream
}else{
console.debug( 'somebody enter this room without stream' )
}
})
```

data

Parameter	Type	Description
userId	String	The userId of the user to which the video stream belongs
stream	Stream	The video stream, which may be null (this callback will be triggered every time a user enters the room whether or not he/she pushes)
videoId	String	The unique ID of the video stream, consisting of tinyid + "_" + random strings
videoType:	Integer	0: NONE, 1: AUDIO, 2: MAIN 7: AUXILIARY AID

onRemoteStreamRemove

Notification of the disconnection of the remote video stream.

Sample codes

```
var RTC = new WebRTCAPI( { ... } );

RTC.on( 'onRemoteStreamRemove' , function( data ){
console.debug( data.userId + ' leave this room with unique videoId '+ data.videoId )
})
```

data

Parameter	Type	Description
userId	String	The userId of the user to which the remote video stream belongs
videoId	String	The unique ID of the remote video stream

onWebSocketClose

Notification of the disconnection of websocket.

Sample codes

```
var RTC = new WebRTCAPI( { ... } );

RTC.on( 'onWebSocketClose' , function( data ){

})
```

onRelayTimeout

Notification of the disconnection of video stream server after timeout.

Sample codes

```
var RTC = new WebRTCAPI( { ... } );

RTC.on( 'onRelayTimeout' , function( data ){
// Video server timeout
})
```

onKickout

Notification of forced logout (the same user logged in repeatedly).

Sample codes

```
var RTC = new WebRTCAPI( { ... } );

RTC.on( 'onKickout' , function( data ){
//Exit the room
})
```

Advanced Features

Last updated：2018-09-28 16:57:13

With these APIs, you can experience the features of initiating push/pull, controlling video collection during the push process, and switching audio/video parameter settings. The detailed API descriptions are as follows:

WebRTCAPI.startRTC

This API is used to initiate push/pull actively.
Parameter:

Parameter	Type	Required	Description
opt	object	Yes	
succ	function	No	Callback successful
fail	function	No	Callback failed

Parameter definitions of opts:

Parameter	Type	Required	Description
stream	MediaStream	No	Audio/video stream, MediaStream
role	string	No	Role name, which determines the bitrate control for the server to receive the video stream

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...

RTC.startRTC({
role: '',
stream: stream
}, function(){
//Successful
},function(){
//Failed
});
```

WebRTCAPI.stopRTC

This API is used to stop the push.

Parameter	Type	Description
opt	object	Reserved field, and the empty object can be passed
succ	function	Callback successful
fail	function	Callback failed

Syntax example

```
var RTC = new WebRTCAPI({ ... });
...
RTC.stopRTC({}, function(){
console.debug('stop succ')
}, function(){
console.debug('stop end')
});
```

WebRTCAPI.getLocalStream

This API is supported in version 2.5 or above
This API is used to get local audio/video streams.

Parameter	Type	Required	Description
opts	Object	No	Empty object can be passed {}
succ	function	Yes	Callback successful
fail	function	No	Callback failed

Parameter definitions of opts:

Parameter	Type	Required	Description
audio	Boolean	No	Whether to collect audio. Defaults to true
video	Boolean	No	Whether to collect video. Defaults to true
screen	Boolean	No	Whether to collect screen sharing. Defaults to false
screenSources	string	No	The media collected by the screen sharing are separated by commas. Optional options include screen window tab audio
attributes	object	No	Attributes of push related configuration
videoDevice	Device	No	Specified device. The video device obtained by getVideoDevices
audioDevice	Device	No	Specified device. The audio device obtained by getAudioDevices
needRetry	Boolean	No	Whether to allow downgrade to remove the configuration and retry when it fails to obtain some items using the parameter configuration. Defaults to true

Parameter definitions of attributes:

Parameter	Type	Required	Description
width	Integer	No	Resolution width
height	Integer	No	Resolution height
frameRate	Integer	No	Frame rate

succ callback (Object):

Parameter	Type	Description
stream	MediaStream	Audio/video stream MediaStream

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...

RTC.getLocalStream({
video:true,
audio:true,
attributes:{
width:640,
height:480,
frameRate:20
}
},function( info ){
// info { stream }
var stream = info.stream;
document.getElementById("localVideo").srcObject = stream
},function ( error ){
console.error( error )
});
```

WebRTCAPI.updateStream

This API is supported in version 2.5.3 or above
This API is used to stop the push.

Parameter	Type	Description
opt	object	Parameter
succ	function	Callback successful
fail	function	Callback failed

opt:

Parameter	Type	Description
stream	MediaStream	Reserved field, and the empty object can be passed
role	string	Optional. This parameter is required if you need to update the screen role settings when updating the stream

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...
RTC.updateStream({
role: "user",
stream: stream
}, function(){
console.debug('updateStream succ')
}, function(){
console.debug('updateStream failed')
});
```

WebRTCAPI.closeAudio

Do not perform the audio collection (mute).

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...

RTC.closeAudio();
```

WebRTCAPI.openAudio

Audio collection identification (unmute).

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...

RTC.openAudio();
```

WebRTCAPI.closeVideo

Do not perform the video collection.
Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...

RTC.closeVideo();
```

WebRTCAPI.openVideo

Enable the video collection.
The openVideo API enables the video collection when the audio/video is being pushed and the video is turned off.
Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...

RTC.openVideo();
```

WebRTCAPI.getLocalMediaStatus

This API is used to obtain the current video collection configuration.
Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...
var status = RTC.getLocalMediaStatus();
//status.video true | false (indicates whether the current configuration collects video)
//status.audio true | false (indicates whether the current configuration collects audio)
```

WebRTCAPI.changeSpearRole

This API is used to switch the user role in the Screen Setting.

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...
RTC.changeSpearRole( "role_name" );
//status.video true | false (indicates whether the current configuration collects video)
//status.audio true | false (indicates whether the current configuration collects audio)
```

WebRTCAPI.getVideoDevices

This API is used to enumerate cameras.

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...
RTC.getVideoDevices( function(devices){
//"devices" is an array that enumerates the video input devices of the current device (DeviceObject)
//For example: [device,device,device]
//These devices will be used when selecting cameras
})
```

WebRTCAPI.chooseVideoDevice

This API is used to select a camera.

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...
RTC.chooseVideoDevice( device );
```

WebRTCAPI.getAudioDevices

This API is used to enumerate microphones.

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...
RTC.getAudioDevices( function(devices){
//"devices" is an array that enumerates the audio input devices of the current device (DeviceObject)
//For example: [device,device,device]
//These devices will be used when selecting microphones
})
```

WebRTCAPI.chooseAudioDevice

This API is used to select a microphone.

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...
RTC.chooseAudioDevice( device );
```

WebRTCAPI.getSpeakerDevices

This API is supported in version 2.6 or above

This API is used to enumerate audio output devices

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...
```

```
RTC.getSpeakerDevices( function(devices){
//"devices" is an array that enumerates the audio input devices of the current device (DeviceObject)
//For example: [device,device,device]
//These devices will be used when selecting microphones
})
```

WebRTCAPI.chooseSpeakerDevice

This API is supported in version 2.6 or above

Parameter	Type	Description
media	HTMLMediaElement	Audio / Video
device	DeviceElement	Audio / Video
succ	function	Callback successful
fail	function	Callback failed

This API is used to select an audio output device
Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...
var speakerList = [];
RTC.getSpeakerDevices( function(devices){
speakerList = devices;
})
....

document.querySelectorAll("video").forEach( function(video){
console.debug(video);
RTC.chooseSpeakerDevice( device, speakerList[1],function(){
console.debug('change speaker succ ')
} ,function(error){
console.error('change speaker error ', error)
} );
});
```

WebRTCAPI.getStats

This API is used to obtain statistics and stop obtaining APIs

Parameter	Type	Required	Description
opt	object	Yes	
succ	function	Yes	Callback successful
fail	function	No	Callback failed

Details of opt parameters:

Parameter	Type	Required	Description
userId	String	No	The user ID of the statistics of the audio/video stream to be obtained; if it is null, the user's own statistics will be obtained
interval	integer	No	Timer (in milliseconds), indicates the time interval for obtaining statistics. If it is left empty, the statistics will be obtained from time to time.

Details of result:

```
//Send a stream
{
video:{
ssrc : "", //Data source ID
codec : "", //Encoding protocol
packetsSent : "", //Number of video packets sent
packetsLost : "", //Number of video packets lost
width : "", //Video resolution - width
height : "", //Video resolution - height
}
audio:{
ssrc : "", //Data source ID
codec : "", //Encoding protocol
packetsSent : "", //Number of audio packets sent
}
}
```

```
//Receive a stream
{
video:{
ssrc : "", //Data source ID
codec : "", //Encoding protocol
packetsReceived : "", //Number of video packets received
packetsLost : "", //Number of video packets lost
width : "", //Video resolution - width
height : "", //Video resolution - height
}
audio:{
ssrc : "", //Data source ID
codec : "", //Encoding protocol
packetsReceived : "", //Number of audio packets received
packetsLost : "", //Number of audio packets lost
}
}
```

Syntax example:

```
var RTC = new WebRTCAPI({ ... });
...
RTC.getStats( {
interval:2000//Obtain the statistics every 2 seconds
},function( result ){
console.debug( result );
// test code
setTimeout( function(){
//No more statistics are collected
result.nomore();
},20000);
} ,function( error ){
console.error( error );
} );
```

WebRTCAPI.SoundMeter

This API is used to detect the volume

Syntax example

This API is supported in version 2.5.3 or above
Syntax example:
`var soundMeter = WebRTCAPI.SoundMeter( opts )`

opts:

Parameter	Type	Required	Description
stream	object	Yes	MediaStream
onprocess	function	Yes	Audio stream monitoring callback

Callback parameters for oonprocess:

Parameter	Type	Description
volume	String	Volume (for example: 0.02)
status	function	volume >= 0.01 ? "speaking" : "silence"
event	AudioProcessingEventObject	

The criterion for "speaking" and "silence" is the volume value. If the value >= 0.01, it is "speaking"; if the value <0.01, it is "silence". You can also make your own judgment.

```
//Analyze the audio stream
var meter = WebRTCAPI.SoundMeter({
stream: info.stream,
onprocess: function( data ){
$("#volume").val( data.volume)
$("#volume_str").text( "volume: "+ data.volume)
$("#status").text( data.status )
}
})

//Stop analyzing the audio stream
meter.stop();
```

Advanced Event Notifications

Last updated：2018-09-28 16:57:27

The details are described as follows:

onStreamNotify

Video stream event notification.

Syntax example

```
RTC.on( 'onStreamNotify' , function( info ){ })
```

info

Parameter	Type	Description
event	String	onadd: addition of an audio/video stream onactive: disconnection of an audio/video stream
isLocal	Bool	Whether it is a local stream
stream	Stream	Whether it is a local stream
type	Type	stream/audio/video (stream is the carrier of audio and video track. If the type is stream, it indicates that the stream is disconnected)

Sample codes

```
RTC.on( 'onStreamNotify' , function( info ){

})
```

onErrorNotify

Error event notification.

Syntax example

```
RTC.on( 'onErrorNotify' , function( info ){ })
```

info

Parameter	Type	Description
errorCode	Integer	Error code
errorMsg	String	Error message

Sample codes

```
RTC.on( 'onErrorNotify' , function( info ){

})
```

onWebSocketNotify

websocket event notification.

Syntax example

```
RTC.on( 'onWebSocketNotify' , function( info ){ })
```

info

Parameter	Type	Description
errorCode	Integer	Error code
errorMsg	String	Error message
extInfo	Object	Specific information of websocket

Sample codes

```
var error_code_map = WebRTCAPI.fn.getErrorCode();

RTC.on( 'onWebsocketNotify' , function( info ){
switch( info.errorCode ){
case 0:
// conn succ
break;
case error_code_map.WS_CLOSE:
// close
console.warn( info );
break;
case error_code_map.WS_ERROR:
```

```
// error
console.error( info );
break;
default:
break;
}
})
```

onPeerConnectionAdd

PeerConnection connection notification. With this notification, the business side can determine whether a connection is required before establishing a p2p connection. It needs to be used in conjunction with the instantiated parameter peerAddNotify

There are also examples of peerconnection in our demo code that can be referenced

Syntax example

```
RTC.on( 'onPeerConnectionAdd' , function( info ){ })
```

info

Parameter	Type	Description
userId	String	User name of the user to which the connection belongs
tinyId	String	The unique 64-bit ID in Tencent Cloud corresponding to the user name of the user to which the connection belongs. You don't need to understand the role of this parameter here, and only need to pass it through when startRTC is used.

Sample codes

```
RTC.on( 'onPeerConnectionAdd' , function( info ){
// The business decides whether to establish peerconnection
if( info.userId === 'Specified user name'){
WebRTCAPI.startRTC( info );
}else{
console.debug('No connection is established')
}
})
```

Error Codes

Last updated：2018-09-28 16:46:48

Error Codes

How to use error codes

```
//Get system-defined error codes
var errorCodeMap = WebRTCAPI.fn.getErrorCode();

//Error processing
function errorHandler(error){
if( error.errorCode >= 70000){
console.error('Account system error',error.errorMsg)
}
else if( error.errorCode === errorCodeMap.XXXXXXXX){
console.error(error.errorMsg)
}
}

//Initialize
var RTC = new WebRTCAPI({ ... });
...
//Callback event
RTC.createRoom({...},function(info){
console.info(info)
}, function(error){
errorHandler(error);
});
...
//Listen for error event notification
RTC.onErrorNotify(function(error){
errorHandler(error);
})
```

Audio & Video

Key	Error Code	Error Type	Description

Key	Error Code	Error Type	Description
SUCC	0	Successful	Successful
PARAM_MISSING	10001	Parameter missing	Indicates whether the parameter is complete
INIT_WS_FAILED	10005	WS initialization failed	Websocket initialization failed
ENTER_ROOM_ERROR	10006	SDK error	Failed to join the room
CREATE_PEERCONNECTION_FAILED	10007	SDK error	Failed to create PeerConnection
GET_USERMEDIA_FAILED	10008	SDK error	Failed to get the user's audio/video device
GET_LOCALSDP_FAILED	10009	SDK error	Failed to get Local SDP
ON_ICE_BROKEN	10014	Connection error	P2P disconnected
ON_ICE_CLOSE	10015	Connection error	P2P connection closed
NOT_IN_WHITE_LIST	11000	The SdkAppid is not in the whitelist	Troubleshooting steps: Step 1: Check whether the SdkAppid is entered correctly (1400xxxxxx) Step 2: Check whether the TRTC service is activated. If not, you cannot use your SdkAppid.
NOT_FOUND	10031	SDK error	No user found
NOT_INITED	10032	SDK error	Not initialized
START_RTC_FAILED	10033	SDK error	Push failed
STOP_RTC_FAILED	10034	SDK error	Failed to stop push
WS_CLOSE	10035	Connection error	Websocket closed
WS_ERROR	10036	Connection error	Websocket error

Key	Error Code	Error Type	Description
UPDATE_VIDEO_SSRC_FAILED	10037	SDK error	Failed to update the video source
UPDATE_AUDIO_SSRC_FAILED	10038	SDK error	Failed to update the audio source
NOT_FOUND_PEER	10039	SDK error	No P2P connection found

Account System

Error Code	Error Type	Description
70001	Account system	UserSig has expired. Try to generate a new one. If it expires immediately after its generation, check if you've entered a short validity period or 0
70002	Account system	The UserSig length is 0. Confirm whether the signature calculation is correct.
70003	Account system	UserSig verification failed. Confirm whether the sig content is truncated, for example, due to insufficient buffer length
70004	Account system	UserSig verification failed. Confirm whether the sig content is truncated, for example, due to insufficient buffer length
70005	Account system	UserSig verification failed. Use a tool to check whether the generated sig is correct
70006	Account system	UserSig verification failed. Use a tool to check whether the generated sig is correct
70007	Account system	UserSig verification failed. Use a tool to check whether the generated sig is correct
70008	Account system	UserSig verification failed. Use a tool to check whether the generated sig is correct
70009	Account system	Failed to verify sig with the business public key. Confirm whether the private key for the generated UserSig matches the SdkAppid
70010	Account system	UserSig verification failed. Use a tool to check whether the generated sig is correct

Error Code	Error Type	Description
70013	Account system	The UserID in UserSig does not match that in the request. Check whether the UserID entered when you log in matches that in sig
70014	Account system	The SdkAppid in UserSig does not match that in the request. Check whether the SdkAppid entered when you log in matches that in sig
70015	Account system	No authentication method found for the appId and account type. Confirm whether the account integration is performed
70016	Account system	The length of the pulled public key is 0. Confirm whether the public key has been uploaded. Try again after 10 minutes if you upload a new public key
70017	Account system	Internal verification timed out for third-party tickets. Try again later. If you have tried repeated attempts without success, contact technical support
70018	Account system	Failed to verify third-party tickets internally
70019	Account system	The ticket field verified using https is empty. Enter a correct sig
70020	Account system	No SdkAppid found. Confirm whether it has been configured on Tencent Cloud
70052	Account system	UserSig has expired. Generate a new one and try again
70101	Account system	Request package information is empty
70102	Account system	Incorrect request package account type
70103	Account system	Incorrect phone number format
70104	Account system	Incorrect email format
70105	Account system	Incorrect TLS account format
70106	Account system	Invalid account format type

Error Code	Error Type	Description
70107	Account system	UserID does not exist
70113	Account system	Invalid batch quantity
70114	Account system	Restricted for security reasons
70115	Account system	The uin does not match the developer uin of the corresponding appId
70140	Account system	SdkAppid does not match AccType
70145	Account system	Incorrect account type
70169	Account system	Internal error. Try again later. If you have tried repeated attempts without success, contact technical support
70201	Account system	Internal error. Try again later. If you have tried repeated attempts without success, contact technical support
70202	Account system	Internal error. Try again later. If you have tried repeated attempts without success, contact technical support
70203	Account system	Internal error. Try again later. If you have tried repeated attempts without success, contact technical support
70204	Account system	appId does not have a corresponding acctype
70205	Account system	Failed to find AccType. Try again later
70206	Account system	Invalid batch quantity in the request
70207	Account system	Internal error. Try again later
70208	Account system	Internal error. Try again later

Error Code	Error Type	Description
70209	Account system	Failed to obtain the developer uin flag
70210	Account system	The uin does not match the developer uin in the request
70211	Account system	Invalid uin in the request
70212	Account system	Internal error. Try again later. If you have tried repeated attempts without success, contact technical support
70213	Account system	Failed to access the internal data. Try again later. If you have tried repeated attempts without success, contact technical support
70214	Account system	Failed to verify internal tickets
70221	Account system	Invalid login. Use UserSig to authenticate again
70222	Account system	Internal error. Try again later
70225	Account system	Internal error. Try again later. If you have tried repeated attempts without success, contact technical support
70231	Account system	Internal error. Try again later. If you have tried repeated attempts without success, contact technical support
70236	Account system	Failed to verify user signature
70308	Account system	Internal error. Try again later. If you have tried repeated attempts without success, contact technical support
70346	Account system	Failed to verify tickets
70347	Account system	Failed to verify expired tickets
70348	Account system	Internal error. Try again later. If you have tried repeated attempts without success, contact technical support

Error Code	Error Type	Description
70362	Account system	Internal timeout. Try again later. If you have tried repeated attempts without success, contact technical support
70401	Account system	Internal error. Try again later
70402	Account system	Invalid parameter. Check whether the required fields are entered, or whether the fields are entered as required in the protocol
70403	Account system	The operator is not an App admin and does not have permission to operate
70050	Account system	Restricted due to too many failures and retries. Check whether the ticket is correct and try again after one minute.
70051	Account system	The account has been blacklisted. Contact technical support

FAQs

Last updated：2018-09-28 16:46:08

Suggested steps for troubleshooting

- Identify the error code to troubleshoot the issue
- Check if the current browser supports this feature

How can I determine if the current browser supports WebRTC

Use WebRTCAPI.fn.detectRTC to check if WebRTC is supported. If false is returned, the business end will provide an error message page to guide you to use the supported environment. In special cases, you can seek help by creating a ticket.

Howling (feedback)

Note that we have set a muted attribute to the local video/audio, which means to mute the local video stream when it is played. Otherwise, there will be a loop where the sound of the local video stream is once again used as an audio input source, causing a problem of "howling" or "feedback".

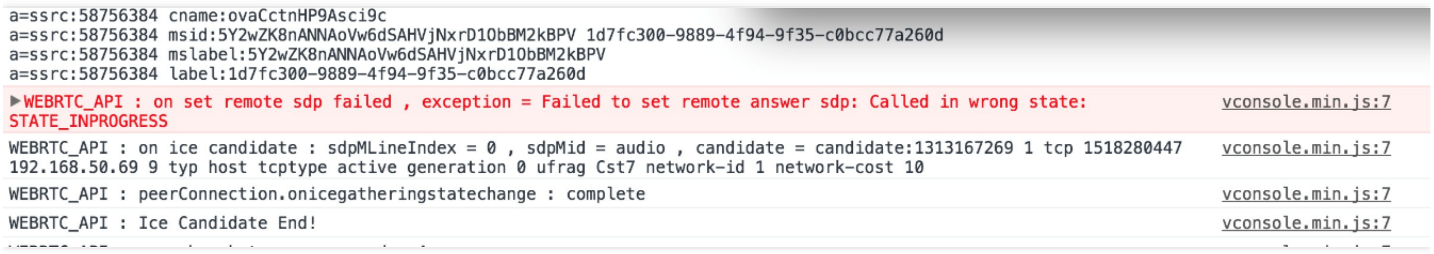
```
<video muted autoplay playsinline></video>

<audio muted autoplay playsinline></audio>
```

It takes a long time to hear the audio

In an audio-only scenario, be sure to use the audio instead of the video tag to load the audio stream.

on set remote sdp failed (as shown below)



```
a=ssrc:58756384 cname:ovaCctnHP9Asci9c
a=ssrc:58756384 msid:5Y2wZK8nANNAoVw6dSAHVjNxrD1ObBM2kBPV 1d7fc300-9889-4f94-9f35-c0bcc77a260d
a=ssrc:58756384 mslabel:5Y2wZK8nANNAoVw6dSAHVjNxrD1ObBM2kBPV
a=ssrc:58756384 label:1d7fc300-9889-4f94-9f35-c0bcc77a260d
▶WEBRTC_API : on set remote sdp failed , exception = Failed to set remote answer sdp: Called in wrong state:    vconsole.min.js:7
STATE_INPROGRESS
WEBRTC_API : on ice candidate : sdpMLineIndex = 0 , sdpMid = audio , candidate = candidate:1313167269 1 tcp 1518280447    vconsole.min.js:7
192.168.50.69 9 typ host tcptype active generation 0 ufrag Cst7 network-id 1 network-cost 10
WEBRTC_API : peerConnection.onicegatheringstatechange : complete                                                       vconsole.min.js:7
WEBRTC_API : Ice Candidate End!                                                                                        vconsole.min.js:7
```

There is a parameter closeLocalMedia in the webrtcapi instantiation method.
It indicates whether the auto push is disabled. If it is set to false (the default value), but startWebRTC is called, this problem will occur.

Power consumption of your mobile phone

Videos need to be encoded/decoded, which is quite power-consuming. However, no push/playback on the page still consumes a lot of power. You must check if the video's srcObject is not reset during the callback for non-push.

```
videoElement.srcObject = null
```

SecurityError [Security error]

The audio/video cannot be obtained correctly.
WebRTC must be enabled in the page of HTTPS or localhost, otherwise the audio/video device cannot be obtained.

NotAllowedError [Rejection error]

The user rejected the request to obtain the audio/video device

OverConstrainedError [Error: The device does not meet the requirements]

The specified requirements cannot be met by the device. This exception is an object of OverconstrainedError type and has a constraint attribute that contains constraint pairs that cannot be satisfied. If multiple tabs are enabled for push at a time, make sure the resolution to be collected is consistent.

NotFoundError [Error: Not found]

The media type that meets the request parameter was not found.

NotReadableError [Error: Unable to read]

Even if the user has been authorized to use an appropriate device, it cannot be accessed due to a hardware, browser or web page error on the operating system.

AbortError [Termination error]

Even if both the user and the operating system have been granted the access to the device hardware and no problem such as NotReadableErro caused by hardware occurs, the device still cannot be used due to some problems.

TypeError [Type error]

The constraints objects are not set or set to false.

No sound

The browser uses the default sound output device. In this case, adjust the sound output device and disable other devices than the amplifier to determine if it works.

Unable to make a video call in the Electron development environment

If you are using Electron and are unable to make a video call after submitting it to the Mac App Store, please add com.apple.securite.network.server to the entitlements.plist file.

Be careful with the black screen caused by dom tree redrawing

If you are using react/vue/angular, pay special attention to the relationship between video and stream, which is controlled by JS. If data changes cause page changes, you need to rebind video with stream, otherwise a black screen will occur.

Black screen on iOS

If you are using react/vue/angular, note that a video created dynamically is not automatically played in a browser.
In the viewer mode (non-push), iOS does not allow auto playback of videos with sound and remote video streams cannot be played automatically. You need to bind the remote streams to the video tag and add video.play() in the onRemoteStreamUpdate event handling function.