CEC
2.5.0.0.0

Agent Integration-OpenEye H5 Softphone Interface Integration

Issue	01
Date	2024-03-01

HUAWEI TECHNOLOGIES CO., LTD.

Huawei Technologies Co., Ltd.

Address:	Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China
Website:	https://www.huawei.com
Email:	support@huawei.com

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Change History

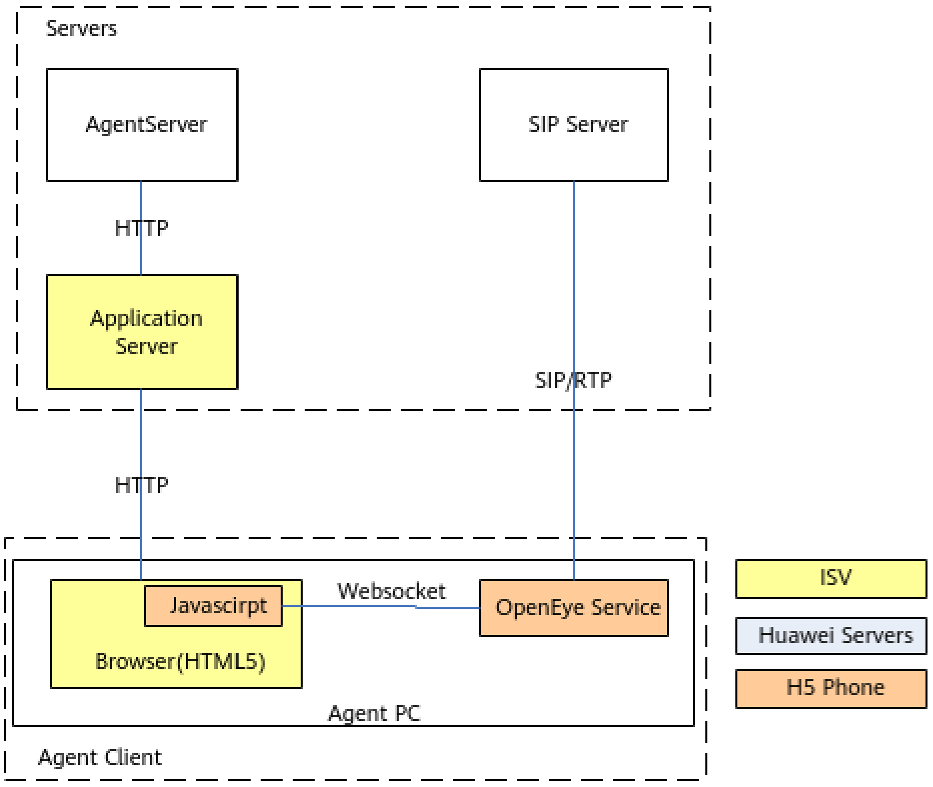
Date	Issue	Description
2021-03-08	03	Updated some interfaces.
2020-07-09	02	Added screen sharing-related interfaces.
2019-12-02	01	This issue is the first official release.

2 OpenEye H5 Softphone Interface Overview

The OpenEye H5 softphone supports most functions of the OpenEye on web pages. The H5 softphone can run only after the OpenEye is installed. The software and hardware requirements of the H5 softphone are the same as those of the OpenEye.

The OpenEye H5 softphone API is a JavaScript interface based on HTML5 (H5) and can be used to control the voice and video call components of the OpenEye client.

The H5 softphone interface is implemented based on the internal component of Huawei OpenEye. Before the browser integrates the H5 JavaScript interface, you need to install the OpenEye client software on the local PC. For details, see **3 OpenEye Installation Guide**.


Servers
AgentServer
SIP Server
HTTP
Application Server
SIP/RTP
HTTP
Javascirpt
Websocket
OpenEye Service
Browser(HTML5)
Agent PC
Agent Client
ISV
Huawei Servers
H5 Phone

3 OpenEye Installation Guide

3.1 Obtaining the Installation Package

Obtain the **AICC_***_OpenEye.zip** package from the AICC support website(Contact O&M personnel to obtain the website.).

After obtaining the software package, you must verify the integrity of the package. For details, see "Installation and Commissioning > Installation Guide." Only verified software packages can be deployed.

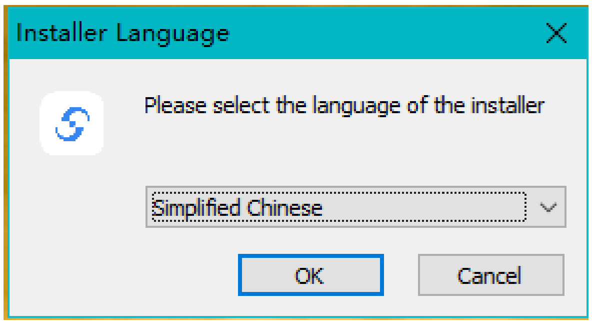
Table 3-1 OpenEye installation package description

Name	Description
AICC_***_OpenEye.zip	OpenEye desktop installation package

3.2 Performing Installation

Step 1 Decompress **AICC_***_OpenEye.zip** and double-click **OpenEyeSetup.exe**.

Figure 3-1 Selecting an installation language

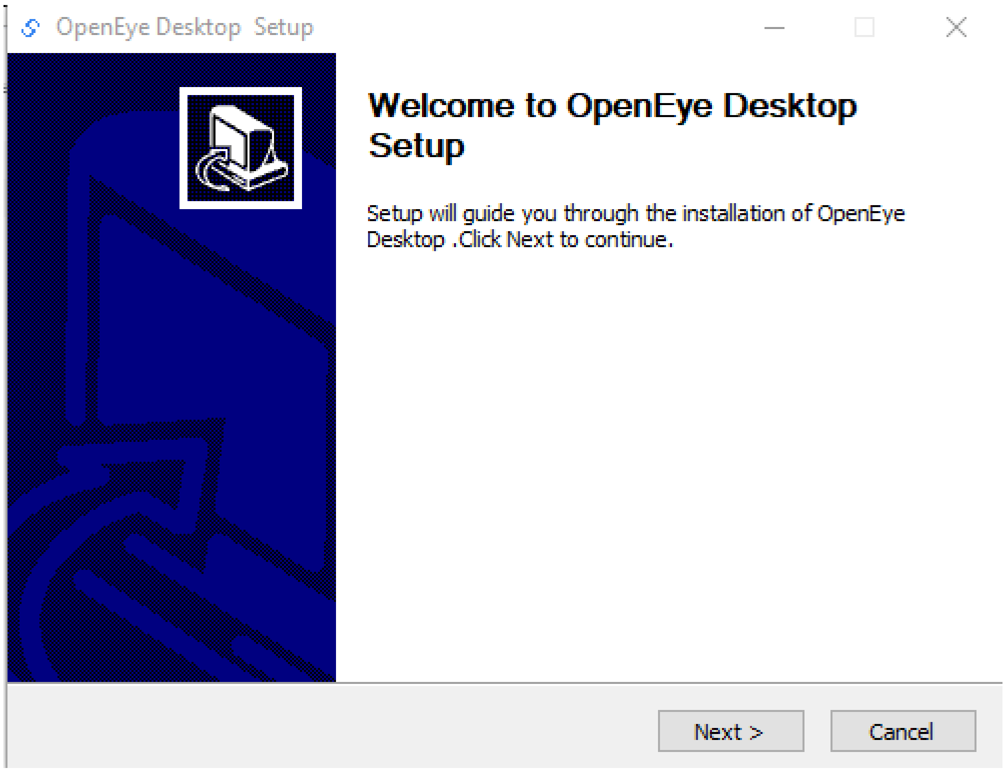

Installer Language

Please select the language of the installer

Simplified Chinese

Step 2 Click **OK**.

Figure 3-2 Installation wizard

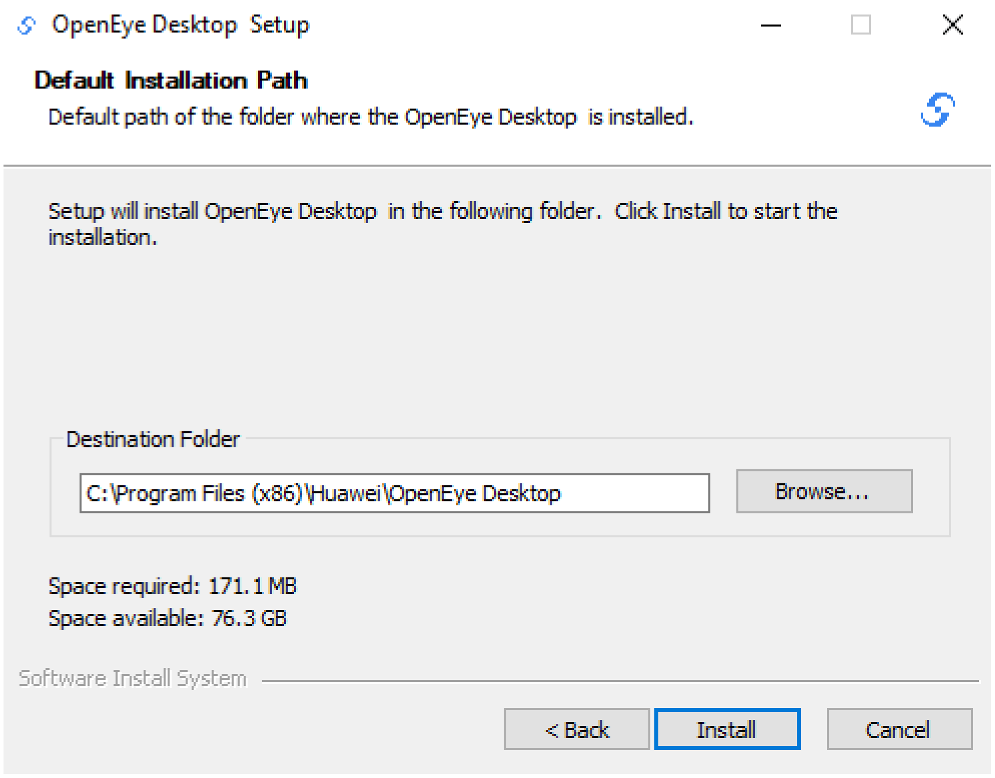


Welcome to OpenEye Desktop Setup

Setup will guide you through the installation of OpenEye Desktop .Click Next to continue.

Step 3 Click **Next**.

Figure 3-3 Selecting the destination folder



Default Installation Path

Default path of the folder where the OpenEye Desktop is installed.

Setup will install OpenEye Desktop in the following folder. Click Install to start the installation.

Destination Folder

C:\Program Files (x86)\Huawei\OpenEye Desktop

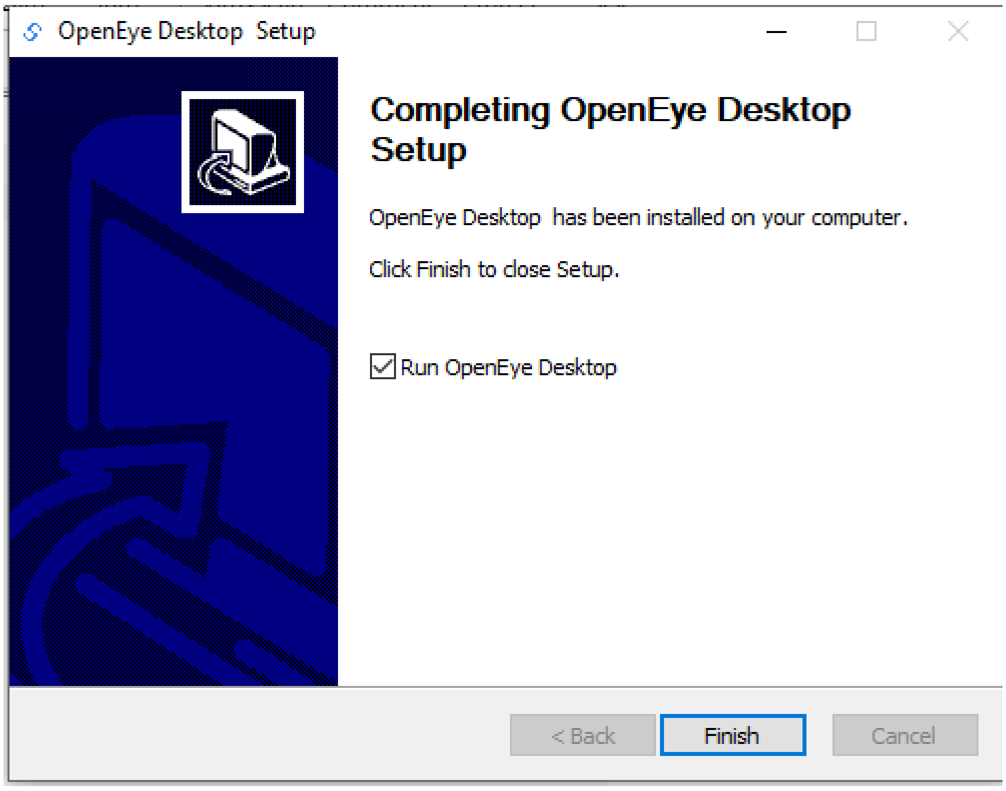
Browse...

Space required: 171.1 MB

Space available: 76.3 GB

Step 4 Click **Install**. After the installation is complete, the following page is displayed.

Figure 3-4 Installation result



Completing OpenEye Desktop Setup

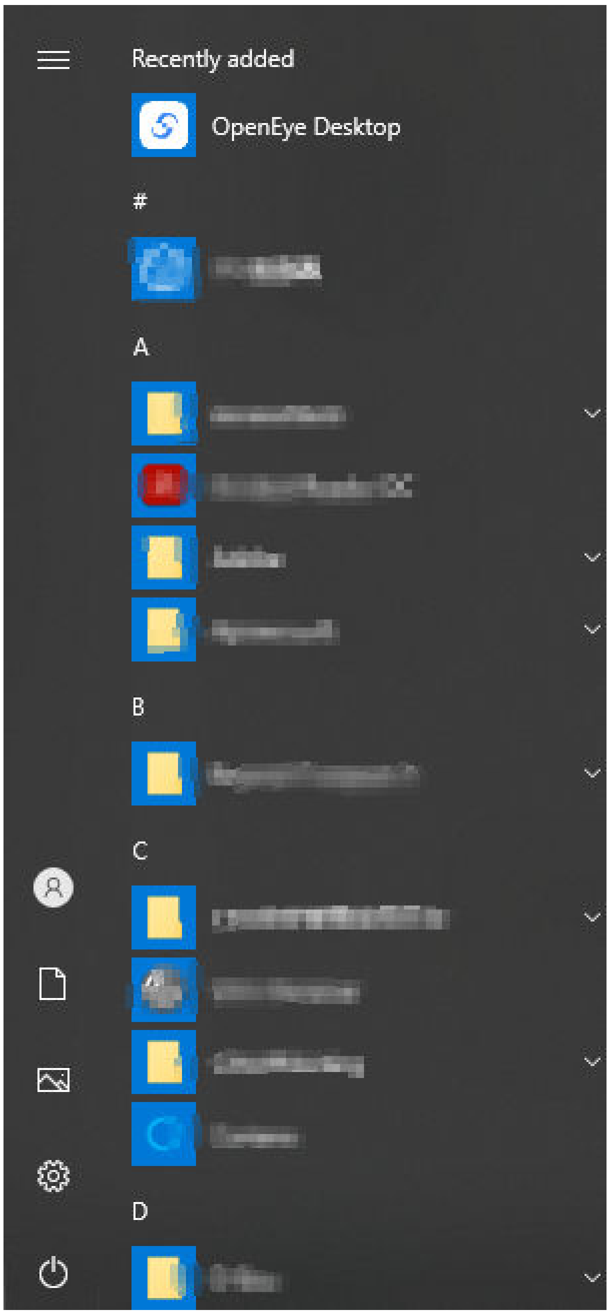
OpenEye Desktop has been installed on your computer.

Click Finish to close Setup.

☑ Run OpenEye Desktop

Step 5 Click **Finish**. In the OS startup items, add the installation information.

Figure 3-5 Software installation information



----End

3.3 Verification

Prerequisites

The OpenEye has been installed based on **3 OpenEye Installation Guide** and is running properly.

The WebDemo has been obtained from R&D engineers. The download URL is https://bbs.huaweicloud.com/forum/thread-132809-1-1.html.

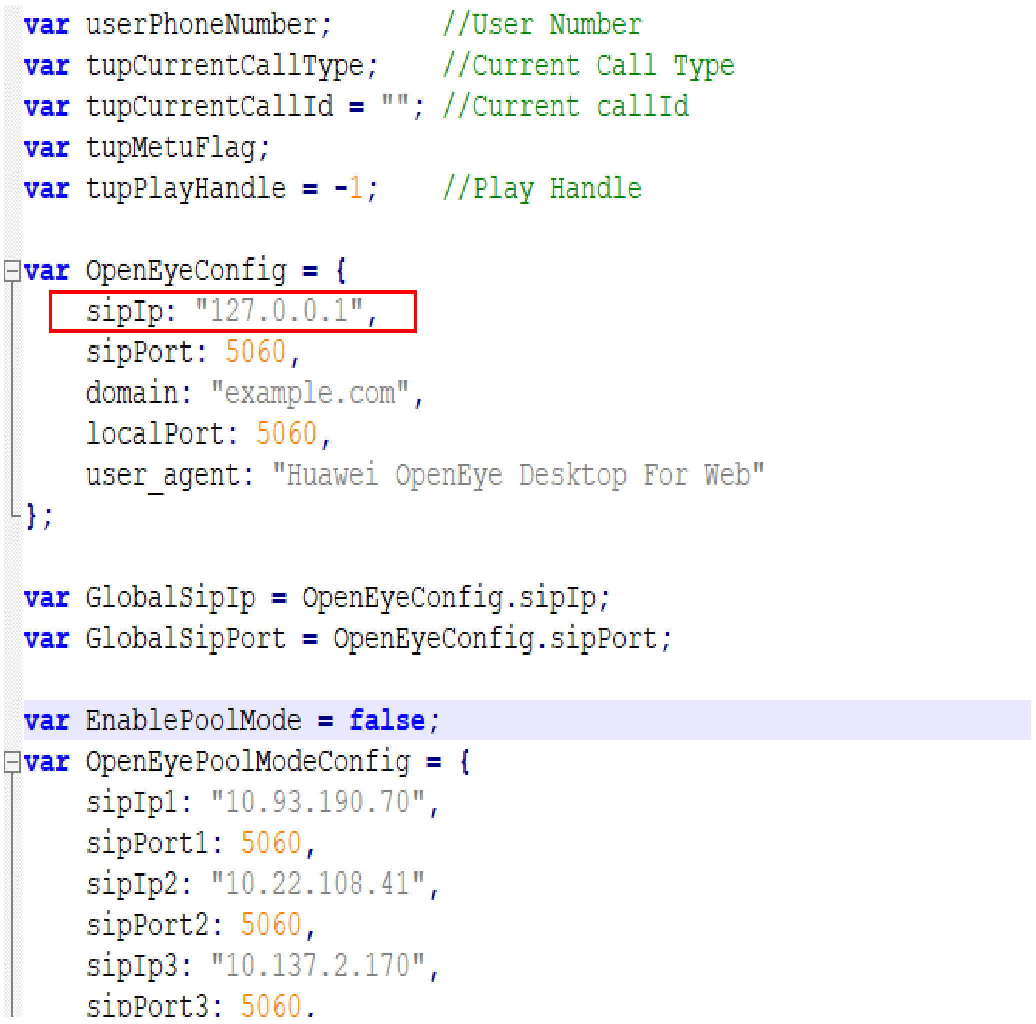
You have run the command for starting the OpenEye through a command line or JS script (that is, start the OpenEye without opening the OpenEye GUI).

1. To start the OpenEye through a command line, run the following command in the program directory:
```
ClientApp.exe -mode 2
```
2. To start the OpenEye through a web page integration JS script, run the following command:
```
window.location.href='OpenEyeWebApiShell://-mode,2/';
```

Procedure

Step 1 Use the text editor to open the **param.js** file in **WebDemo** in the **webdemo** package, as shown in the following figure.

Figure 3-6 param.js



```
var userPhoneNumber;        //User Number
var tupCurrentCallType;     //Current Call Type
var tupCurrentCallId = ""; //Current callId
var tupMetuFlag;
var tupPlayHandle = -1;     //Play Handle

var OpenEyeConfig = {
    sipIp: "127.0.0.1",
    sipPort: 5060,
    domain: "example.com",
    localPort: 5060,
    user_agent: "Huawei OpenEye Desktop For Web"
};

var GlobalSipIp = OpenEyeConfig.sipIp;
var GlobalSipPort = OpenEyeConfig.sipPort;

var EnablePoolMode = false;
var OpenEyePoolModeConfig = {
    sipIp1: "10.93.190.70",
    sipPort1: 5060,
    sipIp2: "10.22.108.41",
    sipPort2: 5060,
    sipIp3: "10.137.2.170",
    sipPort3: 5060,
```

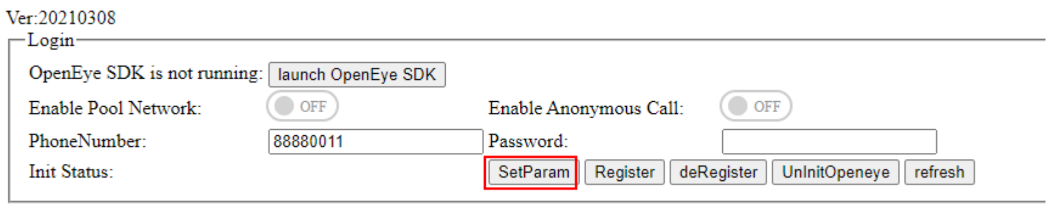
Step 2 Modify the configuration information in **Table 3-2** and save the modification. If the pool mode is used, set **EnablePoolMode** to **true**.

Table 3-2 Configuration information to be modified

Parameter	Description
sipIp	Signaling IP address of the UAP.
sipPort	Port number of the SIP server. The default value is **5060**.
sipIp1~sipIp4	Signaling IP address of the UAP on the pool network.
sipPort1~sipPort4	Port number of the SIP server on the pool network. The default value is **5060**.

Step 3 Use the Google Chrome to open the **WebDemo/index.html** file. The page shown in the following figure is displayed. Click **SetParam** to perform basic configuration.

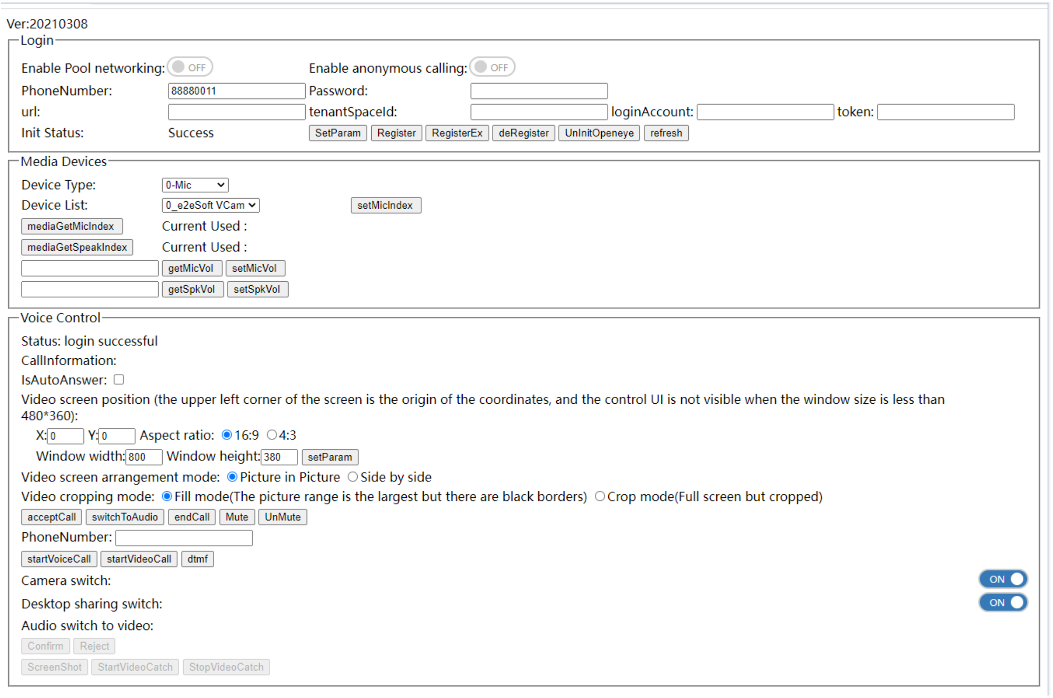
Figure 3-7 index.html



Ver:20210308
Login
OpenEye SDK is not running:
Enable Pool Network: OFF
Enable Anonymous Call: OFF
PhoneNumber: 88880011
Password:
Init Status:

Step 4 Set **PhoneNumber** and click **Register**. After the registration is successful, the page shown in the following figure is displayed.

Figure 3-8 Registration result



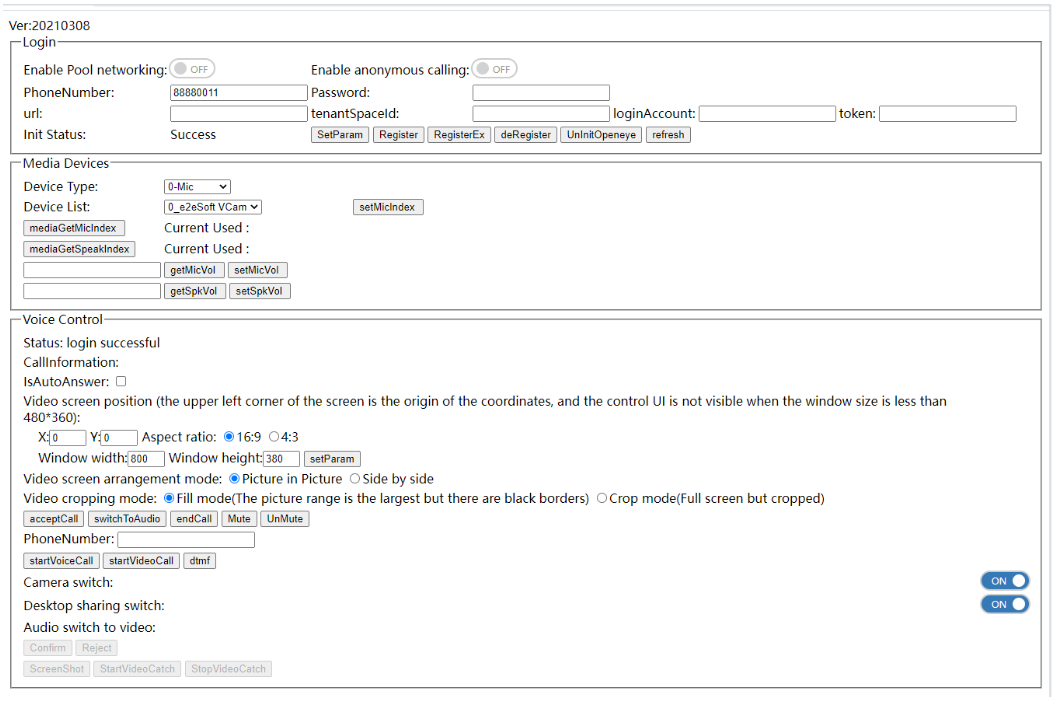
Ver:20210308
Login
Enable Pool networking: OFF
Enable anonymous calling: OFF
PhoneNumber: 88880011
Password:
url:
tenantSpaceId:
loginAccount:
token:
Init Status: Success

Media Devices
Device Type: 0-Mic
Device List: 0_e2eSoft VCam
Current Used :
Current Used :

Voice Control
Status: login successful
CallInformation:
IsAutoAnswer: ☐
Video screen position (the upper left corner of the screen is the origin of the coordinates, and the control UI is not visible when the window size is less than 480*360):
X: 0 Y: 0 Aspect ratio: [✓ 16:9] 4:3
Window width: 800 Window height: 380
Video screen arrangement mode: [✓ Picture in Picture] Side by side
Video cropping mode: [✓ Fill mode(The picture range is the largest but there are black borders)] Crop mode(Full screen but cropped)
PhoneNumber:
Camera switch: ON
Desktop sharing switch: ON
Audio switch to video:

Step 5 In the **Voice Control** area, set **Phone Number** to the number of another phone that has been registered with the USM (that is, the registered softphone number, for example, **444002**), and click **startVoiceCall**, as shown in the following figure.

Figure 3-9 Outbound call operation



Ver:20210308

Login

Enable Pool networking: OFF Enable anonymous calling: OFF
PhoneNumber: 88880011 Password:
url: tenantSpaceId: loginAccount: token:
Init Status: Success SetParam Register RegisterEx deRegister UnInitOpeneye refresh

Media Devices

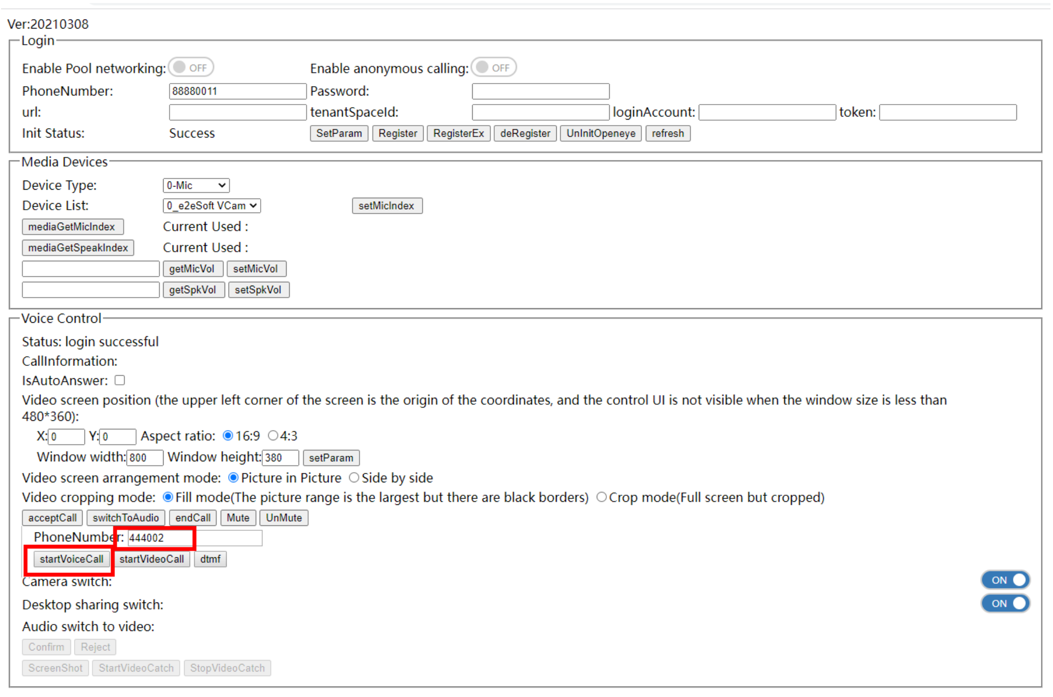
Device Type: 0-Mic
Device List: 0_e2eSoft VCam setMicIndex
mediaGetMicIndex Current Used :
mediaGetSpeakIndex Current Used :
getMicVol setMicVol
getSpkVol setSpkVol

Voice Control

Status: login successful
CallInformation:
IsAutoAnswer: ☐
Video screen position (the upper left corner of the screen is the origin of the coordinates, and the control UI is not visible when the window size is less than 480*360):
X: 0 Y: 0 Aspect ratio: [✓ 16:9] 4:3
Window width: 800 Window height: 380 setParam
Video screen arrangement mode: [✓ Picture in Picture] Side by side
Video cropping mode: [✓ Fill mode(The picture range is the largest but there are black borders)] Crop mode(Full screen but cropped)
acceptCall switchToAudio endCall Mute UnMute
PhoneNumber:
startVoiceCall startVideoCall dtmf
Camera switch: ON
Desktop sharing switch: ON
Audio switch to video:
Confirm Reject
ScreenShot StartVideoCatch StopVideoCatch

Step 6 Answer the call from the number 444002, as shown in the following figure.

Figure 3-10 Outbound call result



Ver:20210308

Login

Enable Pool networking: OFF Enable anonymous calling: OFF
PhoneNumber: 88880011 Password:
url: tenantSpaceId: loginAccount: token:
Init Status: Success SetParam Register RegisterEx deRegister UnInitOpeneye refresh

Media Devices

Device Type: 0-Mic
Device List: 0_e2eSoft VCam setMicIndex
mediaGetMicIndex Current Used :
mediaGetSpeakIndex Current Used :
getMicVol setMicVol
getSpkVol setSpkVol

Voice Control

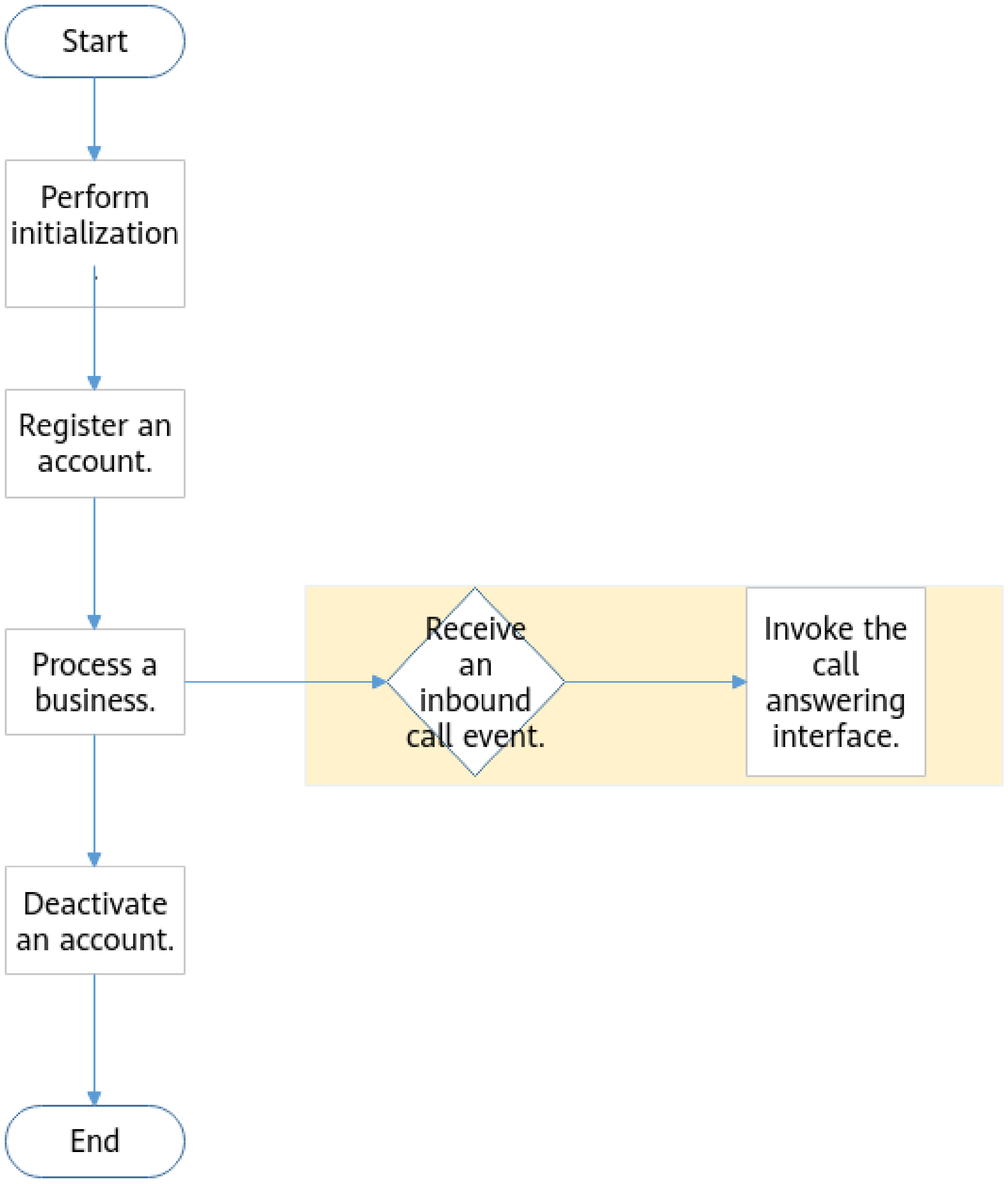
Status: login successful
CallInformation:
IsAutoAnswer: ☐
Video screen position (the upper left corner of the screen is the origin of the coordinates, and the control UI is not visible when the window size is less than 480*360):
X: 0 Y: 0 Aspect ratio: [✓ 16:9] 4:3
Window width: 800 Window height: 380 setParam
Video screen arrangement mode: [✓ Picture in Picture] Side by side
Video cropping mode: [✓ Fill mode(The picture range is the largest but there are black borders)] Crop mode(Full screen but cropped)
acceptCall switchToAudio endCall Mute UnMute
PhoneNumber: 444002
startVoiceCall startVideoCall dtmf
Camera switch: ON
Desktop sharing switch: ON
Audio switch to video:
Confirm Reject
ScreenShot StartVideoCatch StopVideoCatch

----End

4 Development Guide to H5 Softphone Integration on the Agent Side

4.1 Overall Process

When you use the H5 softphone interface provided by the OpenEye for integration development, the basic process includes initializing business components, registering accounts, processing businesses, and deactivating subscriber accounts.



Process Description

1. Component initialization: Initialize resources for business components and set global business parameters for third-party applications.
2. Account registration: The corresponding interface is invoked to register an account with the SIP server.
3. Business processing: After receiving an inbound call event, the third-party application automatically invokes the call answering interface.
4. Account deactivation: Users can log out of the OpenEye to ensure the security of business interfaces. If an account is not deactivated, the OpenEye keeps the account registered.

4.2 Sample Codes

Sample code download path: none

Contact the OpenEye development and operation team to obtain the sample code and technical support.

Name	Description
WebDemo	Web page demo based on the OpenEye desktop, which provides voice and video call and device management functions. This code is for reference only. You are advised to develop web pages by yourself.

4.3 Performing Initialization

Application Scenario

To use the OpenEye softphone interface to register a phone number, an agent needs to initialize the components.

Prerequisites

1. The WebDemo has been downloaded.
2. The local OpenEye program package has been installed and the local OpenEye program has been started.
3. The server information in **param.js** has been configured.

Process Description

NOTE

The code path in the demo is **WebDemo\js\OpenEye_SDK.js**.

Step 1 Initialize the SDK by referring to **5.1.1 OpenEye_SDK (Creating and Initializing an Object)**.

NOTE

After opening the **webdemo** page, you can receive a message returned by the OpenEye. Interfaces under OpeneyeLogin can be invoked only after the onOpeneyeLoginReady message is received. Interfaces under OpeneyeCall can be invoked only after the onOpeneyeCallReady message is received.

```
function initOpeneye(){
    if (global_openEye_SDK== null)
    {
       global_openEye_SDK = new OpenEye_SDK({
        onOpeneyeDeamonReady: onOpeneyeDeamonReady,
        onOpeneyeDeamonClose: onOpeneyeDeamonClose,
        serviceStartUp: serviceStartUp,
        serviceShutDown: serviceShutDown,
        onOpeneyeLoginReady: onOpeneyeLoginReady,
        onOpeneyeLoginClose: onOpeneyeLoginClose,
        onOpeneyeCallReady: onOpeneyeCallReady,
        onOpeneyeCallClose: onOpeneyeCallClose,
        onVersionInfoNotify : onVersionInfoNotify
    });

    }
}
function onVersionInfoNotify(data)
{
    writeLog("version is " + data.param.version);
}
function onOpeneyeDeamonReady() {
    writeLog("OpenEye Deamon is Ready");
}

function onOpeneyeDeamonClose() {
    writeLog("OpenEye Deamon is Closed, please restart OpenEye Deamon it.");
    global_openEye_SDK= null;
}


function serviceStartUp() {
    writeLog("Openeye Service StartUp");
}

function serviceShutDown() {
    writeLog("Openeye Service is shutdown, please restart it.");
}

function onOpeneyeLoginReady(){
    writeLog("onOpeneyeLoginReady");
}


function onOpeneyeLoginClose(){
    writeLog("onOpeneyeLoginClose");
}


function onOpeneyeCallClose() {
    writeLog("onOpeneyeCallClose");
}


function onOpeneyeCallReady() {
    writeLog("onOpeneyeCallReady");
}
```

Step 2 Initialize OpenEyeCall parameters and listen to call events.

```
function initOpeneyeCall(localIp)
{
    console.info("onOpeneyeCallReady");
    initOpeneyeCall();
}
```

- Invoke the events of the OpenEyeCall that are listened on by invoking the **setBasicCallEvent** interface.

```
function initOpeneyeCall()
{
    global_openEye_SDK.openEyeCall.setBasicCallEvent({
        onCallIncoming: onCallIncoming,
        onCallOutGoing: onCallOutGoing,
        onCallRingBack: onCallRingBack,
        onCallConnected: onCallConnected,
        onCallEnded: onCallEnded,
        onCallEndedFailed: onCallEndedFailed,
        onCallRtpCreated: onCallRtpCreated,
    onCallOpenVideoReq: onCallOpenVideoReq
    });
}
```

- Invoke the **config** interface to set the SIP server information of OpenEyeCall.

  NOTE

  Ensure that the SIP server information is correct. Otherwise, the account cannot be registered.

```
function sipBasicCfg() {
    global_cloudIPCC_SDK.openEyeCall.config({
        networkInfo: {
            serverAddr: GlobalSipIp,
            sipServerPort: GlobalSipPort,
            sipTransportMode: 0,
            httpPort: 0
        },
    }, {response: configResponse});
}

function configResponse(data)
{
    if (data.result == 0) {
        writeLog("Set OpenEyeCall Sip Config Success.");
        register();
    } else {
        writeLog("Set OpenEyeCall Sip Config Failed.");
    }

}
```

----End

4.4 Registering an Account

Application Scenario

Register an account using an agent's phone.

Prerequisites

The OpenEyeCall parameters have been initialized, and the SIP server information is correct.

Process Description

Invoke the **register** interface of OpenEyeCall to answer a call.

NOTE

- When the registration interface is invoked, the returned result needs to be processed, and the **onRegStatusUpdate** event needs to be listened to.
- Registration is an asynchronous process. The returned result **0** does not necessarily indicate a successful registration. The registration is successful only when the value of **register_state** returned by onRegStatusUpdate is **3**.

```
/**
* Register
 */
function register() {
    var phoneNumber = document.getElementById("phoneNumber").value;
    var password = document.getElementById("password").value;
    userPhoneNumber = phoneNumber + "@" + GlobalSipIp;//OpenEyeConfig.domain;
    console.info("register info="+userPhoneNumber);

    var sipMode=1;
    if(EnablePoolMode){
        sipMode=1;
    }
    this.global_openEye_SDK.openEyeCall.register(userPhoneNumber, phoneNumber, password, sipMode, {
        onRegStatusUpdate: onRegStatusUpdate,
        onForceUnReg: onForceUnRegInfo,
        response: registerResponse
    });
}
/**
* Registration result
 * @param {*} data
 */
function registerResponse(data) {
   if (data.result == 0) {
        console.info("Register Operation Success");
    getMediaDevices();
    } else {
        console.error("Register Operation Failed");
        console.error(data);
    }

}

/**
* Registration result reporting
 * @param {*} data
 */
function onRegStatusUpdate(data) {
    var userNumber = data.param.user_number;
    var state = ["unregister", "registering", "deregistering", "registered", "deregistered"];
    var reason = data.param.reason_code; //complete reason code refer to: http://blog.csdn.net/
kafeiwuzhuren/article/details/7242791
    if (reason == 403) {
        console.log("403:forbidden");
    }
    if (reason == 408) {
        console.log("408:request overtime");
    }
    var currentState = state[data.param.register_state];
    var obj = { userNum: userNumber, stateInfo: currentState, reasonInfo: reason }
    console.log("onRegStatusUpdate");
    console.log(data);
    document.getElementById("phoneStatus").innerText = obj.stateInfo;
    if (data.param.register_state == 3) {
        document.getElementById("voiceControlDiv").style.visibility = "visible";
    }
    if(data.notify==1004){
        document.getElementById("phoneStatus").innerText = "Login successful";
        document.getElementById("voiceControlDiv").style.visibility = "visible";
    }

}

function onForceUnRegInfo(data) {
    document.getElementById("phoneStatus").innerText = "DeRegister";
    document.getElementById("voiceControlDiv").style.visibility = "hidden";
    userPhoneNumber = "";
    tupCurrentCallId = "";
    console.log("onForceUnRegInfo");
    console.log(data);

}
```

4.5 Answering a Call

Application Scenario

After receiving the inbound call event reported by the OpenEye, the agent automatically invokes the call answering interface to answer the call.

Prerequisites

- An agent has signed in to the CTI platform.
- The **onCallIncoming** message is received.

Process Description

Invoke the **acceptCall** interface of OpenEyeCall to answer a call.

```
/**
* Inbound call event
 **/
function onCallIncoming(data) {
    //Record the inbound call ID. This parameter will be used in the subsequent call answering interface.
    var tupCurrentCallId = data.param.call_id;

    //The following describes how to invoke the interface for answering a call. In actual development, invoke
the  interface as required.
    acceptCall(tupCurrentCallId);
}
/**
* Answer a call
 */
function acceptCall(tupCurrentCallId) {
   if (tupCurrentCallStatus == OPENEYE_CALL_STATUS.ALERTING) {
        this.global_openEye_SDK.openEyeCall.acceptCall(tupCurrentCallId, tupCurrentCallType, { response:
onAcceptCallReponse });
    } else {
        console.error("Phone status is invalid. Now it's " + tupCurrentCallStatus);
        alert("Phone status is invalid. Now it's " + tupCurrentCallStatus);
    }
}
/**
* Response of the answering interface
 */
function onAcceptCallReponse(data) {
    if (data.result == 0) {
        console.error("AcceptCall success. ");
    } else {
        console.error("AcceptCall failed. The ErrorCode is " + data.result);
        console.info(data);
        alert("AcceptCall failed.  The ErrorCode is " + data.result);
    }
}
```

4.6 Making a Call

Application Scenario

After successful login, the agent invokes the call interface to initiate a voice or video call.

Prerequisites

- An agent has signed in to the CTI platform.
- The phone account login and registration are successful.

Process Description

Invoke the **startCall** interface of OpenEyeCall to answer a call.

```
/**
* Outbound call
 */
function startCall() {
    //Distinguish anonymous calls from non-anonymous calls
    var ischecked = document.getElementById("toggle-button-anonymous").checked;
    if(ischecked){

this.global_openEye_SDK.openEyeCall.startAnonymousCall(document.getElementById("calloutNumber").valu
e, false, {
            response: startCallResponse
        });
    }else{
        this.global_openEye_SDK.openEyeCall.startCall(document.getElementById("calloutNumber").value,
false, {
            response: startCallResponse
        });
    }
}

/**
* Response to the outbound call
 */
function startCallResponse(data) {
    if (data.result == 0) {
        console.info("StartCall success. callid="+JSON.stringify(data));
        tupCurrentCallId = data.param.callId;
    } else {
        console.error("StartCall failed. The ErrorCode is " + data.result);
        console.info(data);
        alert("StartCall failed.  The ErrorCode is " + data.result);
    }
}
```

4.7 Ending a call

Application Scenario

After successful login, the agent proactively invokes the call ending interface during an inbound or outbound call to end a voice or video call.

Prerequisites

- An agent has signed in to the CTI platform.
- The phone account login and registration are successful.
- The agent is in a call or in a conversation.

Process Description

Invoke the **endCall** interface of OpenEyeCall to answer a call.

4.8 Deactivating an Account

Application Scenario

After signing out of the CTI platform, the agent also needs to deactivate the phone account.

Prerequisites

- An agent has signed in to the CTI platform.
- The account has been registered.

Process Description

Invoke the **deRegister** interface of the OpenEyeCall to deactivate the account.

```
/**
* Deactivate
 */
function deRegister() {
    if (global_openEye_SDK!= null && global_openEye_SDK.openEyeCall!= null)
    {
        global_openEye_SDK.openEyeCall.deRegister(global_userPhoneNumber, {
            response: deRegisterResponse
        });
    }
}
/**
* Deactivation result
 * @param {*} data
 */
function deRegisterResponse(data) {
    if (data.result == 0) {
        writeLog("Phone DeRegister Success.");
    } else {
        writeLog("Phone DeRegister Failed.");
    }
}
```

5 Voice Call Interfaces

5.1 Initialization

5.1.1 OpenEye_SDK (Creating and Initializing an Object)

Interface Description

When the SDK is initialized, the WebSocket connections with the OpenEyeDeamon, OpenEyeLogin, and OpenEyeCall modules are implemented internally.

Notes

- The local client of the OpenEye is started.
- Only one web page can be used to initialize the SDK on each PC.
- The **OpenEye_SDK.js** file has been loaded to the third-party application page.

Method Definition

```
function OpenEye_SDK(opts)
```

Parameter Description

Table 5-1 Parameter description

Parameter	Type	Mandatory/ Optional	Description
opts	**Opts**	Mandatory	Callback method.

Table 5-2 Opts

Parameter	Type	Mandatory/ Optional	Description
onOpeneyeDeamon-Ready	function	Mandatory	The WebSocket connection with the OpenEye client is set up.
onOpeneyeDeamon-Close	function	Mandatory	The WebSocket connection with the OpenEye client is closed. **NOTE** If the WebSocket connection with the OpenEye client is closed, the WebSocket connections with the OpenEyeCall and OpenEyeLogin are also closed.
serviceStartUp	function	Mandatory	The local OpenEye service is started. **NOTE** The WebSocket connections with OpenEyeCall and OpenEyeLogin can be set up only after the local OpenEye service is started.
serviceShutDown	function	Mandatory	The local OpenEye service is disabled.

Parameter	Type	Mandatory/Optional	Description
onOpeneyeLoginReady	function	Mandatory	The WebSocket connection with the OpenEyeLogin is set up.
onOpeneyeLoginClose	function	Mandatory	The WebSocket connection with the OpenEyeLogin is closed.
onOpeneyeCallReady	function	Mandatory	The WebSocket connection with the OpenEyeCall is set up.
onOpeneyeCallClose	function	Mandatory	The WebSocket connection with the OpenEyeCall is closed.
onVersionInfoNotify	function	Mandatory	Version information notification.

Examples

```
function onOpeneyeDeamonReady() {
    console.info("Openeye Deamon is Ready");
}

function onOpeneyeDeamonClose() {
    console.error("Openeye Deamon is Closed,please restart it");
    global_openEye_SDK = null;
}


function serviceStartUp() {
    console.info("OpenEye Service StartUp");
}

function serviceShutDown() {
    console.error("OpenEye Service is shutdown,please restart it");
}

function onOpeneyeCallClose() {
    console.error("onOpeneyeCallClose");
}


function onOpeneyeCallReady() {
    console.info("onOpeneyeCallReady");
}

function onOpeneyeLoginReady() {
    console.info("onTupLoginReady");
}
```

```
function onOpeneyeLoginClose() {
    console.info("onOpeneyeLoginClose");
}
function onVersionInfoNotify (data) {
    console.info("version is");
    console.info(data);
}

var global_openEye_SDK = null;
function initSDK(){
    global_openEye_SDK = new OpenEye_SDK({
            onOpeneyeReady: onOpeneyeReady,
            onOpeneyeClose: onOpeneyeClose,
            serviceStartUp: serviceStartUp,
            serviceShutDown: serviceShutDown,
            onOpeneyeLoginReady: onOpeneyeLoginReady,
            onOpeneyeLoginClose: onOpeneyeLoginClose,
            onOpeneyeCallReady: onOpeneyeCallReady,
            onOpeneyeCallClose: onOpeneyeCallClose,
            onVersionInfoNotify: onVersionInfoNotify
        });
}
```

5.1.2 config (Configuration)

Interface Description

This interface is invoked to configure the OpenEyeCall running parameters.

Notes

The WebSocket connection with the OpenEyeCall is set up.

Method Definition

```
TUPCall.prototype.config = function(params, callbacks)
```

Parameter Description

Table 5-3 Parameter description

Parameter	Type	Mandatory/Optional	Description
params	**Params**	Mandatory	Configuration parameter.
callbacks	**Callback**	Optional	Callback method.

Table 5-4 Params

Parameter	Type	Mandatory/Optional	Description
log_path	String	Optional	Path for storing SIP message logs. Absolute path or relative path of the OpenEye installation directory. For example, set this parameter to **C:/log** or **./log**. You can also use **D:\\tup\\log**. If the path does not exist, the system automatically creates one. If an absolute path is used, ensure that each client has a specified drive letter. Therefore, a relative path is recommended.
call	**Call**	Mandatory	Call service.
network	**Network**	Mandatory	Network.
media	**Media**	Mandatory	Media.
audio	**Audio**	Mandatory	Audio.
account	**Account**	Mandatory	Account password type.

Table 5-5 Call

Parameter	Type	Mandatory/Optional	Description
call_ipcall_enable	Number	Mandatory	Enable the IP address-based call function. Set this parameter to **0**.

Table 5-6 NetworkInfo

Parameter	Type	Mandatory/Optional	Description
serverAddr	String	Mandatory	SIP server IP address.
sipServerPort	Number	Mandatory	SIP server port number. The default UDP port number is 5060, and the default TLS port number is 5061.
sipTransportMode	Number	Mandatory	SIP signaling transmission mode. The value **0** indicates UDP, and the value **1** indicates TLS.
httpPort	Number	Mandatory	Generally, the value is **0**.

Table 5-7 Sip

Parameter	**Type**	**Mandatory/Optional**	**Description**
user_type	Number	Mandatory	User terminal type. Set this parameter to **0**.
tls_anonymous_enable	Number	Mandatory	Whether to enable TLS anonymous authentication. Anonymous authentication has security risks. Exercise caution when enabling this function. This function is disabled by default. The options are **0** (no) and **1** (yes).
tls_rootcertpath	String	Optional	Full path of the root certificate. The root certificate must be configured when TLS is used for transmission. For example, set this parameter to **F:/test/cert/root_cert_huawei.pem**.
trans_mode	Number	Mandatory	SIP transmission protocol. ● **0**: UDP (default value) ● **1**: TLS ● **2**: TCP

Table 5-8 Media

Parameter	Type	Mandatory/Optional	Description
trans_mode	Number	Mandatory	Media stream encryption mode. Set this parameter to **1**, indicating that RTP (no encryption) and SRTP (encryption) are supported.

Table 5-9 Audio

Parameter	Type	Mandatory/Optional	Description
audio_codec	String	Mandatory	Audio codec priority and supported audio codec modes, for example, **112,98,18,9,8,0**. ● **112**: OPUS ● **98**: iLBC ● **18**: G729 ● **9**: G722 ● **8**: G711a ● **0**: G711u

Parameter	Type	Mandatory / Optional	Description
dtmf_mode	Number	Optional	Dual-tone multi-frequency (DTMF) mode, that is, transmission mode of the key sound and data. ● **0**: in-band transparent transmission mode (default value) ● **1**: RFC2833 auto-negotiation ● **2**: forcible use of the RFC2833 protocol ● **4**: Info mode ● **5**: H245
audio_anr	Number	Optional	Noise suppression. The value ranges from 0 to 4. The value **0** indicates that noise suppression is disabled. A larger value indicates greater noise suppression strength. By default, this function is disabled.
audio_aec	Number	Optional	Echo cancellation. The value **0** indicates that echo cancellation is disabled, and the value **1** indicates that echo cancellation is enabled. The default value is **0**. It is recommended that this function be enabled.

Parameter	Type	Mandatory/Optional	Description
audio_agc	Number	Optional	Automatic gain. The value **0** indicates that automatic gain is disabled, and the value **1** indicates that automatic gain is enabled. By default, this function is disabled.

Table 5-10 Account

Parameter	Type	Mandatory/Optional	Description
account_pwd_type	Number	Mandatory	Account password type. Set this parameter to **0**.

Table 5-11 Callback

Parameter	Type	Mandatory/Optional	Description
callbacks	function	Optional	Callback method.

Table 5-12 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.

Parameter	Type	Description
local_ip	String	Local IP address. The value is an IPv4 address, for example, **192.168.10.100**.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

Examples

```
function sipBasicCfg() {
    global_cloudIPCC_SDK.tupCall.config({
        networkInfo: {
            serverAddr: "example.com",
            sipServerPort: 5060,
            sipTransportMode: "10.175.1.61",
            httpPort: 5060
        }
    },{response: configResponse});
}

function configResponse(data) {
    if (data.result == 0) {
        console.info("Config Success");
    } else {
        console.error("Config Failed");
        console.error(data);
    }
}
```

5.2 Account Registration and Deactivation

5.2.1 register (Registration)

Interface Description

This interface is invoked to register a SIP account.

Notes

- The WebSocket connection with the OpenEyeCall is set up.
- The registration parameters are set.

Method Definition

```
OpenEyeCall.prototype.register = function(sip_num, sip_name, sip_pwd, sip_mode, callbacks)
```

Parameter Description

Table 5-13 Parameter description

Parameter	Type	Mandatory/Optional	Description
sip_num	String	Mandatory	Subscriber number. The value can contain a maximum of 255 characters. Example: **70942@example.com**
sip_name	String	Mandatory	Username. The value can contain a maximum of 255 characters.
sip_pwd	String	Mandatory	Password. The value can contain a maximum of 255 characters in plaintext.
sip_mode	Int	Mandatory	Networking mode. The options are **4** (UAP networking) and **5** (UAP pool networking).
callbacks	**Callback**	Mandatory	Callback method.

Table 5-14 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	Callback method of the registration result. For details about the input parameters of the callback method, see **Table 5-15**.

Parameter	Type	Mandatory/Optional	Description
onRegStatusUpdate	function	Mandatory	For details about the input parameters of the callback method, see **Table 5-155.2.3.1 onRegStatusUpdate (Reporting the Registration Status)**.
onForceUnReg	function	Mandatory	For details about the input parameters of the callback method, see **Table 5-15**.

Table 5-15 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

If the callback interface is successfully invoked, the registration is not necessarily successful. You need to determine whether the registration is successful based on **5.2.3.1 onRegStatusUpdate (Reporting the Registration Status)**.

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_login",
   "result" : 0,
   "rsp" : 65537
}
```

Examples

```
function register() {
    global_openEye_SDK.openEyeCall.register("70942@example.com", "70942@example.com", "1qaz@WSX",
4, {
        onRegStatusUpdate: onRegStatusUpdate,
        onForceUnReg: onForceUnRegInfo,
        response: registerResponse
    });
}
```

```
function onRegStatusUpdate(data){
    console.info(data);
}
function onForceUnReg(data){
    console.info(data);
}

function registerResponse(data) {
    if (data.result == 0) {
        console.info("Register Operation Success");
    } else {
        console.error("Register Operation Failed");
    }
}
```

5.2.2 deRegister (Deactivation)

Interface Description

This interface is invoked to deactivate a SIP account.

Notes

- The WebSocket connection with the OpenEyeCall is set up.
- The corresponding subscriber is registered.
- You are not in a call.

Method Definition

```
OpenEyeCall.prototype.deRegister = function(sip_num, callbacks)
```

Parameter Description

Table 5-16 Parameter description

Parameter	Type	Mandatory/Optional	Description
sip_num	String	Mandatory	Subscriber number. The value can contain a maximum of 255 characters. Example: **70942@example.com**
callbacks	Callback	Mandatory	Callback method.

Table 5-17 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	Callback method of the registration result. For details about the input parameters of the callback method, see **Table 5-18**.

Table 5-18 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_logout",
   "result" : 0,
   "rsp" : 65538
}
```

Examples

```
function deRegister() {
    global_openEye_SDK.OpenEyeCall.deRegister("70942@example.com", {
        response: deRegisterResponse
    });
}
function deRegisterResponse(data) {
    if (data.result == 0) {
        console.info("DeRegister Success");
    } else {
        console.error("DeRegister Failed");
    }
}
```

5.2.3 Events

5.2.3.1 onRegStatusUpdate (Reporting the Registration Status)

Event Description

After a subscriber initiates a registration request, this event is used to report the registration status to the subscriber twice. For the first time, the subscriber is notified that the current account is being registered. For the second time, the subscriber is notified that the account is registered or not registered.

Event Example

```
{
   "description" : "TSDK_E_LOGIN_EVT_LOGIN_SUCCESS",
   "notify" : 1004,
   "param" : {
      "loginSuccessInfo" : {"confEnvType":0},
      "serviceAccountType" : 4,
      "userId" : 0
   }
}
```

Parameter Description

Table 5-19 Parameter description

Parameter	Type	Description
description	String	Description of the current request.
notify	Number	Internal event ID.
param	**Param**	Event content.

Table 5-20 Param

Parameter	Type	Description
userId	Number	User ID.
serviceAccountType	Number	Account type. ● **4**: UAP ● **5**: UAP pool networking
loginSuccessInfo	Number	Login success information.

5.3 Voice and Video Calls

5.3.1 setBasicCallEvent (Setting Basic Call Events)

Interface Description

Callback function for binding call-related events.

Notes

The registration is complete.

Method Definition

```
OpenEyeCall.prototype.setBasicCallEvent = function(callbacks)
```

Parameter Description

Table 5-21 callbacks parameter description

Parameter	Type	Mandatory/Optional	Description
onCallIncoming	function	Optional	For details about the input parameters of the callback method, see **5.3.3.1 onCallIncoming (Inbound Call Event)**.
onCallOutGoing	function	Optional	For details about the input parameters of the callback method, see **5.3.3.2 onCallOutGoing (Outbound Call Event)**.
onCallRingBack	function	Optional	For details about the input parameters of the callback method, see **5.3.3.3 onCallRingBack (Ringback Event)**.
onCallConnected	function	Optional	For details about the input parameters of the callback method, see **5.3.3.4 onCallConnected (Call Connection Event)**.
onCallEnded	function	Optional	For details about the input parameters of the callback method, see **5.3.3.5 onCallEnded (Call End Event)**.
onCallEndedFailed	function	Optional	For details about the input parameters of the callback method, see **5.3.3.6 onCallEndedFailed (Call End Failure Event)**.

Parameter	Type	Mandatory/Optional	Description
onCallRtpCreated	function	Optional	For details about the input parameters of the callback method, see **5.3.3.7 onCallRtpCreated (RTP Channel Establishment Event)**.

Examples

```
function setBasicCallEvent(){
    global_openEye_SDK.openEyeCall.setBasicCallEvent({
        onCallIncoming: onCallIncoming,
        onCallOutGoing: onCallOutGoing,
        onCallRingBack: onCallRingBack,
        onCallConnected: onCallConnected,
        onCallEnded: onCallEnded,
        onCallEndedFailed: onCallEndedFailed,
        onCallRtpCreated: onCallRtpCreated
    });
}
function onCallIncoming(data){
    console.info(data);
}
function onCallOutGoing(data){
    console.info(data);
}
function onCallRingBack(data){
    console.info(data);
}
function onCallConnected(data){
    console.info(data);
}
function onCallEnded(data){
    console.info(data);
}
function onCallEndedFailed(data){
    console.info(data);
}
function onCallRtpCreated(data){
    console.info(data);
}
```

5.3.2 acceptCall (Answering a Call)

Interface Description

This interface is invoked to answer a call after an inbound call event is received.

Notes

An inbound call event is received. For details, see **5.3.3.1 onCallIncoming (Inbound Call Event)**.

Method Definition

```
OpenEyeCall.prototype.acceptCall = function(callid, is_video_call, callbacks)
```

Parameter Description

Table 5-22 Parameter description

Parameter	Type	Mandatory/Optional	Description
callid	Number	Mandatory	Call ID. The value is reported by the **onCallIncoming** event.
is_video_call	Number	Mandatory	Whether a call is a video call. The value is reported by the **onCallIncoming** event.
callbacks	**Callback**	Mandatory	Callback method.

Table 5-23 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 5-24**.

Table 5-24 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.

Parameter	Type	Description
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_accept_call",
   "result" : 0,
   "rsp" : 67538
}
```

Examples

```
function acceptCall() {
    global_openEye_SDK.openEyeCall.acceptCall(1917517824, 0, { response: onAcceptCallReponse });
}

function onAcceptCallReponse(data) {
    if (data.result == 0) {
        console.error("AcceptCall success. ");
    } else {
        console.error("AcceptCall failed. The ErrorCode is " + data.result);
        console.info(data);
    }
}
```

5.3.3 Events

5.3.3.1 onCallIncoming (Inbound Call Event)

Event Description

When the peer party initiates a call, the local party receives an inbound call event containing the call ID and calling number.

Event Example

```
{
   "description" : "TSDK_E_CALL_EVT_CALL_INCOMING",
   "notify" : 2002,
   "param":
    {
    "callId":1559166976,
    "callInfo":
        {
        "callId":1559166976,
        "callState":1,
        "confId":"",
        "confPasscode":"",
        "isAutoAnswer":0,
        "isCaller":0,
        "isFocus":0,
        "isVideoCall":0,
        "peerDisplayName":"",
        "peerNumber":"444002",
        "reasonCode":0,
        "reasonDescription":"",
        "sipAccountID":0
        },
    "maybeVideoCall":0
    }
}
```

Parameter Description

Table 5-25 Parameter description

Parameter	Type	Description
description	String	Description of the current request.
notify	Number	Internal event ID.
param	**Param**	Event content.

Table 5-26 Param

Parameter	Type	Description
callId	Number	Call ID.
peerNumber	String	Phone number of the peer party.

NOTE

Pay attention to the **callId** and **peerNumber** fields in **Param**.

5.3.3.2 onCallOutGoing (Outbound Call Event)

Event Description

When the local party initiates a call, the local party receives an outbound call event, in which the parameter carries the call ID.

Event Example

```
{
   "description" : "TSDK_E_CALL_EVT_CALL_OUTGOING",
   "notify" : 2003,
   "param":
    {
    "callId":1867907072,
    "callInfo":
        {
        "callId":1867907072,
        "callState":2,
        "confId":"",
        "confPasscode":"",
        "isAutoAnswer":0,
        "isCaller":1,
        "isFocus":0,
        "isVideoCall":0,
        "peerDisplayName":"",
        "peerNumber":"444002",
        "reasonCode":0,
        "reasonDescription":"",
        "sipAccountID":0
        }
    }
}
```

Parameter Description

Table 5-27 Parameter description

Parameter	Type	Description
description	String	Description of the current request.
notify	Number	Internal event ID.
param	**Param**	Event content.

Table 5-28 Param

Parameter	Type	Description
callId	Number	Call ID.
peerNumber	String	Phone number of the peer party.

NOTE

Pay attention to the **callId** and **peerNumber** fields in **Param**.

5.3.3.3 onCallRingBack (Ringback Event)

Event Description

After the local party initiates a call, the local party receives the ringback event, in which the parameter carries the call ID.

Event Example

```
{
   "description" : "TSDK_E_CALL_EVT_CALL_RINGBACK",
   "notify" : 2004,
   "param" : {
      "callId" : 1867907072
   }
}
```

Parameter Description

Table 5-29 Parameter description

Parameter	Type	Description
description	String	Description of the current request.
notify	Number	Internal event ID.
param	**Param**	Event content.

Table 5-30 Param

Parameter	Type	Description
callId	Number	Call ID.

NOTE

Pay attention to the **callId** field in **Param**.

5.3.3.4 onCallConnected (Call Connection Event)

Event Description

The call is connected.

Event Example

```
{
   "description" : "TSDK_E_CALL_EVT_CALL_CONNECTED",
   "notify" : 2006,
   "param":
    {
    "callId":1559166976,
    "callInfo":
        {
        "callId":1559166976,
        "callState":3,
        "confId":"",
        "confPasscode":"",
        "isAutoAnswer":0,
        "isCaller":0,
        "isFocus":0,
        "isVideoCall":0,
        "peerDisplayName":"",
        "peerNumber":"444002",
        "reasonCode":0,
        "reasonDescription":"",
        "sipAccountID":0
        }
    }
}
```

Parameter Description

Table 5-31 Parameter description

Parameter	Type	Description
description	String	Description of the current request.
notify	Number	Internal event ID.
param	**Param**	Event content.

Table 5-32 Param

Parameter	Type	Description
callId	Number	Call ID.
peerNumber	String	Phone number of the peer party.

NOTE

Pay attention to the **callId** and **peerNumber** fields in **Param**.

5.3.3.5 onCallEnded (Call End Event)

Event Description

This event is triggered when one party hangs up the call during or after the call is set up.

Event Example

```
{
   "description" : "TSDK_E_CALL_EVT_CALL_ENDED",
   "notify" : 2007,
   "param":
    {
    "callId":1559166976,
    "callInfo":
        {
        "callId":1559166976,
        "callState":5,
        "confId":"",
        "confPasscode":"",
        "isAutoAnswer":0,
        "isCaller":0,
        "isFocus":0,
        "isVideoCall":0,
        "peerDisplayName":"",
        "peerNumber":"444002",
        "reasonCode":0,
        "reasonDescription":"",
        "sipAccountID":0
        }
    }
}
```

Parameter Description

Table 5-33 Parameter description

Parameter	Type	Description
description	String	Description of the current request.
notify	Number	Internal event ID.
param	**Param**	Event content.

Table 5-34 Param

Parameter	Type	Description
callId	Number	Call ID.
peerNumber	String	Phone number of the peer party.

NOTE

Pay attention to the **callId** and **peerNumber** fields in **Param**.

5.3.3.6 onCallEndedFailed (Call End Failure Event)

Event Description

When a call is ended, if the call corresponding to the input call ID does not exist, a call end failure event is returned.

Event Example

```
{
   "description" : "TSDK_E_CALL_EVT_ENDCALL_FAILED",
   "notify" : 2022,
   "param" : {
      "callId" : 0,
      "result" : 50331670
   }
}
```

Parameter Description

Table 5-35 Parameter description

Parameter	Type	Description
description	String	Description of the current request.
notify	Number	Internal event ID.
param	**Param**	Event content.

Table 5-36 Param

Parameter	Type	Description
callId	Number	Call ID.
result	Number	Failure cause code.

5.3.3.7 onCallRtpCreated (RTP Channel Establishment Event)

Event Description

This event is triggered when a media channel is established after a call is connected.

Event Example

```
{
  "description" : "TSDK_E_CALL_EVT_CALL_RTP_CREATED",
  "notify" : 2005,
  "param" : {
    "callId" : 1867907072
  }
}
```

Parameter Description

Table 5-37 Parameter description

Parameter	Type	Description
description	String	Description of the current request.
notify	Number	Internal event ID.
param	**Param**	Event content.

Table 5-38 Param

Parameter	Type	Description
callId	Number	Call ID.

6 Voice and Video Call Interface Extension

6.1 Voice and Video Calls

6.1.1 startCall (Initiating a Call)

Interface Description

This interface is invoked to initiate a VoIP call.

Notes

- An account has been registered.
- Basic call events are configured.

Method Definition

```
OpenEyeCall.prototype.startCall = function(callee_num, is_video_call, callbacks)
```

Parameter Description

Table 6-1 Parameter description

Parameter	Type	Mandatory/Optional	Description
callee_num	String	Mandatory	Called number. The value can contain a maximum of 255 characters.
is_video_call	Boolean	Mandatory	Call type. Set this parameter to **false**.
callbacks	**Callback**	Mandatory	Callback method.

Table 6-2 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-3**.

Table 6-3 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.
param	**Param**	Call information.

Table 6-4 Param parameter description

Parameter	Type	Description
call_id	Number	Call ID. The OpenEye automatically fills the call ID in the parameters of the callback function.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_start_call",
   "param" : {
      "callId" : 1541472256
   },
   "result" : 0,
   "rsp" : 67537
}
```

Examples

```
var tupCurrentCallId;
function startCall() {
    global_openEye_SDK.openEyeCall.openEyeCall("70943", false, {
        response: startCallResponse
    });
}

function startCallResponse(data) {
    if (data.result == 0) {
        console.info("StartCall success. ");
        tupCurrentCallId = data.param.call_id;
    } else {
        console.error("StartCall failed. The ErrorCode is " + data.result);
        console.info(data);
    }
}
```

6.1.2 endCall (Ending a Call)

Interface Description

This interface is invoked to end a call with or an inbound call from another subscriber.

Notes

There is a call with or an inbound call from another subscriber.

Method Definition

```
OpenEyeCall.prototype.endCall = function(callid, callbacks)
```

Parameter Description

Table 6-5 Parameter description

Parameter	Type	Mandatory/Optional	Description
callid	Number	Mandatory	Call ID.
callbacks	**Callback**	Mandatory	Callback method.

Table 6-6 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-7**.

Table 6-7 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_end_call",
   "result" : 0,
   "rsp" : 67539
}
```

Examples

```
function endCall() {
   global_openEye_SDK.openEyeCall.endCall(1755709440, { response: onEndCallReponse });
}

function onEndCallReponse(data) {
    if (data.result == 0) {
        console.error("EndCall success. ");
    } else {
        console.error("EndCall failed. The ErrorCode is " + data.result);
        console.info(data);
    }
}
```

6.1.3 operateMic (Muting the Microphone)

Interface Description

This interface is invoked to mute or unmute the microphone.

Notes

A call is set up.

Method Definition

```
OpenEyeCall.prototype.operateMic = function(callid, to_mute, callbacks)
```

Parameter Description

Table 6-8 Parameter description

Parameter	Type	Mandatory/Optional	Description
callid	Number	Mandatory	Call ID.
to_mute	Number	Mandatory	Whether to mute the microphone. The value **1** indicates that the call is muted, and the value **0** indicates that the call is resumed.
callbacks	**Callback**	Optional	Callback method.

Table 6-9 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-10**.

Table 6-10 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_mute_mic",
   "result" : 0,
   "rsp" : 67547
}
```

Examples

```
/**
* Muting/Unmuting
* @param {*} flag 1: mute; 0: unmute
 */
function operateMic(flag) {
     global_openEye_SDK.openEyeCall.operateMic(1755709440, flag, { response: onOperateMicResponse });
}

function onOperateMicResponse(data) {
    if (data.result == 0) {
        console.info("OperateMic success. ");
    } else {
        console.error("OperateMic failed. The ErrorCode is " + data.result);
        console.info(data);
    }
}
```

6.1.4 DTMF (Two-Stage Dialing)

Interface Description

This interface is invoked to send two-stage dialing information during a call.

Notes

The two-stage dialing information can be sent only during a call.

Method Definition

```
OpenEyeCall.prototype.dtmf = function(callid, keyTone, callbacks)
```

Parameter Description

Table 6-11 Parameter description

Parameter	Type	Mandatory/Optional	Description
callid	Number	Mandatory	Call ID.
keyTone	Number	Mandatory	DTMF key value. The value ranges from 0 to 16.
callbacks	**Callback**	Optional	Callback method.

Table 6-12 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-13**.

Table 6-13 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_send_dtmf",
   "result" : 0,
   "rsp" : 67540
}
```

Examples

```
function dtmf() {
    global_openEye_SDK.openEyeCall.dtmf(1755709440, 12, {
        response: dtmfResponse
    });
}

function dtmfResponse(data) {
    if (data.result == 0) {
        console.info("Dtmf success. ");
    } else {
        console.error("Dtmf failed. The ErrorCode is " + data.result);
        console.info(data);
    }
}
```

6.1.5 Screenshot (Screenshot)

Interface Description

This interface is invoked to capture and save a frame of the peer video.

Notes

Prerequisites: The WebSocket connection with the OpenEyeCall has been set up, and a video call is in progress.

Method Definition

```
OpenEyeCall.prototype.screenShot = function(callbacks)
```

Parameter Description

Table 6-14 Parameter description

Parameter	Type	Mandatory/Optional	Description
callbacks	**Callback**	Mandatory	Callback method.

Table 6-15 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-16**.

Table 6-16 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_share_evt_stopsharewindow",
   "result" : 0,
   "rsp" : 67762
}
```

Examples

```
function startScreenShot(){
 console.info("startScreenShot");
 this.global_openEye_SDK.openEyeCall.screenShot({ response: startScreenShotResponse })
}

function startScreenShotResponse(data){
 console.log(data);
 if (data.result == 0) {
        console.info("startScreenShot Success");
    } else {
        console.error("startScreenShot failed");
        console.error(data);
    }
}
```

6.1.6 catchVideo (Screen Recording)

Interface Description

This interface is invoked to start or end screen recording.

Notes

Prerequisites: The WebSocket connection with the OpenEyeCall has been set up, and a video call is in progress.

Method Definition

```
OpenEyeCall.prototype.videoCatch = function(value, callbacks)
```

Parameter Description

Table 6-17 Parameter description

Parameter	Type	Mandatory/Optional	Description
value	bool	Mandatory	The value **true** indicates that screen recording is started, and the value **false** indicates that screen recording is stopped.
callbacks	**Callback**	Mandatory	Callback method.

Table 6-18 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-19**.

Table 6-19 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_share_evt_stopsharewindow",
   "result" : 0,
   "rsp" : 67763
}
```

Examples

```
function catchVideo(value){
 this.global_openEye_SDK.openEyeCall.videoCatch(value, { response: startVideoCatchResponse })
}

function startVideoCatchResponse(data){
 console.log(data);
 if (data.result == 0) {
       console.info("startVideoCatch Success");
   } else {
       console.error("startVideoCatch failed");
       console.error(data);
   }
}
```

6.1.7 setAnswerWay (Setting the Answering Mode)

Interface Description

This interface is invoked to set the answering mode (automatic hang-up, automatic answering, or do-no-disturb).

Notes

The WebSocket connection with the OpenEye is set up.

Method Definition

```
OpenEyeCall.prototype.setAnswerWay = function(answerWay, times, callbacks)
```

Parameter Description

Table 6-20 Parameter description

Parameter	Type	Mandatory/Optional	Description
answerWay	String	Mandatory	Answering mode. The options are **1** (automatic hang-up), **2** (automatic answering), and **4** (do-no-disturb).
times	Number	Mandatory	Interval for the answering mode to take effect. (For do-not-disturb, this parameter is invalid and optional.)
callbacks	**Callback**	Mandatory	Callback method.

Table 6-21 Callback

Parameter	Type	Mandatory/ Optional	Description
response	Function	Mandatory	For details about the input parameters of the callback method, see **Table 6-22**.

Table 6-22 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.
errMsg	String	Error information.

The following is an example of input parameters of the callback method:

```
{
    description: "SetAnswerWay",
    result: 0,
    rsp: 67767,
    errMsg: ""
}
```

Examples

```
function setAnswerWay(answerWay, times){
 this.global_openEye_SDK.openEyeCall.setAnswerWay(answerWay, times, { response:
onSetAnswerWayResponse });
}

function onSetAnswerWayResponse(data){
 console.log(data);
 if (data.result == 0) {
        console.info("SetAnswerWay Success");
    } else {
        console.error("SetAnswerWay failed. The ErrorCode is " + data.result + ";errMsg:" + data.errMsg);
        alert("SetAnswerWay failed.  The ErrorCode is " + data.result + ";errMsg:" + data.errMsg);
    }
}
```

6.2 Device Management

6.2.1 getMediaDevices (Obtaining the Device List)

Interface Description

This interface is invoked to obtain the local device list, such as the microphone and speaker.

Notes

This interface can be invoked only after an account is successfully registered.

Method Definition

```
OpenEyeCall.prototype.getMediaDevices = function(type, callbacks)
```

Parameter Description

Table 6-23 Parameter description

Parameter	Type	Mandatory/Optional	Description
type	Number	Mandatory	Device type. The value **0** indicates a microphone, and the value **1** indicates a speaker.
callbacks	**Callback**	Optional	Callback method.

Table 6-24 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-25**.

Table 6-25 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.
param	**Param**	Device information.

Table 6-26 Param

Parameter	Type	Description
array	Device (**Table 6-27**) array	Device list.
device_num	Number	Number of devices.

Table 6-27 Devices

Parameter	Type	Description
camera_orient	Number	Reserved field.
index	Number	Serial number of a device.
name	String	Device name.

NOTE

The following is an example of input parameters of the callback method:

```
{
  "description" : "tsdk_get_devices",
  "param" : {
    "deviceInfo" : [
      {
           "cameraOrient":0,
      "deviceId":0,
      "deviceName":
      "default: Speaker (Huawei HDP Audio Driver)",
      "index":0

      },
      {
           "cameraOrient":0,
      "deviceId":0,
      "deviceName":"Speaker (Huawei HDP Audio Driver)"
      "index":1
      }
    ],
    "deviceType": 1,
    "num" : 2
  },
  "result" : 0,
  "rsp" : 67550
}
```

Examples

```
function getMediaDevices() {
    global_openEye_SDK.openEyeCall.getMediaDevices(1, {
        response: getMediaDevicesResponse
    });
}

function getMediaDevicesResponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("GetMediaDevices success");
    } else {
        console.error("GetMediaDevices failed");
    }
}
```

6.2.2 setMicIndex (Setting the Microphone)

Interface Description

This interface is invoked to set the microphone for calls. If this parameter is not set, the OpenEye uses the default microphone.

Notes

- The WebSocket connection with the OpenEye is set up.
- Generally, the device serial number is obtained using getMediaDevice (**6.2.1 getMediaDevices (Obtaining the Device List)**) after the system is initialized.

Method Definition

```
OpenEyeCall.prototype.setMicIndex = function(idx, callbacks)
```

Parameter Description

Table 6-28 Parameter description

Parameter	Type	Mandatory/Optional	Description
idx	Number	Mandatory	Serial number of the microphone.
callbacks	Callback	Optional	Callback method.

Table 6-29 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-30**.

Table 6-30 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_set_mic_index",
   "result" : 0,
   "rsp" : 67551
}
```

Examples

```
function setMicIndex() {
    global_openEye_SDK.openEyeCall.setMicIndex(1, {
        response: setMicIndexResponse
    });
}

function setMicIndexResponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("SetMicIndex success")
    } else {
        console.error("StMicIndex failed");
    }
}
```

6.2.3 mediaGetMicIndex (Querying the Microphone in Use)

Interface Description

This interface is invoked to query the microphone in use and return the serial number of the microphone.

Notes

- The WebSocket connection with the OpenEye is set up.
- This interface is invoked for interface test or product commissioning. It will not be invoked in actual business scenarios.

Method Definition

```
OpenEyeCall.prototype.mediaGetMicIndex = function(callbacks)
```

Parameter Description

Table 6-31 Parameter description

Parameter	Type	Mandatory/Optional	Description
callbacks	**Callback**	Optional	Callback method.

Table 6-32 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-33**.

Table 6-33 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.
param	**Param**	Current microphone.

Table 6-34 Param

Parameter	Type	Description
index	Number	Serial number of the current microphone.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_get_mic_index",
   "param" : {
      "index" : 0
   },
   "result" : 0,
   "rsp" : 67552
}
```

Examples

```
function mediaGetMicIndex() {
    global_openEye_SDK.openEyeCall.mediaGetMicIndex({ response: mediaGetMicIndexResponse });
}
```

```
function mediaGetMicIndexResponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("MediaGetMicIndex success")
    } else {
        console.error("MediaGetMicIndex failed");
    }
}
```

6.2.4 setSpeakIndex (Setting the Speaker)

Interface Description

This interface is invoked to set the speaker for calls.

Notes

- The WebSocket connection with the OpenEye is set up.
- Generally, the device serial number is obtained using getMediaDevice (**6.2.1 getMediaDevices (Obtaining the Device List)**) after the system is initialized.

Method Definition

```
OpenEyeCall.prototype.setSpeakIndex = function(idx, callbacks)
```

Parameter Description

Table 6-35 Parameter description

Parameter	Type	Mandatory/Optional	Description
idx	Number	Mandatory	Serial number of the speaker.
callbacks	**Callback**	Optional	Callback method.

Table 6-36 Callback

Parameter	Type	Mandatory/ Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-37**.

Table 6-37 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_set_speak_index",
   "result" : 0,
   "rsp" : 67553
}
```

Examples

```
function setSpeakIndex() {
    global_openEye_SDK.openEyeCall.setSpeakIndex(1, {
        response: setSpeakIndexResponse
    });
}

function setSpeakIndexResponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("SetSpeakIndex success")
    } else {
        console.error("SetSpeakIndex failed");
    }
}
```

6.2.5 mediaGetSpeakIndex (Querying the Speaker in Use)

Interface Description

This interface is invoked to query the speaker in use and return the serial number of the speaker.

Notes

- The WebSocket connection with the OpenEye is set up.
- This interface is invoked for interface test or product commissioning. It will not be invoked in actual business scenarios.

Method Definition

```
OpenEyeCall.prototype.mediaGetSpeakIndex = function(callbacks)
```

Parameter Description

Table 6-38 Parameter description

Parameter	Type	Mandatory/Optional	Description
callbacks	**Callback**	Optional	Callback method.

Table 6-39 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-40**.

Table 6-40 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.

Parameter	Type	Description
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.
param	**Param**	Current microphone.

Table 6-41 Param

Parameter	Type	Description
index	Number	Serial number of the current microphone.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_get_speak_index",
   "param" : {
      "index" : 0
   },
   "result" : 0,
   "rsp" : 67554
}
```

Examples

```
function mediaGetSpeakIndex() {
    global_openEye_SDK.openEyeCall.mediaGetSpeakIndex({ response: mediaGetSpeakIndexResponse });
}

function mediaGetSpeakIndexResponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("MediaGetSpeakIndex success");
    } else {
        console.error("MediaGetSpeakIndex failed");
    }
}
```

6.2.6 setMicVol (Setting the Microphone Volume)

Interface Description

This interface is invoked to set the microphone volume.

Notes

The WebSocket connection with the OpenEye is set up.

Method Definition

```
OpenEyeCall.prototype.setMicVol = function(volume, device, callbacks)
```

Parameter Description

Table 6-42 Parameter description

Parameter	Type	Mandatory/Optional	Description
volume	Number	Mandatory	Volume. The value ranges from 0 to 100.
device	Number	Mandatory	Device type. Set this parameter to **1**.
callbacks	**Callback**	Optional	Callback method.

Table 6-43 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-44**.

Table 6-44 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_set_mic_volume",
   "result" : 0,
   "rsp" : 67577
}
```

Examples

```
 function setMicVol() {
    global_openEye_SDK.openEyeCall.setMicVol(20, 1, {
        response: setMicVolReponse
    });
}

function setMicVolReponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("SetMicVol Success.")
    } else {
        console.error("SetMicVol failed.");
    }
}
```

6.2.7 getMicVol (Querying the Current Microphone Volume)

Interface Description

This interface is invoked to query the current volume of the microphone.

Notes

The WebSocket connection with the OpenEye is set up.

Method Definition

```
OpenEyeCall.prototype.getMicVol = function(callbacks)
```

Parameter Description

Table 6-45 Parameter description

Parameter	Type	Mandatory/Optional	Description
callbacks	**Callback**	Optional	Callback method.

Table 6-46 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-47**.

Table 6-47 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.
param	**Param**	Volume information.

Table 6-48 Param

Parameter	Type	Description
volume	Number	Current microphone volume.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_get_mic_volume",
   "param" : {
      "volume" : 20
   },
   "result" : 0,
   "rsp" : 67578
}
```

Examples

```
function getMicVol() {
    global_openEye_SDK.openEyeCall.getMicVol({
        response: getMicVolResponse
    });
}
```

```
function getMicVolResponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("GetMicVol Success.");
    } else {
        console.error("GetMicVol failed.");
    }
}
```

6.2.8 setSpkVol (Setting the Speaker Volume)

Interface Description

This interface is invoked to set the speaker volume.

Notes

The WebSocket connection with the OpenEye is set up.

Method Definition

```
OpenEyeCall.prototype.setSpkVol = function(volume, device, callbacks)
```

Parameter Description

Table 6-49 Parameter description

Parameter	Type	Mandatory/Optional	Description
volume	Number	Mandatory	Volume. The value ranges from 0 to 100.
device	Number	Mandatory	Device type. Set this parameter to **0**.
callbacks	**Callback**	Optional	Callback method.

Table 6-50 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-51**.

Table 6-51 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_set_speak_volume",
   "result" : 0,
   "rsp" : 67557
}
```

Examples

```
function setSpkVol() {
    global_openEye_SDK.openEyeCall.setSpkVol(80, 0, {
        response: setSpkVolResponse
    });
}

function setSpkVolResponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("SetSpkVol Success.")
    } else {
        console.error("SetSpkVol failed.");
    }
}
```

6.2.9 getSpkVol (Querying the Current Speaker Volume)

Interface Description

This interface is invoked to query the current volume of the speaker.

Notes

The WebSocket connection with the OpenEye is set up.

Method Definition

```
OpenEyeCall.prototype.getSpkVol = function(callbacks)
```

Parameter Description

Table 6-52 Parameter description

Parameter	Type	Mandatory/Optional	Description
callbacks	**Callback**	Optional	Callback method.

Table 6-53 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-54**.

Table 6-54 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.

Parameter	Type	Description
result	Number	Query result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.
param	**Param**	Volume information.

Table 6-55 Param

Parameter	Type	Description
volume	Number	Current output volume.

NOTE

The following is an example of input parameters of the callback method:

```
{
  "description" : "tsdk_get_speak_volume",
  "param" : {
    "volume" : 80
  },
  "result" : 0,
  "rsp" : 67558
}
```

Examples

```
function getSpkVol() {
    global_openEye_SDK.openEyeCall.getSpkVol({
        response: getSpkVolResponse
    });
}

function getSpkVolResponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("GetSpkVol Success.");
    } else {
        console.error("GetSpkVol failed");
    }
}
```

6.2.10 setVideoWindowParam (Setting the Position, Width, and Height of the Video Window)

Interface Description

This interface is invoked to set video image parameters during a video call, including the position, width, and height (in pixels).

Notes

The WebSocket connection with the OpenEye is set up.

Note that the recommended width and height of the video image are 720 pixels and 480 pixels, respectively. The minimum width and height that can be displayed on the UI are 480 pixels and 360 pixels, respectively. If the values are less than 480 pixels and 360 pixels, only the video image is displayed, and the operation control UI is not displayed. You are not advised to set the width and height to values less than 480 pixels and 360 pixels.

Before answering a video call, invoke this interface to preset the video window information. After this interface is invoked to set the video window information, the video window information is valid for a long time. The video window information is changed only when you invoke this interface again to modify the video window information or close the entire page.

Method Definition

```
OpenEyeCall.prototype.setVideoWindowParam= function(posX,posY,width,height,callbacks)
```

Parameter Description

Table 6-56 Parameter description

Parameter	Type	Mandatory/ Optional	Description
posX	Number	Mandatory	X coordinate at the upper left corner of the video window. The value must be greater than 0.
posY	Number	Mandatory	Y coordinate at the upper left corner of the video window. The value must be greater than 0.
width	Number	Mandatory	Video window width.
height	Number	Mandatory	Video window height.
callbacks	**Callback**	Optional	Callback method.

Table 6-57 Callback

Parameter	Type	Mandatory/ Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-58**.

Table 6-58 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Configuration result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_set_video_rect",
   "result" : 0,
   "rsp" : 67745
}
```

Examples

```
function setVideoWindowParam() {
this.global_openEye_SDK.openEyeCall.setVideoWindowParam(20,30,720,480, { response:
setVideoWindowParamResponse })
}

function setVideoWindowParamResponse(data) {
    if (data.result == 0) {
        console.info("setVideoWindowParam Success");
    } else {
        console.error("setVideoWindowParam failed");
    }
}
```

6.2.11 setVideoLayoutMode (Setting the Video Window Layout Mode)

Interface Description

This interface is invoked to set the video layout mode during a video call. Two video layout modes are available: side-by-side and picture-in-picture.

Notes

The WebSocket connection with the OpenEye is set up.

Method Definition

```
OpenEyeCall.prototype.setVideoLayoutMode = function(layoutMode,callbacks)
```

Parameter Description

Table 6-59 Parameter description

Parameter	Type	Mandatory/Optional	Description
layoutMode	Number	Mandatory	Video mode. The value **0** indicates the picture-in-picture mode and the value **1** indicates the side-by-side mode.
callbacks	**Callback**	Optional	Callback method.

Table 6-60 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-61**.

Table 6-61 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Configuration result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_set_video_layout_mode",
   "result" : 0,
   "rsp" : 67749
}
```

Examples

```
function setVideoWindowParam() {
this.global_openEye_SDK.openEyeCall.setVideoLayoutMode(param, { response:
setVideoLayoutModeResponse })
}
function setVideoLayoutModeResponse(data) {
    if (data.result == 0) {
        console.info("setVideoLayoutMode Success");
    } else {
        console.error("setVideoLayoutMode failed");
    }
}
```

6.2.12 setVideoDisplayMode (Setting the Image Cropping Mode in the Video Window)

Interface Description

This interface is invoked to set the video image cropping parameters during a video call.

Notes

The WebSocket connection with the OpenEye is set up.

Method Definition

```
OpenEyeCall.prototype.setVideoDisplayMode = function(displayMode,callbacks)
```

Parameter Description

Table 6-62 Parameter description

Parameter	Type	Mandatory/Optional	Description
displayMode	Number	Mandatory	Cropping mode of the video image. The value **1** indicates the cropping mode (not stretched) and the value **2** indicates the stretching mode (not stretched).
callbacks	**Callback**	Optional	Callback method.

Table 6-63 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-64**.

Table 6-64 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Configuration result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_set_video_display_mode",
   "result" : 0,
   "rsp" : 67751
}
```

Examples

```
function setVideoWindowParam() {
this.global_openEye_SDK.openEyeCall.setVideoDisplayMode(param, { response:
setVideoDisplayModeResponse })
}
function setVideoDisplayModeResponse(data) {
    if (data.result == 0) {
        console.info("setVideoDisplayMode Success");
    } else {
        console.error("setVideoDisplayMode failed");
    }
}
```

6.2.13 openCamera (Turning on the Camera)

Interface Description

This interface is invoked to turn on the local camera.

Notes

The WebSocket connection with the OpenEye is set up.

Method Definition

```
OpenEyeCall.prototype.openCamera = function(callId, callbacks)
```

Parameter Description

Table 6-65 Parameter description

Parameter	Type	Mandatory/Optional	Description
callId	Number	Mandatory	Call ID of the current call. If no ongoing call is available, this parameter must be set to **–1**.
callbacks	**Callback**	Optional	Callback method.

Table 6-66 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-67**.

Table 6-67 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Configuration result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_control_camera",
   "result" : 0,
   "rsp" : 67759
}
```

Examples

```
function switchCameraMode() {
 var ischecked = document.getElementById("camera-control-toggle-button").checked;
 if (tupCurrentCallId == "") {
  tupCurrentCallId = -1;
 }
 if (ischecked) {
  console.info("switchCameraMode ischecked true.CallId is:"+tupCurrentCallId);
  this.global_openEye_SDK.openEyeCall.openCamera(tupCurrentCallId, {
   response: cameraModeResponse});
 } else {
  console.info("switchCameraMode ischecked false.CallId is:"+tupCurrentCallId);
  this.global_openEye_SDK.openEyeCall.closeCamera(tupCurrentCallId, {
   response: cameraModeResponse});
 }
}

function cameraModeResponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("controlVideo Success.");
    } else {
        console.error("controlVideo failed.");
    }
}
```

6.2.14 closeCamera (Turning off the Camera)

Interface Description

This interface is invoked to turn off the local camera.

Notes

The WebSocket connection with the OpenEye is set up.

Method Definition

```
OpenEyeCall.prototype.closeCamera = function(callId, callbacks)
```

Parameter Description

Table 6-68 Parameter description

Parameter	Type	Mandatory/Optional	Description
callId	Number	Mandatory	Call ID of the current call. If no ongoing call is available, this parameter must be set to **–1**.
callbacks	**Callback**	Optional	Callback method.

Table 6-69 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-70**.

Table 6-70 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Configuration result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_control_camera",
   "result" : 0,
   "rsp" : 67759
}
```

Examples

```
function switchCameraMode() {
 var ischecked = document.getElementById("camera-control-toggle-button").checked;
 if (tupCurrentCallId == "") {
  tupCurrentCallId = -1;
 }
 if (ischecked) {
  console.info("switchCameraMode ischecked true.CallId is:"+tupCurrentCallId);
  this.global_openEye_SDK.openEyeCall.openCamera(tupCurrentCallId, {
   response: cameraModeResponse});
 } else {
  console.info("switchCameraMode ischecked false.CallId is:"+tupCurrentCallId);
  this.global_openEye_SDK.openEyeCall.closeCamera(tupCurrentCallId, {
   response: cameraModeResponse});
 }
}

function cameraModeResponse(data) {
    console.info(data);
    if (data.result == 0) {
        console.info("controlVideo Success.");
    } else {
        console.error("controlVideo failed.");
    }
}
```

6.3 Screen Sharing

This interface is invoked to share the local desktop with the peer party. The desktop sharing function includes desktop sharing, specified area sharing, and specified application window sharing. Note that the screen sharing function is available only when a video call is in progress.

6.3.1 Obtaining the List of Shareable Applications

Interface Description

This interface is invoked to obtain the list of application windows that can be shared in the current operating system.

Notes

Prerequisites: The WebSocket connection with the OpenEye has been set up, and a video call is in progress.

Method Definition

```
OpenEyeCall.prototype.getAppList = function(callbacks)
```

Parameter Description

Table 6-71 Parameter description

Parameter	Type	Mandatory/Optional	Description
callbacks	**Callback**	Mandatory	Callback method.

Table 6-72 Callback

Parameter	Type	Mandatory/Optional	Description
response	Function	Mandatory	For details about the input parameters of the callback method, see **Table 6-73**.

Table 6-73 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.

Parameter	Type	Description
result	Number	Configuration result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.
param	key:value	Key-value pair of the window handle and window name.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_share_evt_getapplist",
   "result" : 0,
   "rsp" : 67753
   "param":{
     65552: "Desktop"
     132070: "app1"
     132974: "app2"
     198240: "app3"
     328180: "app4"
     329712: "app5"
   }
}
```

Note: Numbers like 65552 are corresponding window handles in the window system. The first parameter in the input parameters of **6.3.2 Setting the Information About the Applications to Be Shared** is the value.

Examples

```
function getAppList(){
    this.global_openEye_SDK.openEyeCall.getAppList({ response: getAppListResponse })
}
function getAppListResponse(data) {
    console.log(data);
    if (data.result == 0) {
        console.info("getAppListResponse success");
        document.getElementById("shareAppList").innerHTML = "";
        for (var key in data.param) {
            var item = data.param[key];
            document.getElementById("shareAppList").options.add(new Option(key + "_" + item, key));
        }
    } else {
        console.error("getAppListResponse failed");
    }
}
```

6.3.2 Setting the Information About the Applications to Be Shared

Interface Description

This interface is invoked to set the information about the window to be shared.

Notes

Prerequisites: The WebSocket connection with the OpenEyeCall has been set up, and a video call is in progress.

Method Definition

```
OpenEyeCall.prototype.setShareWindow = function(hwnd,callbacks)
```

Parameter Description

Table 6-74 Parameter description

Parameter	Type	Mandatory/Optional	Description
hwnd	int	Mandatory	Handle of the window to be shared. For details, see the interface description in **6.3.1 Obtaining the List of Shareable Applications**.
callbacks	**Callback**	Mandatory	Callback method.

Table 6-75 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-76**.

Table 6-76 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.

Parameter	Type	Description
result	Number	Configuration result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_share_evt_setshareapp",
   "result" : 0,
   "rsp" : 67754
}
```

Examples

```
function setShareApp(){
    this.global_openEye_SDK.openEyeCall.setShareWindow(document.getElementById("shareAppList").value,
{ response: setShareAppRespone })
}
//step2 callback. Set callback of the window to be shared.
function setShareAppRespone(data){
    console.log(data);
    if (data.result == 0) {
        console.info("setShareApp Success");
    } else {
        console.error("setShareApp failed");
    }
}
```

6.3.3 Starting Sharing

Interface Description

This interface is invoked by the initiator to proactively start window sharing.

Notes

Prerequisites: The WebSocket connection with the OpenEyeCall has been set up, and a video call is in progress.

Method Definition

```
OpenEyeCall.prototype.startShareWindow = function(callbacks)
```

Parameter Description

Table 6-77 Parameter description

Parameter	Type	Mandatory/Optional	Description
callbacks	**Callback**	Mandatory	Callback method.

Table 6-78 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-79**.

Table 6-79 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Configuration result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_share_evt_startsharewindow",
   "result" : 0,
   "rsp" : 67755
}
```

Examples

```
function startShare(){
    this.global_openEye_SDK.openEyeCall.startShareWindow({ response: startShareRespone })
}
function startShareRespone(data){
    console.log(data);
    if (data.result == 0) {
        console.info("startShare Success");
    } else {
        console.error("startShare failed");
    }
}
```

6.3.4 Stopping Sharing

Interface Description

This interface is invoked by the initiator to proactively stop sharing.

Notes

Prerequisites: The WebSocket connection with the OpenEyeCall has been set up, and a video call is in progress.

Method Definition

```
OpenEyeCall.prototype.stopShareWindow = function(callbacks)
```

Parameter Description

Table 6-80 Parameter description

Parameter	Type	Mandatory/Optional	Description
callbacks	**Callback**	Mandatory	Callback method.

Table 6-81 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-82**.

Table 6-82 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Configuration result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_share_evt_stopsharewindow",
   "result" : 0,
   "rsp" : 67756
}
```

Examples

```
function stopShare(){
    this.global_openEye_SDK.openEyeCall.stopShareWindow({ response: stopShareRespone })
}
function stopShareRespone(data){
    console.log(data);
    if (data.result == 0) {
        console.info("stopShare Success");
    } else {
        console.error("stopShare failed");
    }
}
```

6.3.5 Enabling the Sharing Function

Interface Description

This interface is invoked to enable the sharing function.

Notes

Prerequisites: The WebSocket connection with the OpenEyeCall has been established.

Method Definition

```
OpenEyeCall.prototype.shareControl = function(value, callbacks)
```

Parameter Description

Table 6-83 Parameter description

Parameter	Type	Mandatory/Optional	Description
value	bool	Mandatory	Whether to enable the sharing function.
callbacks	Callback	Mandatory	Callback method.

Table 6-84 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	For details about the input parameters of the callback method, see **Table 6-85**.

Table 6-85 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
result	Number	Configuration result. The value **0** indicates success and other values indicate failure.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "tsdk_share_evt_stopsharewindow",
   "result" : 0,
   "rsp" : 67760
}
```

Examples

```
function switchShare() {
 var ischecked = document.getElementById("share-control-toggle-button").checked;
 if (ischecked) {
  console.info("switchShare ischecked true.");
  shareSwitch = true;
  document.getElementById("shareControlDiv").style.visibility = "visible";
  this.global_openEye_SDK.openEyeCall.shareControl(shareSwitch, { response: switchShareRespone })
 } else {
  console.info("switchShare ischecked false.");
  shareSwitch = false;
  document.getElementById("shareControlDiv").style.visibility = "hidden";
  this.global_openEye_SDK.openEyeCall.shareControl(shareSwitch, { response: switchShareRespone })
 }
}

function switchShareRespone(data){
 console.log(data);
 if (data.result == 0) {
        console.info("switchShare Success");
    } else {
        console.error("switchShare failed");
        console.error(data);
    }
}
```

6.4 Screenshot

6.4.1 Returning a Screenshot Path

Interface Description

This interface is invoked to capture an image of the other party and return the image path during a video call.

Notes

Prerequisites: The WebSocket connection with the OpenEyeCall has been set up, and a video call is in progress.

Method Definition

```
OpenEyeCall.prototype.screenShot = function(callbacks)
```

Parameter Description

Table 6-86 Parameter description

Parameter	Type	Mandatory/Optional	Description
callbacks	**Callback**	Mandatory	Callback method.

Table 6-87 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	Callback method.

Table 6-88 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
path	String	If the operation is successful, this parameter indicates the path for storing the generated image. If the operation fails, this parameter does not exist.
result	Number	Configuration result. The value **1** indicates failure. If the operation is successful, this parameter does not exist.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "OEScreenShot",
   "result" : 1,
   "rsp" : 67762
}

{
   "description" : "OEScreenShot",
   "path" : "xxx",
   "rsp" : 67762
}
```

Examples

```
function startScreenShot(){
    console.info("startScreenShot");
    this.global_openEye_SDK.openEyeCall.screenShot({ response: startScreenShotResponse })
}
```

```
function startScreenShotResponse(data){
    console.log(data);
    if (data.result == 0) {
        console.info("startScreenShot Success");
    } else {
        console.error("startScreenShot failed");
        console.error(data);
    }
}
```

6.4.2 Returning the Base64 Code of a Screenshot

Interface Description

This interface is invoked to capture an image of the other party and return the Base64 code of the image during a video call.

Notes

Prerequisites: The WebSocket connection with the OpenEyeCall has been set up, and a video call is in progress.

Method Definition

```
OpenEyeCall.prototype.screenShotBase64 = function(callbacks)
```

Parameter Description

Table 6-89 Parameter description

Parameter	Type	Mandatory/Optional	Description
callbacks	**Callback**	Mandatory	Callback method.

Table 6-90 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	Callback method.

Table 6-91 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
base64	String	If the operation is successful, this parameter indicates the Base64 code of the image. If the operation fails, this parameter does not exist.
result	Number	Configuration result. The value **1** indicates failure. If the operation is successful, this parameter does not exist.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "OEScreenShot",
   "result" : 1,
   "rsp" : 67762
}

{
   "description" : "OEScreenShot",
   "base64" : "xxx",
   "rsp" : 67762
}
```

Examples

```
function startScreenShotBase64(){
    console.info("startScreenShot");
    this.global_openEye_SDK.openEyeCall.screenShotBase64({ response: startScreenShotResponse })
}

function startScreenShotResponse(data){
    console.log(data);
    if (data.result == 0) {
        console.info("startScreenShot Success");
    } else {
        console.error("startScreenShot failed");
        console.error(data);
    }
}
```

6.5 Screen Recording

6.5.1 Starting and Stopping Screen Recording

Interface Description

This interface is invoked to start or stop recording the screen of the other party and return the video path during a video call.

Notes

Prerequisites: The WebSocket connection with the OpenEyeCall has been set up, and a video call is in progress.

Method Definition

```
OpenEyeCall.prototype.videoCatch = function(operation, callbacks)
```

Parameter Description

Table 6-92 Parameter description

Parameter	**Type**	**Mandatory/Optional**	**Description**
operation	int	Mandatory	Operation type. The value **0** indicates starting screen recording and **1** indicates stopping screen recording.
callbacks	**Callback**	Mandatory	Callback method.

Table 6-93 Callback

Parameter	**Type**	**Mandatory/Optional**	**Description**
response	function	Mandatory	Callback method.

Table 6-94 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
path	String	If the operation is successful, this parameter indicates the video path and is returned when the screen recording ends successfully.
result	Number	Configuration result. The value **0** indicates success and **1** indicates failure. If this parameter does not exist, the operation is successful.
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "OECatchVideo",
   "result" : 1,
   "rsp" : 67762
}

{
   "description" : "OECatchVideo",
   "path" : "xxx",
   "rsp" : 67762
}
```

Examples

```
function catchVideo(operation){
    this.global_openEye_SDK.openEyeCall.videoCatch(operation, { response: startVideoCatchResponse })
}

function startVideoCatchResponse(data){
    console.log(data);
    if (data.result == 0) {
        console.info("VideoCatch Success");
    } else {
        console.error("VideoCatch failed");
        console.error(data);
    }
}
```

6.5.2 Starting and Stopping Screen Recording (Periodically Returning Base64 Codes)

Interface Description

This interface is invoked to start or stop recording the screen of the other party and return the Base64 code to implement scheduled recording.

Notes

Prerequisites: The WebSocket connection with the OpenEyeCall has been set up, and a video call is in progress.

Method Definition

```
OpenEyeCall.prototype.videoCatchBase64= function(operation, time, callbacks)
```

Parameter Description

Table 6-95 Parameter description

Parameter	Type	Mandatory/Optional	Description
operation	int	Mandatory	Operation type. The value **0** indicates starting screen recording (encoded by Base64) and **1** indicates stopping screen recording (encoded by Base64).
time	int	Mandatory	When the operation type is **0**, the value of this parameter is the scheduled screen recording duration, which cannot exceed 10 seconds. When the operation type is **1**, the value of this parameter is **0**.
callbacks	**Callback**	Mandatory	Callback method.

Table 6-96 Callback

Parameter	Type	Mandatory/Optional	Description
response	function	Mandatory	Callback method.

Table 6-97 Input parameters of the callback method

Parameter	Type	Description
description	String	Description of the current request.
base64	String	Base64 code of the video, which is returned when the screen recording is successful.
result	Number	Configuration result. The value **0** indicates success and **1** indicates failure (returned when screen recording fails).
rsp	Number	Internal message ID.

NOTE

The following is an example of input parameters of the callback method:

```
{
   "description" : "OECatchVideo",
   "result" : 1,
   "rsp" : 67762
}

{
   "description" : "OECatchVideo",
   "base64" : "xxx",
   "rsp" : 67762
}
```

Examples

```
function catchVideoBase64(operation){
    var time =  document.getElementById('CatchVideoTime').value;
    if (time != "" || operation == 3){
        this.global_openEye_SDK.openEyeCall.videoCatchBase64(operation, parseInt(time), { response:
startVideoCatchResponse })
    }
}

function startVideoCatchResponse(data){
    console.log(data);
    if (data.result == 0) {
        console.info("VideoCatch Success");
    } else {
        console.error("VideoCatch failed");
        console.error(data);
    }
}
```

7 Error Codes

Module	Error Code (Decimal Value)	Description
openeyeSDK	0	The interface is invoked successfully.
openeyeSDK	Non-zero value	The interface fails to be invoked or other errors occur. For details, see other descriptions in the returned value.