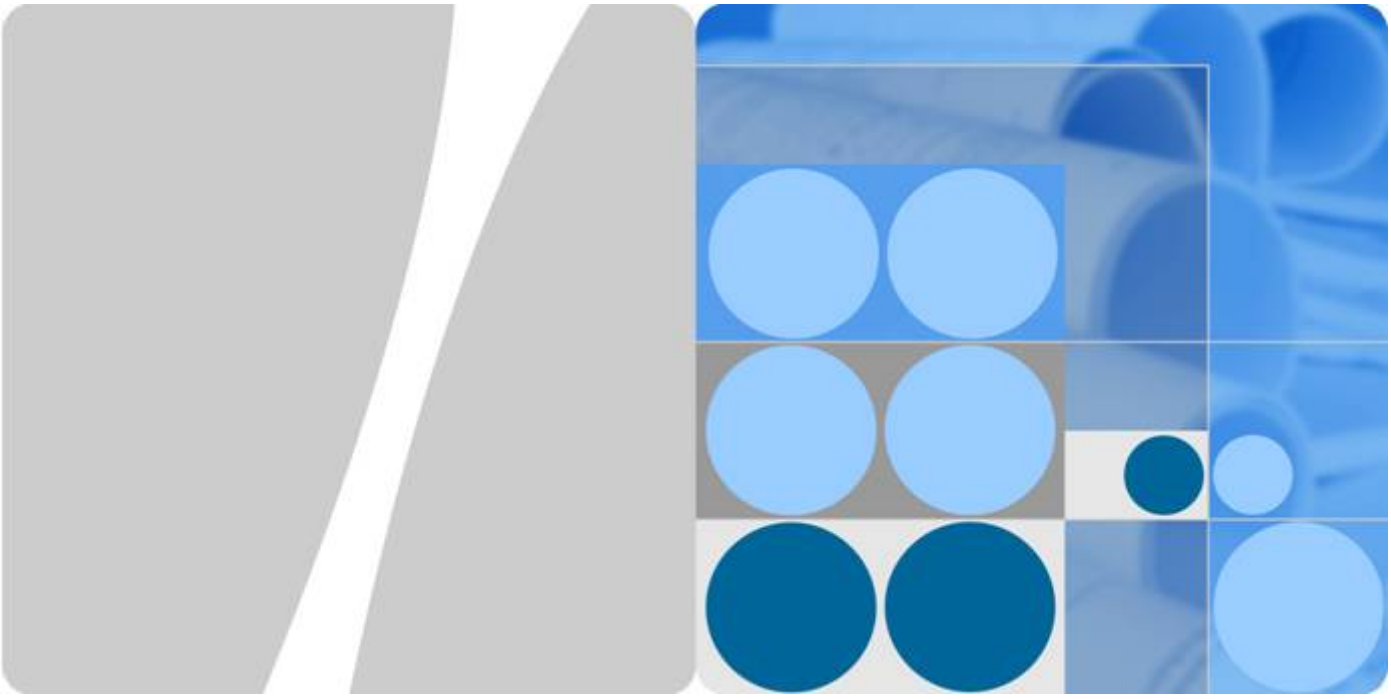

eSDK ICP

V200R001C10

Interface Reference (CC, iOS)

Issue 01

Date 2017-03-27

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.huawei.com

Email: support@huawei.com

Contents

1 About This Document

Purpose

This document describes eSDK CC iOS interface reference. It consists of related data types, interface functions, method definition, parameter description, and examples.

Product Version

The following table lists the product versions related to this document.

Product Name	Version
eSDK ICP CC	V200R001C10
TUP library	V600R006C00B022

Technical Support

- In case of a problem during the development process, submit the problem to the **DevCenter** system for tracking.
- In case of any doubt during the use process, contact us using any of the following methods:
  - Huawei technical support hotline: 400-8828-000
  - Huawei technical support email address: esdk@huawei.com

2 Data Type

2.1 Notifications

Event Notification Name	Return Value	Description
AUTH_MSG_ON_LOGIN	Obtained by using object	Login result notification. The return value **0** indicates that the login is successful, and other values indicate that the login failed.
AUTH_MSG_ON_LOGOUT	Obtained by using object	Logout result notification. The return value **0** indicates that the logout is successful, and other values indicate that the logout failed.
CALL_MSG_ON_QUEUING	None	Call queuing notification.
CALL_MSG_ON_CANCEL_QUEUE	None	Call queuing cancellation notification.
CALL_MSG_ON_QUEUE_INFO	NSDictionary, obtained by using userInfo	Queue information acquisition notification. The keywords are described as follows: POSITION_KEY: in the queue LONGESTWAITTIME_KEY: longest waiting time ONLINEAGENTNUM_KEY: number of online agents
CALL_MSG_ON_QUEUE_TIMEOUT	None	Call queuing timeout notification.
CALL_MSG_ON_CONNECTED	Obtained by using object	Successful call setup notification.
CALL_MSG_ON_DISCONNECTED	Obtained by using object	End-of-call notification. In the TP environment, the return value null indicates that the text chatting ends, the return value **AUDIO_CALL** indicates that the voice conversation ends, and the return value **VIDEO_CALL** indicates that the conference ends.
CALL_MSG_ON_FAIL	Obtained by using object	Call failure notification. The return value is a failure error code.
CALL_GET_VERIFY_CODE	NSDictionary, obtained by using userInfo	Verification code acquisition notification. The keyword is described as follows: verifyCode: verification code picture data encoded using base64

2.2 Enumerations

Message ID	Description
AUTH_MSG_ON_LOGIN	Login
AUTH_MSG_ON_LOGOUT	Logout
CALL_MSG_ON_CONNECTED	Connected to the agent
CALL_MSG_ON_DISCONNECTED	Disconnected from the agent
CALL_MSG_ON_QUEUE_INFO	Queue information
CALL_MSG_ON_VERIFYCODE	Verification code
CALL_MSG_ON_QUEUE_TIMEOUT	Queuing timeout
CALL_MSG_ON_QUEUING	Queuing
CALL_MSG_ON_CANCEL_QUEUE	Queue cancellation
CHAT_MSG_ON_SEND	Send messages
CHAT_MSG_ON_RECEIVE	Receive messages
CALL_MSG_REFRESH_LOCALVIEW	Displaying the local video
CALL_MSG_REFRESH_REMOTEVIEW	Displaying the remote video
CALL_MSG_ON_DROPCALL	Call release
CALL_MSG_ON_FAIL	Failed to make the call
CALL_MSG_ON_CALL_END	The call is disconnected
CALL_MSG_ON_SUCCESS	The call is connected
CALL_MSG_ON_CONNECT	Connection information
CALL_MSG_USER_JOIN	Join the meeting notice
CALL_MSG_USER_END	Notice of termination of meeting
CALL_MSG_USER_STATUS	Conference creation notification
CALL_MSG_GET_VIDEO_INFO	Video stream information
CALL_MSG_USER_NETWORK_ERROR	Notice of disconnection of meetings
CALL_MSG_USER_RECEIVE_SHARED_DATA	Remote sharing data
CALL_MSG_ON_NET_QUALITY_LEVEL	Network status
CALL_MSG_ON_STOP_MEETING	Quit the meeting

Message ID	Description
CALL_MSG_ON_APPLY_MEETING	Apply for a meeting
CALL_MSG_ON_POLL	Polling
CC_MSG_CONTENT	Broadcast

2.2.1 Log Level

Type Definition

```
//日志级别
typedef NS_ENUM(NSInteger, CCLogLevel)
{
    LOG_ERROR   = 0,
    LOG_WARNING = 1,
    LOG_INFO    = 2,
    LOG_DEBUG   = 3,
    LOG_NONE    = 9
 };
```

2.2.2 Video Angle

Type Definition

```
typedef NS_ENUM(NSInteger, VIDEO_ROTATE)
{
ROTATE_DEFAULT = 0,//Not rotate
ROTATE_90 = 90,//Rotate 90° counterclockwise
ROTATE_180 = 180,//Rotate 180° counterclockwise
ROTATE_270 = 270//Rotate 270° counterclockwise
};
```

2.3 Structure

2.3.1 Video Stream Information

Type Definition

```
typedef struct StreamParam
{
float sendLossFraction;         //Packet loss rate of the sender (%)
float sendDelay;                //Average delay of the sender (ms)
float receiveLossFraction;      //Packet loss rate of the recipient (%)
float receiveDelay;             //Average delay of the recipient (ms)
char encodeSize[16];            //Image resolution (sender)
char decodeSize[16];            //Image resolution (recipient)
int videoWidth;               //Video resolution-width
int videoHeight;              //Video resolution-height
}Stream_INFO;
```

3 Initialization Interfaces

3.1 Obtain the Version Information

Interface Description

This interface is used to obtain the version information about the current SDK.

Method Definition

```
//objc code
- (NSString *)getVersion;
```

Usage Description

None

Parameter Description

None

Return Value

Type	Description
NSString*	Version information character string

Example

```
//objc code
[[CCSDK shareInstance] getVersion];
```

3.2 Set the Log Path and Log Level

Interface Description

This interface is used to set the log path and log level of the SDK for facilitating problem identification.

Method Definition

```
//objc code
- (BOOL)setLogPath:(NSString *)path level:(CCLogLevel)level;
```

Usage Description

This interface is invoked before the initialization interface is invoked.

Parameter Description

Parameter	Type	Description
path	NSString*	Log path
level	ENUM	Log level. See**Log Level**

Return Value

Type	Meaning
BOOL	**YES**: successful **NO**: failed

Example

```
//objc code
NSString *path = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES) objectAtIndex:0];
NSString *logPath = [path stringByAppendingPathComponent:@"TUP_LOG"];
[[CCSDK shareInstance] setLogPath:logPath level:CCLogLevelInfo];
```

3.3 Verify the Server Certificate

Interface Description

Server certificate verification is enabled during a network request. If the certificate verification fails, a server access failure is caused.

Method Definition

```
//objc code
-(void)setNeedValidate:(BOOL)needValidateneedValidateDomain:
(BOOL)needValidateDomaincertificateData:(NSData *)certificateData
```

Parameter Description

Parameter	Type	Description
needValidate	BOOL	Whether to verify the certificate
needValidateDomain	BOOL	Whether to verify the domain name
certInputStream	NSData*	Certificate

Return Value

None

Example

```
//objc code
[[CCSDK shareInstance] setHostAddress:self.ipText.text
port:self.portText.text
transSecurity:_isHTTPS
sipServerType:serverType];
if (_isHTTPS)
{
NSString * cerPath = [[NSBundle mainBundle] pathForResource:@"server"
ofType:@"der"];
NSData *dataCA = [NSData dataWithContentsOfFile:cerPath];
[[CCSDK shareInstance] setNeedValidate:YES needValidateDomain:NO
certificateData:dataCA];
}
```

3.4 Initialize the SDK

Interface Description

This interface is used to initialize the TUP and conference components.

Method Definition

```
//objc code
- (void)initSDK;
```

Usage Description

This interface is invoked when the application is started. This interface matches the **De-initialize the SDK**.

Parameter Description

None

Return Value

None

Example

```
//objc code
[[CCSDK shareInstance] initSDK];
```

3.5 De-initialize the SDK

Interface Description

This interface is used to release SDK resources.

Method Definition

```
//objc code
- (void)unInitSDK;
```

Usage Description

This interface matches the **Initialize the SDK**.

Parameter Description

None

Return Value

None

Example

```
//objc code
[[CCSDK shareInstance] unInitSDK];
```

3.6 Set the Access Gateway Address

Interface Description

This interface is used to set the login gateway address.

Method Definition

```
//objc code
- (NSInteger)setHostAddress:(NSString *)ip port:(NSString *)port transSecurity:
(BOOL)transSec sipServerType:(int)serverType;
```

Usage Description

This interface must be invoked before the login interface is invoked. Otherwise, the server address cannot be obtained.

Parameter Description

Parameter	Type	Description
ip	NSString*	Server address
port	NSString*	Server port, which must be a number
transSec	BOOL	**YES**: HTTPS request **NO**: HTTP request
serverType	int	**SERVER_TYPE_TP**: TP environment

Return Value

Type	Description
NSInteger	**0**: The interface is invoked successfully. Other values: The interface failed to be invoked. For details, see the **Error Codes**

Example

```
//objc code
NSInteger setRes = [[CCSDK shareInstance] setHostAddress:self.ipText.text
port:self.portText.text transSecurity:NO];
if (setRes != RET_OK){
[self showMessage:[NSString stringWithFormat:@"setting error:%ld",(long)setRes]];
}
```

3.7 Set the SIP Server Address

Interface Description

This interface is used to obtain the SIP server address for a call.

Method Definition

```
//objc code
- (NSInteger)setSIPServerAddress:(NSString *)ip port:(NSString *)port ;
```

Usage Description

By default, the address obtained from the server is used for the call.

Parameter Description

Parameter	Type	Description
ip	NSString*	Server address
port	NSString*	Server port, which must be a number

Return Value

Type	Meaning
NSInteger	**0**: The interface is invoked successfully. Other values: The interface failed to be invoked. For details, see the **Error Codes**

Example

```
//objc code
[[CCSDK shareInstance] setSIPServerAddress:@"10.174.5.54" port:@"5060"];
```

3.8 Set the Data Encryption Mode

Interface Description

This interface is used to set the encryption mode for data transmission. Both TLS and SRTP encryption is supported.

Method Definition

```
//objc code
- (void)setTransportSecurityUseTLS:(BOOL)enableTLS useSRTP:(BOOL)enableSRTP;
```

Usage Description

Whether encryption is required must be judged based on the environment information.

Parameter Description

Parameter	Type	Description
enableTLS	BOOL	**YES**: TLS encryption **NO**: no encryption
enableSRTP	BOOL	**YES**: SRTP encryption **NO**: no encryption

Return Value

None

Example

```
//objc code
[[CCSDK shareInstance] setTransportSecurityUseTLS:YES useSRTP:YES];
```

3.9 Obtain the Verification Code

Interface Description

This interface is used to obtain the call verification code.

Method Definition

```
//objc code
- (BOOL)getVerifyCode;
```

Usage Description

The verification code is obtained before a call. The returned verification code is obtained by receiving the notification **CALL_GET_VERIFY_CODE**. The returned verification code information is picture data encoded using base64. The verification code can be viewed after the data is displayed as a picture. The verification code that is viewed needs to be imported during the call.

Parameter Description

None

Return Value

Type	Description
BOOL	**YES**: The interface is invoked successfully. **NO**: The interface failed to be invoked.

Example

```
//objc code
//Register the CALL_GET_VERIFY_CODE notification
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(showVerifyCode:)
name:CALL_GET_VERIFY_CODE object:nil];

//Invoke the interface for obtaining the verification code
[[CCSDK shareInstance] getVerifyCode];
//Process the verification code acquisition notification
- (void)showVerifyCode:(NSNotification *)notify
{
if ([notify.object intValue] == 0) {
NSString *encodedImageStr = [notify.userInfo objectForKey:@"verifyCode"];
NSData *decodedImageData =  [[NSData alloc]
initWithBase64EncodedString:encodedImageStr options:0];

UIImage *decodedImage= [UIImage imageWithData:decodedImageData];
dispatch_async(dispatch_get_main_queue(), ^{
//Throw back to the main thread for display
});
}
else{
dispatch_async(dispatch_get_main_queue(), ^{
//Throw back to the main thread for prompting that the acquisition failed
});
}
}
```

4 Login Interfaces

4.1 Login

Interface Description

This interface is used by a user for login to the soft client.

Method Definition

```
//objc code
- (NSInteger)login:(NSString *)vndid userName:(NSString *)userName;
```

Usage Description

The return value only indicates whether the interface is invoked successfully. The notification is registered before the login. Whether the login is successful must be judged based on the notification **AUTH_MSG_ON_LOGIN**.

Parameter Description

Parameter	Type	Description
vndid	NSString*	Virtual call center ID, value range: 1-999
userName	NSString*	User name, length: 1-20 characters (digits and letters)

Return Value

Type	Description
NSInteger	**0**: The interface is invoked successfully. Other values: The interface failed to be invoked. For details, see the **Error Codes**.

Example

```
//objc code
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(loginResult:)
name:AUTH_MSG_ON_LOGIN object:nil];

NSInteger ret = [[CCSDK shareInstance] login:@"1"
userName:self.userNameText.text];
if (ret == RET_OK) {
}else{
[self showMessage:[NSString stringWithFormat:@"Failed to invoke the login
interface:%ld",(long)ret]]; };
```

4.2 Logout

Interface Description

This interface is used by a user for logout from the soft client.

Method Definition

```
//objc code
- (void)logout;
```

Usage Description

The notification is registered before the logout. Whether the logout is successful must be judged based on the notification **AUTH_MSG_ON_LOGOUT**.

Return Value

None

Example

```
//objc code
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(logoutResult:)
name:AUTH_MSG_ON_LOGOUT object:nil];

[[CCSDK shareInstance] logout];
```

5 Call Interfaces

5.1 Text conversation

5.1.1 Initiate a text conversation

Interface Description

Users can chat with the agent, which currently supports only the MS environment。

Method Definition

```
//objc code
- (NSInteger)webChatCall:(NSString *)accessCode callData:(NSString *)callData
verifyCode:(NSString *)verifyCode;
```

Usage Description

The return value can only indicate whether the interface is successful, initiate a text chat before the successful registration and failure notification, and successfully establish a text conversation with the agent received a **CALL_MSG_ON_CONNECTED** notification, the failure will receive **CALL_MSG_ON_FAIL notice**, see the notice.

If the server to open the verification code function, you need to call the authentication code interface, and then call this interface; if the server verification code function is closed, then directly call the interface, and the verification code parameters may not pass.

Parameter Description

Parameter	Type	Description
accessCode	NSString	Access code, platform configuration data, 1-24 digits
callData	NSString	Call with data, length range [0,1024]
verifyCode	NSString*	The authentication code returned by the incoming interface getVerifyCode

Return Value

Type	Description
NSInteger	0 indicates that the interface call is successful, the other means that the interface call fails, reference **Error Codes**.

Example

```
//objc code
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(callSuccess:)
name:CALL_MSG_ON_CONNECTED
object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(callFail:)
name:CALL_MSG_ON_FAIL object:nil];

[[CCSDK shareInstance] webChatCall:@"60011" callData:self.CallData
verifyCode:@"1234"];
```

5.1.2 Send text messages

Interface Description

Send and receive text chat messages, this feature currently only supports MS environment

Method Definition

```
//objc code
- (NSInteger)sendMsg:(NSString *)message;
```

Usage Description

The return value can only indicate whether the interface is successful, send a text message before the registration is successful, failed and received a message notification, send successfully received CHAT_MSG_ON_SUCCESS , send a failure will receive CHAT_MSG_ON_FAIL , the message will be CHAT_MSG_ON_RECEIVE notice, see details **Notifications**

Parameter Description

Parameter	Type	Description
message	NSString	Message content, minimum of 1 character, maximum of 300

Return Value

Type	Description
NSInteger	0 indicates that the interface call is successful, the other means that the interface call fails, reference **Error Codes**

Example

```
//objc code
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(sendMsgSuccess)
name:CHAT_MSG_ON_SUCCESS object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(sendMsgFail:)
name:CHAT_MSG_ON_FAIL object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(receiveMsg:)
name:CHAT_MSG_ON_RECEIVE object:nil];
[[CCSDK shareInstance] sendMsg:self.chatText.text]
```

5.1.3 Release the text call

Interface Description

Send and receive text chat messages, this feature currently only supports MS environment.

Method Definition

```
//objc code
- (NSInteger)sendMsg:(NSString *)message;
```

Usage Description

The return value can only indicate whether the interface is successful, send a text message before the registration is successful, failed and received a message notification, send successfully received **CHAT_MSG_ON_SUCCESS** , send a failure will receive **CHAT_MSG_ON_FAIL** , the message will be **CHAT_MSG_ON_RECEIVE** , see details **Notifications**.

Parameter Description

None.

Return Value

None

Example

```
//objc code
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(callEnd:)
name:CALL_MSG_ON_DISCONNECTED object:nil];

[[CCSDK shareInstance] releaseWebChatCall];
```

5.2 Voice/Video Call

5.2.1 Establish a Voice/Video Call

Interface Description

This interface is used to establish a voice/video call with an agent.

Method Definition

```
//objc code
- (NSInteger)makeCall:(NSString *)accessCode callType:(NSString *)callType
callData:(NSString *)callData verifyCode:(NSString *)verifyCode;
```

Usage Description

The return value only indicates whether the interface is invoked successfully. The success and failure notifications are registered before a call. The **CALL_MSG_ON_CONNECTED**notification is received when the call is successfully established, and the **CALL_MSG_ON_FAIL** notification is received when the call failed to be established. For details, see the **Notifications**.

Parameter Description

Parameter	Type	Description
accessCode	NSString*	Access code, data configured on the platform, length: 1-24 digits
callType	NSString*	Call type **AUDIO_CALL**: voice call **VIDEO_CALL**: video call
callData	NSString*	Associated call data, length: [0,1024]
verifyCode	NSString*	Verification code returned by the **Obtain the Verification Code** interface

Return Value

Type	Description
NSInteger	**0**: The interface is invoked successfully. Other values: The interface failed to be invoked. For details, see the **Error Codes**.

Example

```
//objc code
[[NSNotificationCenter defaultCenter] addObserver:self
```

```
selector:@selector(callSuccess:)
name:CALL_MSG_ON_CONNECTED
object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(callFail:)
name:CALL_MSG_ON_FAIL object:nil];

[[CCSDK shareInstance] makeCall:self.aCode callType: AUDIO_CALL
callData:self.CallData verifyCode:@"5679"];//The verification code is that
returned by the interface for obtaining the verification code.
[[CCSDK shareInstance] makeCall:self.aCode callType: VIDEO_CALL
callData:self.CallData verifyCode:@"5679"];//The verification code is that
returned by the interface for obtaining the verification code.
```

5.2.2 Voice calls are upgraded to video calls

Interface Description

Upgrade the voice call to a video call and currently only support the MS environment。

Method Definition

```
//objc code
- (NSInteger)updateToVideo;
```

Usage Description

Called in the presence of a voice call, the return value can only indicate whether the interface was successfully called. The successful and failed notification before the upgrade was successfully received. The **CALL_MSG_ON_CONNECTED** notification was received successfully. The upgrade failed to receive the **CALL_MSG_ON_FAIL** notification.

Parameter Description

None

Return Value

Type	Description
NSInteger	0 indicates that the interface call is successful, the other means that the interface call fails, reference **Error Codes**.

Example

```
//objc code
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(callSuccess:)
name:CALL_MSG_ON_CONNECTED
object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(callFail:)
name:CALL_MSG_ON_FAIL object:nil];
[[CCSDK shareInstance] updateToVideo];
```

5.3 End a Call

Interface Description

This interface is used to end the current call.

Method Definition

```
//objc code
- (void)releaseCall;
```

Usage Description

After the notification is registered, the **CALL_MSG_ON_DISCONNECTED** notification is received when the call ends, and the **CALL_MSG_ON_USER_LEAVE** notification is received when a user leaves the conference. For details, see the **Error Codes**.

Parameter Description

None

Return Value

None

Example

```
//objc code
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(callEnd:)
name:CALL_MSG_ON_DISCONNECTED object:nil];

[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(UserLeave:)
name:CALL_MSG_ON_USER_LEAVE object:nil];
[[CCSDK shareInstance] releaseCall];
```

5.4 Queuing-Related Interfaces

5.4.1 Obtain Queue Information

Interface Description

When a call is waiting in the queue, this interface is used to obtain the queue information.

Method Definition

```
//objc code
- (void)getCallQueueInfo;
```

Usage Description

After the notification is registered, the CALL_MSG_ON_QUEUE_INFO notification is received when the queue information is obtained.

Parameter Description

None

Return Value

None

Example

```
//objc code
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(receiveQueueInfo:)
name:CALL_MSG_ON_QUEUE_INFO object:nil];
[[CCSDK shareInstance] getCallQueueInfo];
```

5.4.2 Cancel Queuing

Interface Description

When a call is waiting in the queue, this interface is used to cancel the queuing and end the call.

Method Definition

```
//objc code
- (void)cancelQueue;
```

Usage Description

After the notification is registered, the **CALL_MSG_ON_CANCEL_QUEUE** notification is received when the queuing is canceled, and the **CALL_MSG_ON_QUEUE_TIMEOUT** notification is received when the queuing times out. For details, see the **Error Codes**.

Parameter Description

None

Return Value

None

Example

```
//objc code
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(queueIsCancel:)
name:CALL_MSG_ON_CANCEL_QUEUE object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(callQueueTimeOut:)
name:CALL_MSG_ON_QUEUE_TIMEOUT object:nil];
[[CCSDK shareInstance] cancelQueue
```

6 Device Interfaces

6.1 Set the display container

6.1.1 Set the Local/Remote Video Display Container

Interface Description

This interface is used to set the local/remote video window handle.

Method Definition

```
//objc code
- (void)setVideoContainer:(id)localView remoteView:(id)remoteView;
```

Usage Description

This interface must be invoked before the video call is established to ensure normal display of the video window.

Parameter Description

Parameter	Type	Description
localView	id	Local video container. See the EAGLView and subclasses.
remoteView	id	Remote video container. See the EAGLView and subclasses.

Return Value

None

Example

```
//objc code
self.remoteView = [EAGLView getRemoteVideoView];
self.localView = [EAGLView getLocalVideoView];
[[CCSDK shareInstance] setVideoContainer:self.localView
remoteView:self.remoteView];
```

6.1.2 Set up desktop sharingDisplay container

Interface Description

Set the desktop sharing window handle, currently only support the MS environment.

Method Definition

```
//objc code
- (void)setDesktopShareContainer:(UIImageView *)shareView;
```

Usage Description

Currently only support the MS platform, you must open the desktop share before the other call to ensure that the shared window is displayed properly.

Parameter Description

Parameter	Type	Description
shareView	UIImageView	Share the display container

Return Value

None.

Example

```
//objc code
CGRect sFrame = self.remoteView.frame;
self.ScreenshareView = [[UIImageView alloc] initWithFrame:sFrame];
[[CCSDK shareInstance] setDesktopShareContainer:self.ScreenshareView];
```

6.2 Set Media Network Parameters

6.2.1 Set the Video Display Mode

Interface Description

This interface is used to set the video display mode (video quality preferred or video fluency preferred).

Method Definition

```
//objc code
- (BOOL)setVideoMode:(int)videoMode;
```

Usage Description

By default, video quality is preferred. To change the default value, invoke this interface before the video call is established.

Parameter Description

Parameter	Type	Description
videoMode	int	**0**: video quality preferred **1**: video fluency preferred

Return Value

Type	Meaning
BOOL	**YES**: The setting is successful. **NO**: The setting failed.

Example

```
//objc code
[[CCSDK shareInstance] setVideoMode:0];
```

6.2.2 Set the Bandwidth

Interface Description

This interface is used to set the media network bandwidth for ensuring normal display of the screen.

This interface does not support calls at this time

Method Definition

```
//objc code
- (BOOL)setDataRate:(int)dataRateValue;
```

Usage Description

The default value is **512K**. To change the default value, invoke this interface before the video call is established.

Parameter Description

Parameter	Type	Description
dataRateValue	int	Bandwidth value, value options: **128K**, **256K**, **384K**, **512K**, and **768K**

Return Value

Type	Meaning
BOOL	**YES**: The setting is successful. **NO**: The setting failed.

Example

```
//objc code
[[CCSDK shareInstance] setDataRate:768];
```

6.3 Set the Video Rotation Angle

Interface Description

This interface is used to manually adjust the video angle during a video call.

Method Definition

```
//objc code
- (BOOL)setVideoRotate:(VIDEO_ROTATE)rotate;
```

Usage Description

During a video call, the video angle may be deflective after switching between horizontal screen and vertical screen or after switching between front-facing camera and rear-facing camera. In this case, invoke this interface for adjustment.

Parameter Description

Parameter	Type	Description
rotate	**Video Angle**	Value options: **0**, **90**, **180**, and **270**

Return Value

Type	Meaning
BOOL	**YES**: successful **NO**: failed

Example

```
//objc code
[[CCSDK shareInstance] setVideoRotate:180];
```

6.4 Obtain the Video Stream Information

Interface Description

During a video call, this interface is used to view the video resolution, network delay, and packet loss rate.

Method Definition

```
//objc code
- (Stream_INFO)getChannelInfo;
```

Usage Description

Video information can be viewed after a video call is successfully established and the images of both sides can be viewed.

Parameter Description

None

Return Value

Type	Meaning
Video Stream Information	Video stream information

Example

```
//objc code
- (void)getVideoStreamInfo{
Stream_INFO info = [[CCSDK shareInstance] getChannelInfo];
self.videoInfoView.text = [NSString stringWithFormat:@"packet loss rate of the
sender:%f\n average delay of the sender:%fms\n packet loss rate of the recipient:
%f\n average delay of the recipient:%fms\n sending resolution:%s\n receiving
resolution:
%s",info.sendLossFraction,info.sendDelay,info.receiveLossFraction,info.receiveDela
y,info.encodeSize,info.decodeSize];
}
```

6.5 Switch Between the Front-facing Camera and Rear-facing Camera

Interface Description

This interface is used to switch between the front-facing camera and rear-facing camera during a video call.

Method Definition

```
//objc code
- (BOOL)switchCamera:(int)index;
```

Usage Description

By default, the front-facing camera is selected.

Parameter Description

Parameter	Type	Description
index	int	**1**: front-facing camera **0**: rear-facing camera

Return Value

Type	Meaning
BOOL	**YES**: The switching is successful. **NO**: The switching failed.

Example

```
//objc code
[[CCSDK shareInstance] switchCamera:0];
```

6.6 Speaker-Related Interfaces

6.6.1 Switch Between the Speaker Mode and Earpiece Mode

Interface Description

This interface is used to switch between the speaker mode and earpiece mode during a voice call.

Method Definition

```
//objc code
- (BOOL)changeAudioRoute:(int)route;
```

Usage Description

By default, the earpiece mode is selected.

Parameter Description

Parameter	Type	Description
route	int	**0**: earpiece mode **1**: speaker mode

Return Value

Type	Meaning
BOOL	**YES**: The switching is successful. **NO**: The switching failed.

Example

```
//objc code
[[CCSDK shareInstance] changeAudioRoute:1];
```

6.6.2 Obtain the Speaker Volume

Interface Description

This interface is used to obtain the speaker volume.

Method Definition

```
//objc code
- (NSInteger)getSpeakerVolume;
```

Usage Description

None

Parameter Description

None

Return Value

Type	Meaning
NSInteger	Speaker volume

Example

```
//objc code
NSInteger volume = [[CCSDK shareInstance] getSpeakerVolume];
NSLog(@"speak volume:%ld",(long)volume);
```

6.6.3 Mute the Speaker

Interface Description

This interface is used to mute the speaker or cancel muting the speaker.

Method Definition

```
- (BOOL)setSpeakMute:(BOOL)isMute;
```

Usage Description

```
This interface is used during a voice call.
```

Parameter Description

Parameter	Type	Description
isMute	BOOL	**YES**: mute the speaker **NO**: cancel muting the speaker

Return Value

Type	Description
BOOL	**YES**: The setting is successful. **NO**: The setting failed.

Example

```
//objc code
[[CCSDK shareInstance] setSpeakMute:NO];
```

6.7 Mute the Microphone

Interface Description

This interface is used to mute or cancel muting the microphone during a voice call.

Method Definition

```
//objc code
- (BOOL)setMicMute:(BOOL)isMute;
```

Usage Description

None

Parameter Description

Parameter	Type	Description
isMute	BOOL	**YES**: mute the microphone **NO**: cancel muting the microphone

Return Value

Type	Description
BOOL	**YES**: successful **NO**: failed

Example

```
//objc code
[[CCSDK shareInstance] setMicMute:YES];
```

7 Error Codes

Error Code	Description
0	Successful.
-1	Parameter error.
-2	The text chat is not established.
-3	The voice call is not connected.
-4	Nobody answered the call.
-5	Network error.
10-100-002	Incorrect settings of EVENT_METHOD in platform.properties on the server.
10-100-003	Not connected to the CCS, and the WAS service is not available.
10-100-004	WebmAnyService is unavailable.
10-100-005	No permission to invoke this interface.
10-100-006	The user has not logged in.
10-100-007	The user request parameter is empty or invalid.
10-100-008	The user has logged in.
10-100-009	The resource is not available.
10-100-010	The method is not supported.
10-100-011	Status error.
10-100-012	WebmServer of the user is blank.
10-100-013	The VDN ID does not exist.
10-100-014	The access code does not exist.

Error Code	Description
10-100-015	The number of logged-in users has reached the maximum.
10-100-016	The configuration agent is null.
10-100-017	Configuration agent transmission error.
10-200-001	A call matching this access code already exists.
10-200-002	Exceeded the maximum number of calls.
10-200-003	The call does not exist.
10-200-004	Code conversion failed.
10-200-006	Not in a queue.
10-200-007	The call is not established.
10-200-008	The verification code is invalid.
10-200-009	Failed to generate the verification code.
10-300-001	In a conference or requesting for a conference.
10-300-002	The conference does not exist.

8 Change History

Date	Issue	Change Description
2017-3-20	V2.1.10	The document V200R001C10 is released. Full fit MS & TP function.
2016-12-31	V2.1.00	This issue is the first official release.