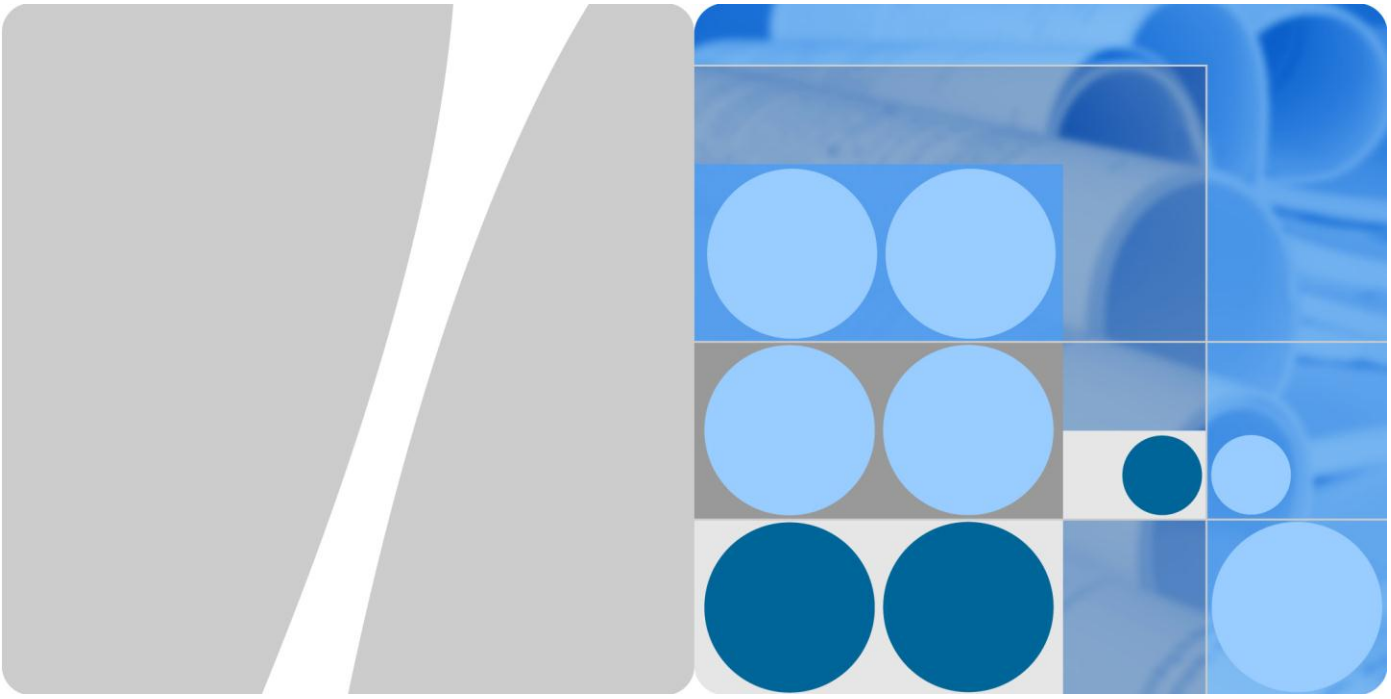

eSDK ICP V200R001C00 Interface Reference 01 (CC, Android)

Issue 01

Date 2017-02-20

HUAWEI TECHNOLOGIES CO., LTD.

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 the interfaces provided by the eSDK CC Android from aspects including the data type, interface function, method definition, parameter description, and example.

Product Version

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

Product Name	Version
eSDK ICP CC	V200R001C00
TUP	V100R001C50SPC900B010

Technical Support

- During the development, you can submit a problem ticket to DevCenter and follow the progress.
- If you have any problem when using this document, contact us in either of the following ways:

  Hotline: 400-8828-000

  Email: esdk@huawei.com

2 Event Notification

The following messages are returned after you invoke the interfaces. You can perform the next-step operation based on these messages.

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
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_CONNECT	Connection information
CALL_MSG_GET_VIDEO_INFO	Video stream information
CALL_MSG_ON_NET_QUALITY_LEVEL	Network status
CALL_MSG_ON_POLL	Polling
CC_MSG_CONTENT	Broadcast

3 Initialization Interfaces

3.1 getVersion

Interface Description

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

Method Definition

```
//java code
public String getVersion()
```

Parameter Description

None

Return Value

Type	Description
String	Version number

Example

```
//java code
MobileCC.getInstance().getVersion();
```

3.2 initSDK

Interface Description

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

Usage Description

You can invoke this interface in the onCreate() method under the project's Application. It is invoked after the 3.4 setLog interface is invoked.

Method Definition

```
//java code
public void initSDK (Application app)
```

Parameter Description

Parameter	Type	Description
app	Application	Indicates the project's Application, used to start the SDK service.

Return Value

None

Example

```
//java code
public class CCApplication extends Application
{
   @Override
   public void onCreate()
   {
      super.onCreate();
      MobileCC.getInstance().setLog("eSDK", 3);
      MobileCC.getInstance().initSDK(this);

   }

}
```

3.3 unInitSDK

Interface Description

This interface is used to de-initialize the relevant modules to release resources when the application ends.

Usage Description

You can invoke this interface in the onTerminate() method under the project's Application.

Method Definition

```
//java code
public void unInitSDK ()
```

Parameter Description

None

Return Value

None

Example

```
//java code
public class CCApplication extends Application
{
   ......

   @Override
   public void onTerminate()
   {
      super.onTerminate();
      MobileCC.getInstance().unInitSDK(); //Stop the SDK service.
   }
}
```

3.4 setLog

Interface Description

This interface is used to initialize the TUP or HME log.

Method Definition

```
//java code
public int setLog(String path, int level)
```

Parameter Description

Parameter	Type	Description
path	String	Indicates the log path. A maximum of 60 characters are allowed.
level	int	Indicates the log level, from level 1 to level 3.

Return Value

Type	Description
int	• **0**: success • **-1**: incorrect parameter

Example

```
//java code
public class CCApplication extends Application
{
   @Override
   public void onCreate()
   {
      super.onCreate();
      MobileCC.getInstance().setLog("eSDK", 3);

   }

}
```

3.5 setServerCertificateValidation

Interface Description

This interface is used to enable the verification on the server certificate in the network request. If the certificate fails to be verified, access to the server fails.

Method Definition

```
//java code
public void setServerCertificateValidation(boolean needValidate, boolean
needValidateDomain, InputStream certInputStream)
```

Parameter Description

Parameter	Type	Description
needValidate	boolean	Indicates whether to verify the certificate.

Parameter	Type	Description
needValidateDomain	boolean	Indicates whether to verify the domain name.
certInputStream	InputStream	Indicates the certificate.

Return Value

None

Example

```
//java code
InputStream inputStream = null;
try
{
    inputStream = CCAPP.getInstances().getApplication().getAssets().open("certs/" +
"server.cer");
   MobileCC.getInstance().setServerCertificateValidation(false,false, inputStream);

} catch (IOException ioe)
{
   ICSLogUtil.d(TAG, "e: " + ioe.getMessage());
}
finally
{
   if (inputStream != null)
   {
      try
      {
         inputStream.close();
      }
      catch (IOException e)
      {
         ICSLogUtil.d(TAG, "e" + e.getMessage());
      }
   }
}
```

3.6 setHostAddress

Interface Description

This interface is used to set the URL for the application to access the background server, virtual contact center ID, and user name. Set **transSecurity** according to the server configuration.

Usage Description

After invoking this interface, judge the return value and perform the next-step operation based on the value.

Method Definition

```
//java code
public int setHostAddress(String IPStr, String portStr, boolean transSecurity, int sipServerType)
```

Parameter Description

Parameter	Type	Description
IPStr	String	Indicates the server IP address, for example, **10.66.110.253**.
portStr	String	Indicates the server port number, for example, **8080**.
transSecurity	boolean	Indicates the authentication mode. • **true**: HTTPS • **false**: HTTP
sipServerType	int	Indicates the platform type. **MobileCC.SERVER_TP**: TP platform

Return Value

Type	Description
int	• **0**: success • **-1**: incorrect parameter

Example

```
 //java code
private int setHostAddress()
   {
      String ipStr = etIP.getText().toString();
      String portStr = etPort.getText().toString();

      return MobileCC.getInstance().setHostAddress(ipStr,
                  portStr, false, MobileCC.SERVER MS);

      return -1;
   }
```

3.7 setSIPServerAddress

Interface Description

The SIP server address can be obtained automatically. This interface is invoked for special requirement only.

Method Definition

```
//java code
public int setSIPServerAddress(String IPStr, String portStr)
```

Parameter Description

Parameter	Type	Description
IPStr	String	Indicates the server IP address, for example, **10.66.110.253**.
portStr	String	Indicates the port number, for example, **8080**.

Return Value

Type	Description
int	• **0**: success • **-1**: incorrect parameter

Example

```
 //java code
MobileCC.getInstance().setSIPServerAddress("10.174.5.54", "5060");
```

3.8 setTransportSecurity

Interface Description

This interface is used to set the encryption mode, which is determined by the server configuration.

Method Definition

```
//java code
public void setTransportSecurity(boolean enableTLS, boolean enableSRTP)
```

Parameter Description

Parameter	Type	Description
enableTLS	boolean	Indicates whether to enable TLS encryption.
enableSRTP	boolean	Indicates whether to enable SRTP encryption.

Return Value

None

Example

```
//java code
MobileCC.getInstance().setTransportSecurity(true, true);
```

4 Authentication Interfaces

Class: MobileCC

4.1 login

Interface Description

This interface is used to log in to the server.

Usage Description

Before you log in to the server, set the server IP address and port by invoking the 3.6 setHostAddress interface.

The return value only indicates whether the interface is invoked successfully or not. Register the receiver before you log in to the server to receive notification messages. AUTH_MSG_ON_LOGIN indicates whether you have logged in to the server successfully.

Method Definition

```
//java code
epublic int login(String vdnId, String userName)
```

Parameter Description

Parameter	Type	Description
vdnId	String	Indicates the VDN ID, which is a number. The default value is **1**.
userName	String	Indicates the user name, for example, **test**. The user name is no longer than 20 characters.

Return Value

Type	Description
int	• **0**: success • **-1**: incorrect parameter

Example

```
//java code
//Set the filter.
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.AUTH_MSG_ON_LOGIN);

//Initialize the receiver.
private BroadcastReceiver receiver = new BroadcastReceiver()
   {
      @Override
      public void onReceive(Context context, Intent intent)
      {
         String action = intent.getAction();
         BroadMsg broadMsg = (BroadMsg) intent
            .getSerializableExtra(NotifyMessage.CC_MSG_CONTENT);

         if (NotifyMessage.AUTH_MSG_ON_LOGIN.equals(action))
         {
            if (("0").equals(broadMsg.getRecode()))
            {
               //Log in successfully.
            }
         }
      }
   };

//Monitor the broadcast.
registerReceiver(receiver, filter);

//Perform the login operation.
 if (0 != MobileCC.getInstance().login("1", etName.getText()
                .toString().trim()))
   {
      //The login request has been initiated.
   }
```

4.2 logout

Interface Description

This interface is used to log out of the server.

Usage Description

Register the receiver before logging out. AUTH_MSG_ON_LOOUT indicates whether you have logged out of the server successfully.

Method Definition

```
//java code
public void logout()
```

Parameter Description

None

Return Value

None

Example

```
//java code
//Set the filter.
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.AUTH MSG ON LOGOUT);

//Initialize the receiver.
private BroadcastReceiver receiver = new BroadcastReceiver()
   {
      @Override
      public void onReceive(Context context, Intent intent)
      {
         String action = intent.getAction();
         BroadMsg broadMsg = (BroadMsg) intent
           .getSerializableExtra(NotifyMessage.CC MSG CONTENT);

        if (NotifyMessage.AUTH MSG ON LOGOUT.equals(action))
          {
             if (("0").equals(broadMsg.getRecode()))
             {
               //Logout succeeds.
             } else
             {
               //Logout fails.
             }
          }
      }
   };

//Register the receiver.
registerReceiver(receiver, filter);

 @Override
 public void onClick(View view)
 {
```

```
        switch (view.getId())
        {
            case R.id.btn_exit:
                //Click.
                MobileCC.getInstance().logout();
                break;

            default:
                break;
        }
    }
```

5 Call Interfaces

5.1 makeCall

Interface Description

This interface is used to initiate a video call.

Usage Description

The following messages are returned after you invoke this interface:

- CALL_MSG_ON_CONNECTED: connecting to the agent
- CALL_MSG_ON_QUEUING: queuing
- CALL_MSG_ON_FAIL: failed to make the call

Prerequisites

- A user has logged in to the app.
- The user has obtained the verification code.

Method Definition

```
//java codepublic int makeCall(String accessCode, String callType, String callData,
String varifyCode)
```

Parameter Description

Parameter	Type	Description

Parameter	Type	Description
accessCode	String	Indicates the access code.
callType	String	Indicates the call type. **1**: MobileCC.SERVER_TP
callData	String	Indicates the channel-associated data.
varifyCode	String	Indicates the verification code.

Return Value

Type	Description
int	• **0**: success • **-1**: failure

Example

```
//java code
//Set the filter.
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.CALL MSG ON CONNECTED);
filter.addAction(NotifyMessage.CALL MSG ON QUEUING);
filter.addAction(NotifyMessage.CALL MSG ON SUCCESS);

//Initialize the receiver.
 private BroadcastReceiver receiver = new BroadcastReceiver()
   {
       @Override
       public void onReceive(Context context, Intent intent)
       {
           String action = intent.getAction();
           BroadMsg broadMsg = (BroadMsg) intent
              .getSerializableExtra(NotifyMessage.CC MSG CONTENT);

           if (NotifyMessage.CALL MSG ON CONNECTED.equals(action))
           {
              //Connect to the agent.
           }
          else if (NotifyMessage.CALL_MSG_ON_QUEUING.equals(action))
           {
              //Queuing…

           }
       }
   }
```

```
//Register the receiver.

 @Override
   protected void onResume()
   {
       super.onResume();
       registerReceiver(receiver, filter);
   }

//Make a call.
MobileCC.getInstance().makeCall(accessCode, MobileCC.SERVER_TP + "", callData,
verifyCode)
```

5.2 releaseCall

Interface Description

This interface is used to release a video call.

Usage Description

After an anonymous call is released, the eSDK layer reports CALL_MSG_ON_DISCONNECTED. You can enable the listening function for this notification on the application so that the application can process the message.

Prerequisites

A video call has been set up.

Method Definition

```
//java code
public void releaseCall()
```

Parameter Description

None

Return Value

None

Example

```
//Set the filter.
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.NotifyMessage.CALL MSG ON DISCONNECTED);

//Initialize the receiver.
 private BroadcastReceiver receiver = new BroadcastReceiver()
```

```
    {
        @Override
        public void onReceive(Context context, Intent intent)
        {
            String action = intent.getAction();
            BroadMsg broadMsg = (BroadMsg) intent
               .getSerializableExtra(NotifyMessage.CC_MSG_CONTENT);

            if (NotifyMessage.NotifyMessage.CALL_MSG_ON_DISCONNECTED.equals(action))
            {
             //If the event is received in TP, the user has been disconnected from the
agent.
            }
        }
    }


//Register the receiver.

 @Override
    protected void onResume()
    {
        super.onResume();
        registerReceiver(receiver, filter);
    }

//Send the request.
MobileCC.getInstance().releaseCall();
```

5.3 getCallQueueInfo

Interface Description

This interface is used to obtain the queue information, such as location, after the user initiates a call and enters a queue.

Usage Description

When the query result is returned, the eSDK layer reports CALL_MSG_ON_QUEUE_INFO. You can enable the listening function for this notification on the application so that the application can process the message.

The returned result is saved in notifyMsg, and its structure is described in the table below:

Key	Description
position	Indicates where the call is in the queue.

Prerequisites

- A user has logged in to the app.

- The call is in a queue.

Method Definition

```
//java code
public void getCallQueueInfo()
```

Parameter Description

None

Return Value

None

Example

```
//java code
//Set the filter.
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.NotifyMessage.CALL MSG ON QUEUE INFO);

//Initialize the receiver.
 private BroadcastReceiver receiver = new BroadcastReceiver()
   {
       @Override
       public void onReceive(Context context, Intent intent)
       {
           String action = intent.getAction();
           BroadMsg broadMsg = (BroadMsg) intent
            .getSerializableExtra(NotifyMessage.CC_MSG_CONTENT);

           if (NotifyMessage.NotifyMessage.CALL_MSG_ON_QUEUE_INFO.equals(action))
           {
              if (null == broadMsg.getRetCode())
              {
                 //Failed to obtain the queue information. View the error code using
broadMsg.getErrorCode().             }
               else
               {
                   String retcode = broadMsg.getRetCode();
                   if (MobileCC.MESSAGE_OK.equals(retcode))
                   {
                   //The call is in a queue. View the number of calls waiting in front
of this call using broadMsg.getPostion().
                   }
                   else
                   {
                     //Not in any queue
                   }
               }


           }
```

```
        }
    }


//Register the receiver.

 @Override
    protected void onResume()
    {
        super.onResume();
        registerReceiver(receiver, filter);
    }

//Send the request.
MobileCC.getInstance().getCallQueueInfo();
```

5.4 cancelQueue

Interface Description

This interface is used to cancel queuing.

Usage Description

When the query result is returned, the eSDK layer reports CALL_MSG_ON_CANCEL_QUEUE. You can enable the listening function for this notification on the application so that the application can process the message.

Prerequisites

- A user has logged in to the app.
- The call is in a queue.

Method Definition

```
//java code
public void cancelQueue()
```

Parameter Description

None

Return Value

None

Example

```
//java code
//Set the filter.
private IntentFilter filter;
filter = new IntentFilter();
```

```
filter.addAction(NotifyMessage.NotifyMessage.CALL_MSG_ON_CANCEL_QUEUE);

//Initialize the receiver.
 private BroadcastReceiver receiver = new BroadcastReceiver()
    {
        @Override
        public void onReceive(Context context, Intent intent)
        {
            String action = intent.getAction();
            BroadMsg broadMsg = (BroadMsg) intent
               .getSerializableExtra(NotifyMessage.CC_MSG_CONTENT);

            if (NotifyMessage.NotifyMessage.CALL_MSG_ON_CANCEL_QUEUE.equals(action))
            {
              //Cancel queuing successfully.
            }
        }
    }


//Register the receiver.

 @Override
    protected void onResume()
    {
        super.onResume();
        registerReceiver(receiver, filter);
    }

//Send the request.
MobileCC.getInstance().cancelQueue();
```

5.5 getChannelInfo

Interface Description

This interface is used to obtain the video stream information during a call, including the resolution, frame rate, bit rate, packet loss rate, delay, and jitter.

Usage Description

When this interface is invoked, the eSDK layer reports CALL_MSG_GET_VIDEO_INFO. You can enable the listening function for this notification on the application so that the application can process the message.

Method Definition

```
//java code
public boolean getChannelInfo()
```

Parameter Description

None

Return Value

Type	Description
boolean	• **true**: invoked successfully • **false**: failed to be invoked

Example

```
//java code
//Set the filter.
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.NotifyMessage.CALL_MSG_GET_VIDEO_INFO);

//Initialize the receiver.
 private BroadcastReceiver receiver = new BroadcastReceiver()
   {
       @Override
       public void onReceive(Context context, Intent intent)
       {
           String action = intent.getAction();
           BroadMsg broadMsg = (BroadMsg) intent
              .getSerializableExtra(NotifyMessage.CC_MSG_CONTENT);

           if (NotifyMessage.NotifyMessage.CALL_MSG_GET_VIDEO_INFO.equals(action))
           {
             StreamInfo streamInfo = broadMsg.getStreamInfo();
               if (streamInfo == null)
               {
                   //No video stream
               }
               else
               {
                   //Resolution
                   videoSendFramsize = streamInfo.getEncoderSize();
                   //Frame rate
                   videoSendFrameRate = streamInfo.getSendFrameRate() + "";
                 //Bit rate
                   videoSendDatarate.setLength(0);
                   videoSendDatarate.append(streamInfo.getVideoSendBitRate() / 1000);
                   videoSendDatarate.append('k');
                   //Packet loss rate
                   videoSendPacketLossProbability =
Float.valueOf(streamInfo.getVideoSendLossFraction()).intValue() + "";
                   videoRecvPacketLossProbability =
Float.valueOf(streamInfo.getVideoRecvLossFraction()).intValue() + "";
                   //Delay
                   videoSendDelay =
Float.valueOf(streamInfo.getVideoSendDelay()).intValue() + "";
                   videoRecvDelay =
Float.valueOf(streamInfo.getVideoRecvDelay()).intValue() + "";
                   //Jitter
```

```
                videoSendJitter =
Float.valueOf(streamInfo.getVideoSendJitter()).intValue() + "";
                videoRecvJitter =
Float.valueOf(streamInfo.getVideoRecvJitter()).intValue() + "";

                /**Received part**/
                //Frame rate
                videoRecvFrameRate = streamInfo.getRecvFrameRate() + "";
                //Resolution
                videoRecvFramsize = streamInfo.getDecoderSize();
                //Bit rate
                videoRecvDatarate.setLength(0);
                videoRecvDatarate.append(streamInfo.getVideoRecvBitRate() / 1000);
                videoRecvDatarate.append('k');
                Toast.makeText(ChatActivity.this, "Sending resolution:" +
videoSendFramsize + ",frame rate:" + videoSendFrameRate + ",bit rate:" +
videoSendDatarate
                        + ",packet loss rate:" + videoSendPacketLossProbability +
"%,delay:" + videoSendDelay + "ms,jitter:" + videoSendJitter + "\n"
                        + "Receiving resolution:" + videoRecvFramsize + ",frame rate:"
+ videoRecvFrameRate + ",bit rate:" + videoRecvDatarate
                        + ",packet loss rate:" + videoRecvPacketLossProbability +
"%,delay:" + videoRecvDelay + "ms,jitter:" + videoRecvJitter,
Toast.LENGTH LONG).show();
            }
          }
       }
   }


//Register the receiver.

 @Override
   protected void onResume()
   {
      super.onResume();
      registerReceiver(receiver, filter);
   }

//Send the request.
MobileCC.getInstance().getChannelInfo();
```

6 Setting Interfaces

6.1 getVerifyCode

Interface Description

This interface is used to obtain the Base64 string of verification code, which is parsed to a graphic verification code.

Method Definition

```
//java code
public void getVerifyCode()
```

Parameter Description

None

Return Value

NotifyMessage.CALL_MSG_ON_VERIFYCODE contains the Base64 string.

Example

```
//java code
 private BroadcastReceiver receiver = new BroadcastReceiver()
    {
        @Override
        public void onReceive(Context context, Intent intent)
        {
            String action = intent.getAction();
            BroadMsg broadMsg = (BroadMsg) intent
                   .getSerializableExtra(NotifyMessage.CC_MSG_CONTENT);

            if (broadMsg != null)
            {

                if (NotifyMessage.CALL_MSG_ON_VERIFYCODE.equals(action))
                {
???? if (null == broadMsg.getRetCode())
                    {
                        Toast.makeText(MainActivity.this,
getString(R.string.get_varifycode_fail) + broadMsg.getErrorCode(),
Toast.LENGTH_SHORT).show();
                    }
                    else
                    {
                        String retcode = broadMsg.getRetCode();

                        if ("0".equals(retcode))
                        {
                            //The verification code is received.
                            String verifyCode = broadMsg.getMsg();

                          Message message = new Message();
                            message.what = GET VERIFYCODE;
                            message.obj = verifyCode;
                            handler.sendMessage(message);
                        }
                        else
                        {
                            //The verification code is not received.
                            Toast.makeText(MainActivity.this,
getString(R.string.get varifycode fail) + retcode, Toast.LENGTH SHORT).show();
                        }
                    }

                }
            }
        }
    };

private Handler handler = new Handler()
    {
```

```
      @Override
      public void handleMessage(Message msg)
      {
          switch (msg.what)
          {
              case GET_VERIFYCODE:
                  String verifyValue = (String)msg.obj;
                  Bitmap bitmap = base64ToBitmap(verifyValue);
                  imageVerifycode.setImageBitmap(bitmap);

              default:
                  break;
          }
      }
   };
MobileCC.getInstance().getVerifyCode();
```

6.2 getVolume

Interface Description

This interface is used to obtain the speaker volume.

Method Definition

```
//java code
public int getVolume()
```

Parameter Description

None

Return Value

Type	Description
int	Indicates the current volume, ranging from 0 to 100.

Example

```
//java code
MobileCC.getInstance().getSpeakVolume();
```

6.3 changeAudioRoute

Interface Description

This interface is used to switch the voice between the speaker and receiver when the call is set up.

Prerequisites

An audio call is set up.

Method Definition

```
//java code
public boolean changeAudioRoute(int route)
```

Parameter Description

Parameter	Type	Description
route	int	Indicates the voice route. **MobileCC.AUDIO_ROUTE_SPEAKER**: speaker **MobileCC.AUDIO_ROUTE_RECEIVER**: receiver

Return Value

Type	Description
boolean	• **true**: invoked successfully • **false**: failed to be invoked

Example

```
//java code
//Turn on the speaker.
MobileCC.getInstance().changeAudioRoute(MobileCC.AUDIO_ROUTE_SPEAKER);
```

6.4 setVideoContainer

Interface Description

This interface is used to set the local and remote video container after a conference is set up to display the video.

Usage Description

This interface is invoked after the video is started.

Prerequisites

A conference is set up.

Method Definition

```
//java code
public void setVideoContainer(Context context, ViewGroup localView,ViewGroup
remoteView)
```

Parameter Description

Parameter	Type	Description
context	Context	Indicates the context.
localView	ViewGroup	Indicates the local video container.
remoteView	ViewGroup	Indicates the remote video container.

Return Value

None

Example

```
//java code
//Define the container.
private LinearLayout mLlRemoteSurface;
private LinearLayout mLlLocalSurface;

mLlRemoteSurface = (LinearLayout) findViewById(R.id.view remote);   mLlLocalSurface
= (LinearLayout) findViewById(R.id.view local);


//Invoke the interface to load the video.
MobileCC.getInstance().setVideoContainer(MeetingActivity.this, mLlLocalSurface,
mLlRemoteSurface);
```

6.5 switchCamera

Interface Description

This interface is used to switch between the front-facing camera and rear-facing camera after the camera has been turned on.

Prerequisites

The camera has been turned on in a video conference.

Method Definition

```
//java code
public void switchCamera()
```

Parameter Description

None

Return Value

None

Example

```
//java code
MobileCC.getInstance().switchCamera();
```

6.6 setVideoRotate

Interface Description

This interface is used to adjust the viewing angle after the local video is displayed.

Prerequisites

A video conference is set up and the video is displayed.

Method Definition

```
//java code
public  boolean setVideoRotate(int rotate)
```

Parameter Description

Parameter	Type	Description
rotate	int	Indicates the angle of counterclockwise rotation: **90**: Rotate 90 degrees in a counterclockwise direction. **180**: Rotate 180 degrees in a counterclockwise direction. **270**: Rotate 270 degrees in a counterclockwise direction.

Return Value

Type	Description
boolean	• **true**: success • **false**: failure

Example

```
//java code
//The local video rotates 90 degrees in counterclockwise direction.
MobileCC.getInstance().setVideoRotate(90);
```

6.7 setDataRate

Interface Description

This interface is used to set the bandwidth. The bandwidth determines the video quality.

Prerequisites

Set the bandwidth before making the call.

Method Definition

```
//java code
public boolean setDataRate(int bandwidth)
```

Parameter Description

Parameter	Type	Description
bandwidth	int	Indicates the bandwidth, ranging from 1 to 768. A higher bandwidth indicates a better video quality.

Return Value

Type	Description
boolean	• **true**: success • **false**: failure

Example

```
//java code
if (MobileCC.getInstance().setDataRate(256))
 {
    //The bandwidth is set successfully.
 }
```

6.8 setVideoMode

Interface Description

This interface is used to set the video mode. The video mode determines the video fluency.

Method Definition

```
//java code
public int setVideoMode(int videoMode)
```

Parameter Description

Parameter	Type	Description
videoMode	int	Indicates the video mode. **VIDEOMODE_QUALITY** (default): The video clarity is preferred. MobileCC.VIDEOMODE_FLUENT: The video fluency is preferred.

Return Value

Type	Description
int	• **0**: success • **-1**: failure

Example

```
//java code
MobileCC.getInstance().setVideoMode(MobileCC.VIDEOMODE_QUALITY)
```

6.9 setSpeakerMute

Interface Description

This interface is used to mute and unmute the speaker during a call.

Prerequisites

A call is set up and the speaker is used to play the voice.

Method Definition

```
//java code
public boolean setSpeakerMute(boolean isMute)
```

Parameter Description

Parameter	Type	Description
isMute	boolean	**true**: Mute the speaker. **false**: Unmute the speaker.

Return Value

Type	Description
boolean	• **true**: success • **false**: failure

Example

```
//java code
MobileCC.getInstance().setSpeakerMute(true);//Mute the speaker
```

6.10 setMicMute

Interface Description

This interface is used to mute and unmute the microphone during a voice call.

Method Definition

```
 //java code
public boolean setMicMute(boolean isMute)
```

Parameter Description

Parameter	Type	Description
isMute	boolean	**true**: Mute the microphone. **false**: Unmute the microphone.

Return Value

Type	Description
boolean	• **true**: success • **false**: failure

Example

```
//java code
if(MobileCC.getInstance().setMicMute(true))
{
 //Mute the microphone.
}
```

6.11 videoOperate

Interface Description

This interface is used to ensure that the application works properly when it is switched to the background during a video call.

Method Definition

```
//java code
public void videoOperate(int operation)
```

Parameter Description

Parameter	Type	Description
operation	int	Indicates starting or stopping the video. **MobileCC.START**: start **MobileCC.STOP**: stop

Return Value

None

Example

```
//java code
@Override
   protected void onResume()
   {
      super.onResume();
      registerReceiver(receiver, filter);
```

```
    //Invoke the videoOperate to start the video.
    MobileCC.getInstance().videoOperate(MobileCC.START);
}

@Override
protected void onPause()
{
    super.onPause();
    unregisterReceiver(receiver);
    //Invoke the videoOperate to stop the video.
    MobileCC.getInstance().videoOperate(MobileCC.STOP);
}
```

7 Error Codes

Error Code	Description
0	Success.
-1	Invalid parameter.
-2	Not logged in.
-3	The voice call is not connected.
-4	Incorrect server response.
-5	Network failure.
10-100-002	Incorrect settings of EVENT_METHOD in platform.properties on the server.
10-100-003	Not connected to the CCS. The WAS service is unavailable.
10-100-004	WebmAnyService is unavailable.
10-100-005	No permission to invoke the interface.
10-100-006	The user has not logged in.
10-100-007	Null or invalid request parameter.
10-100-008	The user has logged in.
10-100-009	Unavailable resource.
10-100-010	The method is not supported.
10-100-011	Status error.
10-100-012	User WebmServer is null.
10-100-013	The VDN ID does not exist.
10-100-014	The access code does not exist.
10-100-015	The number of logged-in users has reached the maximum.
10-100-016	The configuration agent is null.

Error Code	Description
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 fails.
10-200-006	Not in a queue.
10-200-007	The call is not set up.
10-200-008	Invalid verification code.
10-200-009	Failed to generate the verification code.
10-300-001	A conference in progress or in request.
10-300-002	The conference does not exist.

8 Change History

Release Date	Issue	Description
2016-12-31	V2.1.00.00	This is the first official release.