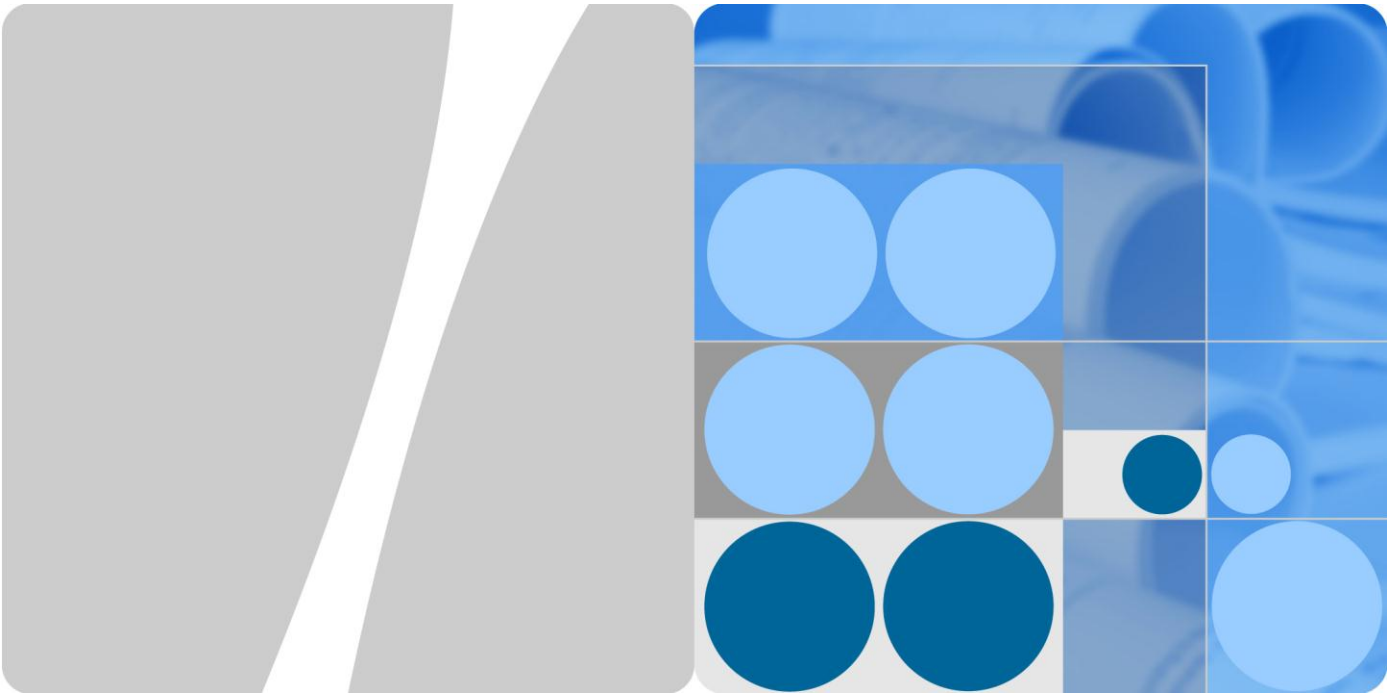

eSDK ICP V200R001C10 接口参考 01(CC, Android)

文档版本 01

发布日期 2017-03-17

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 概述

目的

本文主要介绍 eSDK ICP Android 接口参考。介绍内容包括相关数据类型、接口功能、方法定义、参数描述和使用示例等。

版本配套

本文的产品配套关系如下图所示：

产品名称	版本
eSDK ICP CC	V200R001C10
TUP 库	V600R006C00B022

技术支持

- 开发过程中，您有任何问题可以在 DevCenter 系统中提单跟踪。
- 如果您在使用过程中有任何疑问，都可以通过以下方式联系我们。
  a. 华为技术支持热线电话：400-8828-000。
  b. 华为技术支持邮箱：esdk@huawei.com。

2 事件通知

在实际调用各接口后服务器会返回以下消息，通过这些消息，可执行下一步的操作。

消息 ID	说明
AUTH_MSG_ON_LOGIN	登录事件，返回值为 0 表示登录成功，其它表示失败
AUTH_MSG_ON_LOGOUT	注销事件，返回值为 0 表示登出成功，其它表示失败
CALL_MSG_ON_CONNECTED	与坐席建立连接，广播携带信息 Msg，1 表示文字 2 表示语音。返回 retCode 值为 0 表示成功，其它表示失败，具体失败原因可查看错误码
CALL_MSG_ON_DISCONNECTED	与坐席断开连接，广播携带信息 Msg，1 表示文字 2 表示语音
CALL_MSG_ON_QUEUE_INFO	排队信息，返回值为 0 表示成功，其它表示失败，具体失败原因可查看错误码。通过 position 字段查看当前位置，totalWaitTime -累计排队时长
CALL_MSG_ON_QUEUE_TIMEOUT	排队超时
CALL_MSG_ON_QUEUING	正在排队
CALL_MSG_ON_CANCEL_QUEUE	取消排队事件
CHAT_MSG_ON_SEND	发送消息，返回值为 0 表

消息 ID	说明
	示成功，其它表示失败，具体失败原因可查看错误码
CHAT_MSG_ON_RECEIVE	接收消息，返回值 retcode 为 0 表示成功，其它表示失败，具体失败原因可查看错误码
CALL_MSG_REFRESH_LOCALVIEW	显示本地视频
CALL_MSG_REFRESH_REMOTEVIEW	显示对端视频
CALL_MSG_ON_DROPCALL	释放呼叫。返回值为 0 表示成功，其它表示失败，具体失败原因可查看错误码
CALL_MSG_ON_FAIL	呼叫失败
CALL_MSG_ON_CALL_END	呼叫断开
CALL_MSG_ON_SUCCESS	语音接通
CALL_MSG_ON_CONNECT	获取文字或者语音能力，具体通过携带的 Type 值来判断，1 是文字，2 是语音。
CALL_MSG_USER_JOIN	加入会议通知
CALL_MSG_USER_END	会议终止通知
CALL_MSG_USER_STATUS	会议创建通知
CALL_MSG_GET_VIDEO_INFO	获取到视屏流信息
CALL_MSG_USER_NETWORK_ERROR	会议断线通知
CALL_MSG_USER_RECEIVE_SHARED_DATA	远端共享数据
CALL_MSG_ON_NET_QUALITY_LEVEL	网络状态
CALL_MSG_ON_STOP_MEETING	退出会议
CALL_MSG_ON_APPLY_MEETING	申请会议
CALL_MSG_ON_POLL	轮询，返回-5 表示网络超时
CC_MSG_CONTENT	广播标记，具体参考 Demo 使用

3 初始化类接口

3.1 getVersion（获取 SDK 版本信息）

接口描述

获取当前 SDK 版本信息。

方法定义

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

参数描述

无

返回值

- String：版本号

使用示例

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

3.2 initSDK（初始化 SDK）

接口描述

对用 TUP 以及会议组件进行初始化。

使用说明

在程序的 Application 中的 onCreate()方法内，且在 setLog(String path, int level)接口之后调用。

方法定义

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

参数描述

参数	类型	描述
app	Application	工程的 Application，用于启动 SDK 服务。

返回值

无。

使用示例

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

   }
   ......
}
```

3.3 unInitSDK（去初始化 SDK）

接口描述

用于应用正常结束时，将相关模块去初始化，释放资源。

使用说明

在程序 Application 中的 onTerminate()方法中调用此接口。

方法定义

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

参数描述

无

返回值

无

使用示例

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

   @Override
   public void onTerminate()
   {
      super.onTerminate();
      MobileCC.getInstance().unInitSDK(); // 停止 SDK 服务
   }
}
```

3.4 setLog（设置日志路径和级别）

接口描述

用于初始化 TUP/HME 日志。

方法定义

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

参数描述

参数	类型	描述
path	String	日志路径。长度为 60
level	int	日志等级（1-3）。

返回值

- 0 ： 设置成功
- -1： 参数有误

使用示例

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

3.5 setServerCertificateValidation（校验服务器证书）

接口描述

对服务器证书进行校验。

如果需要证书验证功能，需要将证书放在工程中 assets 文件夹下，例如“assets/certs/server.cer”。

使用说明

在网络请求中，开启校验服务器证书，如果证书校验失败，会导致访问服务器失败。

方法定义

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

参数描述

参数	类型	描述
needValidate	boolean	是否验证证书
needValidateDomain	boolean	是否验证域名
certInputStream	InputStream	证书

返回值

无。

使用示例

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

} catch (IOException ioe)
{
   LogUtil.d(TAG, "io exception";
}
finally
{
   if (inputStream != null)
   {
      try
      {
         inputStream.close();
      }
      catch (IOException e)
      {
         LogUtil.d(TAG, "io exception" );
      }
   }
}
```

3.6 setHostAddress（设置接入网关地址）

接口描述

用于设置应用访问后台服务器的 URL、虚拟呼叫中心 ID 和用户名。第三个参数需要根据具体的服务器的配置来设置相应的参数。

使用说明

调用此接口，应对返回值做判断，再做下一步操作。

方法定义

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

参数描述

参数	类型	描述
IPStr	String	服务器 IP 地址 如：10.66.110.253
portStr	String	服务器端口号 如：8080
transSecurity	boolean	认证方式 • true：HTTPS • false：HTTP
sipServerType	int	平台类型 • MobileCC.SERVER_MS：MS 平台 • MobileCC.SERVER_TP：TP 平台

返回值

- 0 ： 设置成功
- -1： 参数有误

使用示例

```
 //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（设置 SIP 服务器地址）

接口描述

用于设置语音呼叫地址和端口。TP 环境自动获取。

方法定义

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

参数描述

参数	类型	描述
IPStr	String	呼叫地址 如：10.66.110.253
portStr	String	端口号 如：8080

返回值

- 0 ： 设置成功
- -1： 参数有误

使用示例

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

3.8 setTransportSecurity（设置数据加密模式）

接口描述

用于设置加密模式，需要根据当前服务器的配置来做相应的设置。

方法定义

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

参数描述

参数	类型	描述
enableTLS	boolean	是否开启 TLS 加密

参数	类型	描述
enableSRTP	boolean	是否开启 SRTP 加密

返回值

无

使用示例

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

4 认证类接口

所属类：MobileCC

4.1 login（登录）

接口描述

用于登录服务器。

使用说明

在登录之前，需要通过 3.6 setHostAddress（设置接入网关地址）设置服务器地址和端口。

返回值只能说明接口是否调用成功，登录之前注册通知，是否成功登录要根据通知 2 事件通知判断。MS 支持 NAT 穿越功能。

方法定义

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

参数描述

参数	类型	描述
vdnId	String	数字类型码，默认填“1”。
userName	String	用户帐号，如“test”。

返回值

- 0： 调用成功
- -1：参数有误

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//设置 filter
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.AUTH MSG ON LOGIN);

//初始化 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.getRequestCode().getRetCode()))
            {
               //登录成功
            }
         }
      }
   };

//监听广播
localBroadcastManager.registerReceiver(receiver, filter);

//登录操作
 if (0 != MobileCC.getInstance().login("1", etName.getText()
               .toString().trim()))
   {
      //登录请求已发出
   }
```

4.2 logout（登出）

接口描述

用于登出服务器。

使用说明

登出之前注册通知，是否成功登出要根据通知 2 事件通知判断。

方法定义

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

参数描述

无

返回值

无

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//设置 filter
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.AUTH_MSG_ON_LOGOUT);

//初始化 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.getRequestCode().getRetCode()))
             {
                //注销成功
             } else
             {
                //注销失败
             }
          }
       }
    };

//注册
receiverlocalBroadcastManager.registerReceiver(receiver, filter);

 @Override
```

```
public void onClick(View view)
  {
      switch (view.getId())
      {
          case R.id.btn_exit:
              //点击操作
              MobileCC.getInstance().logout();
              break;

          default:
              break;
      }
   }
```

5 呼叫类接口

5.1 文字交谈

5.1.1 webChatCall（发起文字交谈）

接口描述

用于第三方应用发起文字交谈呼叫。

使用说明

在 MS 平台中，当发起呼叫请求后，当坐席处于忙碌时，用户会先进入排队，eSDK 层会上报名为 2 事件通知的事件通知；当坐席接听后，通话建立，此时 eSDK 层会上报名为 2 事件通知的事件通知，应用可以增加对此通知的监听，从而处理相应获取呼叫事件。若要关闭验证码功能，可以在 IcsGateway 服务器中打开 home/prometheus/tomcat7/webapps/icsgateway/WEB-INF/config/verifycode.properties 这个文件，修改 VERIFYCODE_ISUSERFORCALL = false ，然后重启 IcsGateway 服务器即可。函数最后一个参数只需要传入空字符就行。

方法定义

```
//java code
public int webChatCall(String accessCode, String callData, String varifyCode)
```

参数描述

参数	类型	描述
accessCode	String	接入码，由平台配置。
callData	String	呼叫随路数据。

参数	类型	描述
varifyCode	String	验证码,由获取验证码接口获得

返回值

- 0：成功
- -1：参数检验失败

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//创建 filter
IntentFilter filter = new IntentFilter();
filter.addAction(NotifyMessage.CALL MSG ON CONNECTED);
filter.addAction(NotifyMessage.CALL MSG ON QUEUING);

//初始化 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))
            {

               if
(MobileCC.MESSAGE TYPE TEXT.equals(broadMsg.getRequestCode().getRetCode()))
                {
                   //与坐席连接成功，可以与坐席文字聊天
                }
            }
            else if (NotifyMessage.CALL MSG ON QUEUING.equals(action))
            {
               //正在排队，此处可以调用获取排队信息接口查询排队信息
            }
         }
    }


//注册 receiver

   @Override
    protected void onResume()
    {
```

```
      super.onResume();
      localBroadcastManager.registerReceiver(receiver, filter);    }

//发出文字交谈请求
 if (0 != MobileCC.getInstance().webChatCall("60011", "data", ""))
  {
      //请求已发出
  }
```

5.1.2 sendMsg（发送文字消息）

接口描述

该接口目前只支持 MS 平台，用于用户发送文本消息给坐席（可用于功能叠加，如在语音、视频或数据上叠加此功能）。

使用说明

在用户登录成功，且建立好文字连接时，可以调用此接口用于向坐席侧发送文字。此时 eSDK 层会上报名为 2 事件通知的事件通知，应用可以增加对此通知的监听，从而处理相应的事件。当坐席端向用户发送消息的时候，用户会收到 2 事件通知的事件通知，应用可以增加对此通知的监听，从而处理相应的事件。

方法定义

```
//java code
public int sendMsg (String content)
```

参数描述

参数	类型	描述
content	String	发送的文本内容,长度为 300 字符以内

返回值

0：接口调用成功

-1：接口调用失败

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//设置 filter
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.CHAT_MSG_ON_RECEIVE);
filter.addAction(NotifyMessage.CHAT_MSG_ON_SEND);
```

```
//初始化 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.CHAT_MSG_ON_SEND.equals(action))
         {
            if (null == broadMsg.getRequestCode().getRetCode())
            {
               //retcode 为空时
               Toast.makeText(MSChatActivity.this, "文字发送失败，错误码码是：" +
broadMsg.getRequestCode().getErrorCode(), Toast.LENGTH_LONG).show();
            }
            else
            {
               //retcode 不为为空时
               String retcode = broadMsg.getRequestCode().getRetCode();
               if (MobileCC.MESSAGE OK.equals(retcode))
               {
                  //消息发送成功
               }
               else
               {
                  //消息发送未成功
               }
            }
         }
        else if (NotifyMessage.CHAT MSG ON RECEIVE.equals(action))
         {
            //收到消息，消息内容是 broadMsg.getMsg()
         }
      }
   }


//注册 receiver

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

//发送操作
MobileCC.getInstance().sendMsg("test");
```

5.1.3 releaseWebChatCall（释放文字呼叫）

接口描述

释放文字连接，目前该接口只支持 MS 平台。

使用说明

当调用该接口时，eSDK 层会上报名为 2 事件通知的事件通知，应用可以增加对此通知的监听，从而处理相应的事件。

前提条件

文字已连接成功。

方法定义

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

参数描述

无

返回值

无

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//创建 filter
IntentFilter filter = new IntentFilter();
filter.addAction(NotifyMessage.CALL MSG ON DISCONNECTED);
filter.addAction(NotifyMessage.CALL MSG ON DROPCALL);

//初始化 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 DISCONNECTED.equals(action))
         {
             if
(MobileCC.MESSAGE_TYPE_TEXT.equals(broadMsg.getRequestInfo().getMsg()))
```

```
                {
                    //文字连接释放成功
                }
             }
            else if (NotifyMessage.CALL_MSG_ON_DROPCALL.equals(action))
            {
               //连接释放失败
            }

         }
    }


//注册 receiver

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

//发出请求
MobileCC.getInstance().releaseWebChatCall();
```

5.2 呼叫

所属类：MobileCC

如果出现语音连上就立即断开的情况，请到路由器的配置中，在“WAN 设置”中把 SIP ALG 开启。

5.2.1 makeCall（发起语音/视频呼叫）

接口描述

用于用户发起语音/视频呼叫。MS 发起语音呼叫后，可进行功能叠加，如文字、视频和数据。

使用说明

当调用该接口时，会收到以下几个通知：

- 2 事件通知：与坐席接通
- 2 事件通知：正在排队
- 2 事件通知：语音接通
- 2 事件通知：视屏呼叫后，申请会议成功
- 2 事件通知：加入会议成功

关闭验证码功能时，最后一个参数只需要传入空字符就行。

视频呼叫过程中，当前 Activity 设置为横屏，也就是 android:screenOrientation="landscape"，展现的视频就是横的，Actvity 设置为竖屏，也就是 android:screenOrientation="portrait"，展现的视频就是竖的。视频通话期间，可以通过设置视频角度的接口来做相应调整。

前提条件

应用已登录成功。

方法定义

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

参数描述

参数	类型	描述
accessCode	String	接入码
callType	String	呼叫类型： MobileCC.AUDIO_CALL：语音 MobileCC.VIDEO_CALL：视频
callData	String	呼叫随路数据
varifyCode	String	验证码

返回值

- 0：成功
- -1：参数校验失败

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//设置 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);
filter.addAction(NotifyMessage.CALL MSG ON APPLY MEETING);

//初始化 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))
          {
              //与坐席接通
          }
         else if (NotifyMessage.CALL_MSG_ON_QUEUING.equals(action))
          {
              //正在排队

          }
         else if (NotifyMessage.CALL_MSG_ON_SUCCESS.equals(action))
          {
              //语音接通
          }
         else if(NotifyMessage.CALL_MSG_ON_APPLY_MEETING.equals(action))
          {
              //申请会议成功
          }
      }
   }


//注册 receiver

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

//呼叫操作

MobileCC.getInstance().makeCall("60021", MobileCC.VIDEO_CALL, "data",  " ");
```

5.2.2 releaseCall（结束呼叫）

接口描述

释放视频呼叫。

使用说明

当匿名呼叫被释放后，eSDK 层会上报名为 2 事件通知的事件通知，应用可以增加对此通知的监听，从而处理相应呼叫释放事件。

前提条件

语音/视频呼叫已建立。

方法定义

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

参数描述

无

返回值

无

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//设置 filter
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.NotifyMessage.CALL_MSG_ON_DISCONNECTED);

//初始化 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))
         {
          //TP 中收到此事件说明与坐席断开。MS 中分以下情况：
              if
(MobileCC.MESSAGE TYPE TEXT.equals(broadMsg.getRequestInfo().getMsg()))
              {
                 //文字断开
              }
              else if
(MobileCC.MESSAGE TYPE AUDIO.equals(broadMsg.getRequestInfo().getMsg()))
              {
                 //语音断开
              }
         }
      }
   }
```

```
//注册 receiver

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

//发送操作
MobileCC.getInstance().releaseCall();
```

5.2.3 getCallQueueInfo（获取排队信息）

接口描述

用户在发起呼叫进入排队后，通过该接口获取当前排队的位置等信息。

使用说明

当查询结果返回时，eSDK 层会上报名为 2 事件通知的响应通知，应用可以增加对此通知的监听，从而处理相应获取排队信息事件。

返回结果保存在 notifyMsg 中，其结构参考下表。

Key	说明
position	本呼叫在队列中的位置
onlineAgentNum	在线坐席数
longestWaitTime	最长等待时间

前提条件

1、应用已登录。

2、呼叫已进入排队状态。

方法定义

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

参数描述

无。

返回值

无

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//设置 filter
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.NotifyMessage.CALL MSG ON QUEUE INFO);

//初始化 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.getRequestCode().getRetCode())
            {
               //获取排队信息失败，通过 broadMsg.getRequestCode().getErrorCode()看错误
码              }
              else
              {
                  String retcode = broadMsg.getRequestCode().getRetCode();
                  if (MobileCC.MESSAGE_OK.equals(retcode))
                  {
                  //处于排队状态 通过 broadMsg.getQueueInfo().getPosition()查看前面人数
                  }
                  else
                  {
                    //非排队状态
                  }
              }


         }
      }
   }


//注册 receiver

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

//发送操作
MobileCC.getInstance().getCallQueueInfo();
```

5.2.4 cancelQueue（取消排队）

接口描述

用于用户取消在当前队列的排队。

使用说明

当查询结果返回时，eSDK 层会上报名为 2 事件通知的响应通知，应用可以增加对此通知的监听，从而处理相应取消排队事件。

前提条件

1、应用已登录。

2、呼叫已进入排队状态。

方法定义

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

参数描述

无

返回值

无

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//设置 filter
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.NotifyMessage.CALL MSG ON CANCEL QUEUE);

//初始化 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))
         {
           //取消排队成功
```

```
            }
        }
    }


//注册 receiver

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

//发送操作
MobileCC.getInstance().cancelQueue();
```

5.2.5 updateToVideo（升级语音呼叫到视频呼叫）

接口描述

该接口目前只支持 MS 平台，用于用户从语音通话升级到视频通话。本质上是实现在语音的基础上叠加 MS 的视频会议功能。

使用说明

当调用该接口时，会收到以下通知：

2 事件通知：申请会议

2 事件通知：创建会议成功

2 事件通知：加入会议成功

方法定义

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

参数描述

无。

返回值

- 0：成功
- -3：语音未连接

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
```

```
//设置 filter
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.CALL_MSG_ON_APPLY_MEETING);
filter.addAction(NotifyMessage.CALL_MSG_USER_STATUS);

//初始化 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_APPLY_MEETING.equals(action))
         {
            //申请会议
             if (null == broadMsg.getRequestCode().getRetCode())
             {
                //申请会议失败，错误码是 broadMsg.getRequestCode().getErrorCode()
             }
             else
             {
                String retcode = broadMsg.getRequestCode().getRetCode();

                if (MobileCC.MESSAGE OK.equals(retcode))
                {
                   //申请会议成功
                }
                else
                {
                   //申请会议失败，错误码是 retcode
                }
             }
         }
        else if(NotifyMessage.CALL MSG USER STATUS.equals(action))
         {
            //会议创建成功
         }
      }
   }

//注册 receiver

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


//升级视屏操作
```

```
MobileCC.getInstance().updateToVideo();
```

5.2.6 getChannelInfo（获取视屏流信息）

接口描述

用于在通话中获取分辨率，帧率、码率、丢包率、延迟、抖动等信息。

使用说明

当调用该接口时，eSDK 层会上报名为 2 事件通知的事件通知，应用可以增加对此通知的监听，从而获取相应的排队信息。

方法定义

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

参数描述

无

返回值

- true：调用成功
- false：调用失败

使用示例

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//设置 filter
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.NotifyMessage.CALL MSG GET VIDEO INFO);

//初始化 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.getRequestInfo().getStreamInfo();
            if (streamInfo == null)
            {
```

```
                //没有视屏流
            }
            else
            {
                //分辨率
                videoSendFramsize = streamInfo.getEncoderSize();
                //帧率
                videoSendFrameRate = streamInfo.getSendFrameRate() + "";
                //码率
                videoSendDatarate.setLength(0);
                videoSendDatarate.append(streamInfo.getVideoSendBitRate() / 1000);
                videoSendDatarate.append('k');
                //丢包率
                videoSendPacketLossProbability =
Float.valueOf(streamInfo.getVideoSendLossFraction()).intValue() + "";
                videoRecvPacketLossProbability =
Float.valueOf(streamInfo.getVideoRecvLossFraction()).intValue() + "";
                //延迟
                videoSendDelay =
Float.valueOf(streamInfo.getVideoSendDelay()).intValue() + "";
                videoRecvDelay =
Float.valueOf(streamInfo.getVideoRecvDelay()).intValue() + "";
                //抖动
                videoSendJitter =
Float.valueOf(streamInfo.getVideoSendJitter()).intValue() + "";
                videoRecvJitter =
Float.valueOf(streamInfo.getVideoRecvJitter()).intValue() + "";

                /**接收部分**/
                // 帧率
                videoRecvFrameRate = streamInfo.getRecvFrameRate() + "";
                // 分辨率
                videoRecvFramsize = streamInfo.getDecoderSize();
                // 码率
                videoRecvDatarate.setLength(0);
                videoRecvDatarate.append(streamInfo.getVideoRecvBitRate() / 1000);
                videoRecvDatarate.append('k');
                Toast.makeText(ChatActivity.this, "发送分辨率:" + videoSendFramsize
+ ",帧率:" + videoSendFrameRate + ",码率:" + videoSendDatarate
                        + ",丢包率:" + videoSendPacketLossProbability + "%,延迟:" +
videoSendDelay + "ms,抖动:" + videoSendJitter + "\n"
                        + "接收分辨率:" + videoRecvFramsize + ",帧率:" +
videoRecvFrameRate + ",码率:" + videoRecvDatarate
                        + ",丢包率:" + videoRecvPacketLossProbability + "%,延迟:" +
videoRecvDelay + "ms,抖动:" + videoRecvJitter, Toast.LENGTH_LONG).show();
            }
          }
       }
   }


//注册 receiver

  @Override
   protected void onResume()
```

```
    {
        super.onResume();
        localBroadcastManager.registerReceiver(receiver, filter);    }

//发送操作
MobileCC.getInstance().getChannelInfo();
```

6 设置类接口

6.1 getVerifyCode（获取验证码）

接口描述

用于获取服务器传来的验证码的 Base64 字符串，需要用户解析成验证码图片。

若不需要验证码，可以在 IcsGateway 服务器中打开 home/prometheus/tomcat7/webapps/icsgateway/WEB-INF/config/verifycode.properties 这个文件，修改 VERIFYCODE_ISUSERFORCALL = false ，然后重启 IcsGateway 服务器即可。

方法定义

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

参数描述

无。

返回值

2 事件通知事件里面包含 Base64 字符串。

使用示例

```
 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))
                    {
                       //收到验证码
                       String verifyCode = broadMsg.getMsg();

                       Message message = new Message();
                       message.what = GET_VERIFYCODE;
                       message.obj = verifyCode;
                       handler.sendMessage(message);
                    }
                    else
                    {
                       //没有收到验证码
                       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 getSpeakerVolume（获取扬声器音量）

接口描述

用于获取扬声器音量。

方法定义

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

参数描述

无。

返回值

- int：当前音量值，取值范围为[0,100]

使用示例

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

6.3 changeAudioRoute（扬声器/听筒模式切换）

接口描述

在语音接通的情况下，通过此接口可以实现切换听筒/扬声器作为播放设。

前提条件

语音已接通。

方法定义

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

参数描述

参数	类型	描述
route	int	语音路由 MobileCC.AUDIO_ROUTE_SPEAKER 扬声器 MobileCC.AUDIO_ROUTE_RECEIVER 听筒

返回值

- true：调用成功
- false：调用失败

使用示例

```
//java code
//打开扬声器
MobileCC.getInstance().changeAudioRoute(MobileCC.AUDIO_ROUTE_SPEAKER);
```

6.4 setVideoContainer（设置视频显示容器）

接口描述

会议建立之后，通过此接口来设置本地与远端视频容器，这样就可以在界面上显示本地和远端的视屏画面了。视频呼叫过程中，当前 Activity 设置为横屏，也就是 android:screenOrientation="landscape"，展现的视频就是横的，Actvity 设置为竖屏，也就是 android:screenOrientation="portrait"，展现的视频就是竖的。

前提条件

会议已建立，视频开启后调用该接口。

方法定义

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

参数描述

参数	类型	描述
context	Context	上下文
localView	ViewGroup	本地视屏容器
remoteView	ViewGroup	远端视屏容器

返回值

无

使用示例

```
//java code
//定义容器
private LinearLayout mLlRemoteSurface;
private LinearLayout mLlLocalSurface;

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


//调用接口加载视屏
MobileCC.getInstance().setVideoContainer(MeetingActivity.this, mLlLocalSurface,
mLlRemoteSurface);
```

6.5 setDesktopShareContainer（设置桌面共享显示容器）

接口描述

此接口目前只支持 MS 平台。会议过程中，通过此接口可以设置共享容器，以查看共享内容。

前提条件

MS 会议已建立。

方法定义

```
//java code
public void setDesktopShareContainer(Context context, ViewGroup sharedView)
```

参数描述

参数	类型	描述
context	Context	上下文
sharedView	ViewGroup	桌面共享容器

返回值

无

使用示例

```
//java code
//定义容器
private RelativeLayout desktopSharedLayout;
desktopSharedLayout = (RelativeLayout) findViewById(R.id.desktopSharedLayout);

//调用接口
MobileCC.getInstance().setDesktopShareContainer(getBaseContext(),
                    desktopSharedLayout);
```

6.6 switchCamera（切换前置/后置摄像头）

接口描述

在摄像头开启的的情况下，通过此接口可以实现摄像头的前置/后置转换。

前提条件

视屏状态下，摄像头已开启。

方法定义

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

参数描述

无

返回值

无

使用说明

无

使用示例

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

6.7 setVideoRotate（设置视频旋转角度）

接口描述

在本地视屏显示出来之后，可以通过本接口来调整本地视屏显示角度。

前提条件

视屏已接通并显示

方法定义

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

参数描述

参数	类型	描述
rotate	int	逆时针旋转角度 90：逆时针旋转 90 度 180：逆时针旋转 180 度 270：逆时针旋转 270 度

返回值

无

使用说明

无

使用示例

```
//java code
//本地视屏画面逆时针旋转 90
MobileCC.getInstance().setVideoRotate(90);
```

6.8 setDataRate（设置带宽）

接口描述

用于设置带宽。不同的带宽值所对应的画面效果是不一样的。

前提条件

TP 中是在呼叫之前设置。

MS 中是在会议建立之后调用。

方法定义

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

参数描述

参数	类型	描述
bandwidth	int	TP 环境：带宽值范围[1-768]。 1<=带宽值<=128 取 128K 128<带宽值<=256 取 256K 256<带宽值<=384 取 384K 384<带宽值<=512 取 512K 512<带宽值<=768 取 768K 数值越高，画面越清晰。 MS 环境：带宽值范围[1-768]。 带宽值小于 512K 表示画质流畅，大于 512K，表示画质越清晰

返回值

- true：成功
- false：失败

使用说明

无

使用示例

```
//java code
if (MobileCC.getInstance().setDataRate(256))
   {
     //带宽设置成功
   }
```

6.9 setVideoMode（设置视频显示模式）

接口描述

用于设置视频显示模式，不同的显示模式对应着不同的流畅度。

方法定义

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

参数描述

参数	类型	描述
videoMode	int	视频模式。 VIDEOMODE_QUALITY：画质优先（默认） MobileCC.VIDEOMODE_FLUENT：流畅优先

返回值

- 0：成功
- -1：设置失败

使用示例

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

6.10 setSpeakerMute（设置扬声器静音）

接口描述

在通话过程中，通过此接口来设置扬声器静音和恢复。

前提条件

语音已接通，使用扬声器作为放声设备。

方法定义

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

参数描述

参数	类型	描述
isMute	boolean	true：静音扬声器 false：恢复

返回值

- true：成功
- false：设置失败

使用示例

```
//java code
MobileCC.getInstance().setSpeakerMute(true);//静音扬声器
```

6.11 setMicMute（设置麦克静音）

接口描述

在语音通话的过程中，通过此接口可以实现麦克的静音和恢复。

方法定义

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

参数描述

参数	类型	描述

参数	类型	描述
isMute	boolean	true：设置静音 false：取消静音

返回值

- true：成功
- false：设置失败

使用示例

```
//java code
if(MobileCC.getInstance().setMicMute(true))
{
   //麦克静音
}
```

6.12 videoOperate（视屏控制）

接口描述

视屏过程中，通过设置该接口可以保证应用切到后台时不崩溃。（应用于 TP 平台）

方法定义

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

参数描述

参数	类型	描述
operation	int	开启或者停止。 MobileCC.START：开启 MobileCC.STOP：停止

返回值

无

使用示例

```
//java code
@Override
```

```
protected void onResume()
{
    super.onResume();
    registerReceiver(receiver, filter);
    //控制视屏--> 开启
    MobileCC.getInstance().videoOperate(MobileCC.START);
}

@Override
protected void onPause()
{
    super.onPause();
    unregisterReceiver(receiver);
    //控制视屏--> 停止
    MobileCC.getInstance().videoOperate(MobileCC.STOP);
}
```

7 错误码

错误码	说明
0	成功。
-1	参数错误。
-2	未登录。
-3	语音未连接。
-4	服务器响应错误。
-5	网络故障。
10-100-002	服务端 platform.properties 中的 EVENT_METHOD 配置错误。
10-100-003	没有连接到 CCS，Was 服务不可用。
10-100-004	WebmAnyService 服务不可用。
10-100-005	无权限调用接口。
10-100-006	用户未登陆，网络长时间断连后，后续的请求都会失败，返回该错误码。
10-100-007	用户请求参数为空或不合法。
10-100-008	用户已登录。
10-100-009	资源不可用。
10-100-010	方法不支持。
10-100-011	状态错误。
10-100-012	用户的 WebmServer 为空。
10-100-013	vdnId 不存在。
10-100-014	接入码不存在。

错误码	说明
10-100-015	已达到最大用户登录数。
10-100-016	配置代理为空。
10-100-017	配置代理发送异常。
10-200-001	接入码对应的呼叫已存在。
10-200-002	超过用户最大呼叫数。
10-200-003	呼叫不存在。
10-200-004	编码转换失败。
10-200-006	呼叫不是排队态。
10-200-007	呼叫未建立。
10-300-001	正在会议中或申请会议中。
10-300-002	会议不存在。

8 修订记录

发布日期	文档版本	描述
2017-3-15	V2.1.10	文档 V200R001C10 版本发布。 全量适配 MS&TP 功能。
2017-3-8	1.1.T1	文档 1.1.T1 版本发布。。 新增： • 5.1 文字交谈 • 5.2.5 updateToVideo（升级语音呼叫到视频呼叫） • 6.5 setDesktopShareContainer（设置桌面共享显示容器） • MS NAT 功能 修改： 2 事件通知
2016-12-31	V2.1.00	第一次正式发布。