

eSDK CloudVC 20.1.RC1 开发指南(Windows,JS)

eSDK CloudVC 20.1.RC1 开发指南 (Windows,JS)

文档版本 01
发布日期 2019-11-22



版权所有 © 华为技术有限公司 2019。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI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：<https://www.huawei.com>

客户服务邮箱：support@huawei.com

客户服务电话：4008302118

目 录

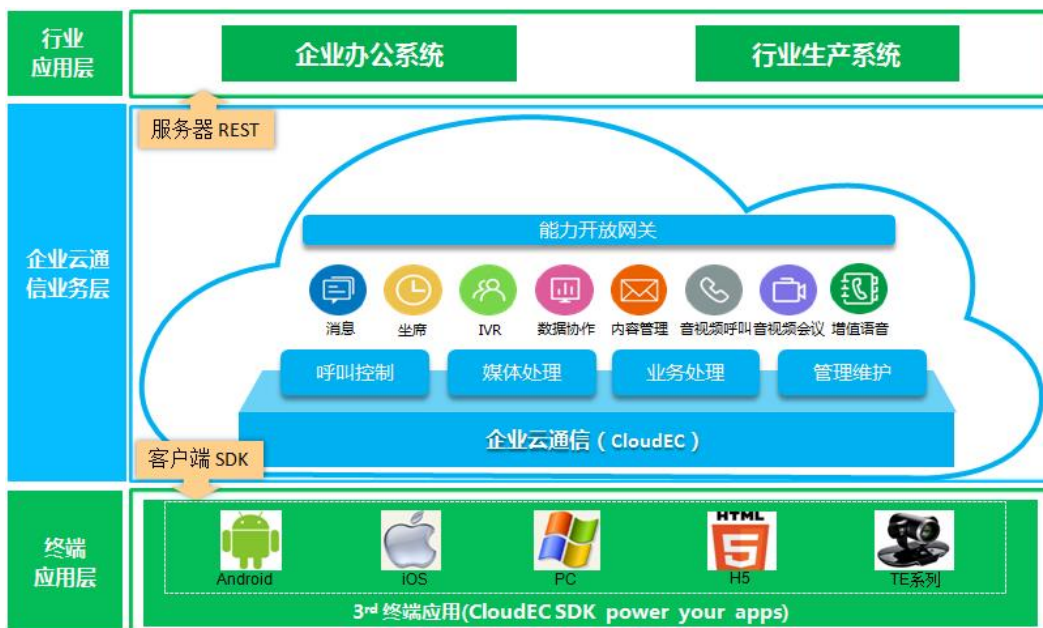
1 CloudVC SDK 简介.....	1
2 内容导航.....	4
3 相关资源.....	5
3.1 SDK 和文档下载路径.....	5
3.2 Sample Codes 下载路径.....	6
3.3 免费申请远程实验室.....	6
3.4 技术支持渠道.....	7
4 Hello World.....	8
4.1 概述.....	8
4.2 环境准备.....	8
4.3 HelloWorld.....	9
4.3.1 步骤 1 创建目录.....	9
4.3.2 步骤 2 创建工程.....	9
4.3.3 步骤 3 引入依赖.....	11
4.3.4 步骤 4 创建 tsdkClient 对象.....	11
4.3.5 步骤 5 登录 CloudVC 服务器.....	12
4.3.6 步骤 6 加入会议.....	13
4.3.7 步骤 7 登出 CloudVC 服务器.....	18
4.3.8 步骤 8 创建页面主体.....	18
4.4 运行 HelloWorld.....	19
4.4.1 步骤 1 预约会议.....	19
4.4.2 步骤 2 启动 TSDK 守护进程.....	20
4.4.3 步骤 3 WEB 服务器配置.....	20
4.4.4 步骤 4 访问 Hello_CloudEC 页面.....	21
4.4.5 步骤 5 加入会议.....	21
5 业务开发指导.....	22
5.1 业务开发总体流程.....	22
5.2 初始化.....	23
5.3 登录鉴权.....	26
5.4 创建会议.....	28
5.5 加入会议.....	30
5.6 基本会控.....	33

5.7 视频会控.....	38
5.8 媒体设备.....	40
5.9 音视频呼叫.....	43
6 问题定位.....	47
6.1 错误码列表.....	47
6.2 在线分析.....	59
6.3 日志分析.....	62
7 附录.....	64
7.1 更新驱动.....	64
7.2 https 访问配置.....	66
7.3 index.html 完整代码.....	67
7.4 TSDK 安装包制作.....	70
7.4.1 环境准备.....	70
7.4.2 TSDK 安装包快速制作步骤.....	71
7.4.3 TSDK 安装及卸载.....	73
7.4.4 进阶指南.....	74
7.4.4.1 打包目录路径自定义修改.....	74
7.4.4.2 开机自启动.....	76
7.4.4.3 修改证书及业务端口.....	77
8 修订记录.....	78

1 CloudVC SDK 简介

CloudVC 简介

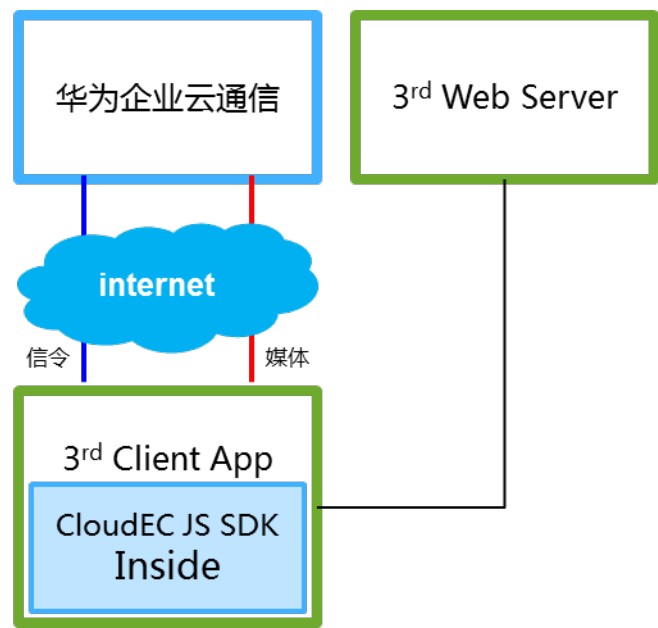
Cloud VC（Cloud Video Conference）是华为企业云视讯解决方案，它是面向政府、交通、安全、金融、大企业、中小企业等领域，提供高临场感的远程会议、桌面及移动视频接入、企业流媒体应用等场景下的视频会议整体解决方案。Cloud VC融合了语音、视频、数据等多种媒体能力，为企业提供一站式的融合的通信平台，包括高清音视频会议、Web会议、协作等丰富的业务，可适配不同企业不同的场景应用。视讯解决方案在架构上分为业务平台层、媒体处理层、用户接入层，各层产品功能紧密配合。



服务器REST: 北向通过能力开放网关提供网络侧REST接口，支持与第三方行业系统基于HTTPS进行交互。

客户端SDK: 南向开放提供Windows/Android/iOS平台的客户端SDK，第三方客户端应用通过简单的接口调用来集成相关通信能力。

JS SDK 是什么？



CloudVC的JS SDK是一个JavaScript接口的集合，是基于企业云通信整体解决方案，将音视频呼叫、音视频会议、桌面共享等能力通过简单的API调用，快速集成到您的客户端WEB应用中，无须您关注底层的音视频技术实现细节。

JS SDK 目前包含以下业务对象：

类别	对象	描述
SDK	CloudVC	SDK全局对象
客户端	Client	客户端管理对象，是JSSDK的基础对象，提供用户登录、登出，会议管理、媒体设备管理和企业通讯录等接口

CloudEC

Client

eSDK CloudEC 提供什么？

CloudVC eSDK二次开发目前提供JavaScript接口的集合，将华为eSDK CloudVC二次开发能力通过接口调用的方式对外开放。

我们提供的eSDK CloudVC包内容由以下几部分：

- SDK包和文档
CloudVC二次开发使用的SDK包和文档，提供CloudVC接口的Windows动态库、接口参考、开发指南。详情请参见[SDK和文档下载路径](#)。

- **Sample Codes**

CloudVC系列的Sample Codes，演示如何调用接口，帮助您完成企业通信相关业务开发。详情请参见[Sample Codes下载路径](#)。

2 内容导航

须知

因相应SDK版本暂仅适用于CloudVC解决方案下的融合会议组网，所以此文档中描述的功能全部仅适用于CloudVC解决方案下的融合会议组网。

本文主要描述CloudVC能力开放的常用功能，以及指导您如何调用SDK接口集成这些功能。主要内容如下：

- **相关资源**：二次开发过程中可能涉及到的软件、文档资源链接、技术支持。
- **Hello World**：如果你仅仅是尝试把SDK运行起来，您应该首先看这部分内容，它会详细说明如何下载及安装SDK以及如何配置您的开发环境。
- **业务开发指导**：介绍SDK开放的典型功能场景，包括开发流程、样例代码以及注意事项等。
- **问题定位**：介绍开发过程中常见问题的定位方法。
- **附录**：介绍开发过程中可能需要用到的内容。
- **修订记录**：各版本开发指南更新细节。

阅读建议

- 如果只是想快速入门，可仅参考**Hello World**章节。
- 如果想深入了解eSDK CloudEC核心业务的二次开发，建议您参考**业务开发指导**章节。
- 使用SDK过程中遇到问题可以参考**问题定位**章节；或联系**技术支持渠道**。

3 相关资源

- 3.1 SDK和文档下载路径
- 3.2 Sample Codes下载路径
- 3.3 免费申请远程实验室
- 3.4 技术支持渠道

3.1 SDK 和文档下载路径

华为开发者社区

- 访问[华为开发者社区资源中心](#)，获取对应解决方案和平台的SDK和文档。

华为企业产品技术支持网站

- 访问[华为企业产品技术支持网站](#)，单击“软件下载 > 企业业务公共”，显示的页面选择“eSDK解决方案 > eSDK VC”，获取SDK。
- 访问[华为企业产品技术支持网站](#)，单击“产品文档 > 企业业务公共”，显示的页面选择“eSDK VC”，获取文档。

校验软件包完整性

- 步骤1** 从“[软件数字签名\(OpenPGP\)验证工具](#)”下载软件校验需要的资源。（选择V100R001C00版本即可）
- 具体所需资源如[表3-1](#)所示。

表 3-1 资源说明

选项	说明
VerificationTools.zip	数字签名校验工具。
KEYS	华为软件数字签名公钥。
OpenPGP签名验证指南.pdf	数字签名验证指南。

步骤2 将被签名文件和对应的签名文件放在同一个目录下。

步骤3 参见《OpenPGP签名验证指南.pdf》进行软件数字签名验证

----结束

3.2 Sample Codes 下载路径

访问[github](#)网站， 下载Sample Codes。

这里建议您使用Visual Studio Code 1.8版本编译执行Sample Codes。

如果需要下载及安装SDK，请参考 [Hello World](#)。

Sample Code名称	描述
CloudEC_Client_API_Demo_Windows_JS	包含事件处理模板、接口使用示例；包含TSDK安装包制作依赖的配置文件和工具源码。您可以基于Demo快速了解到大部分功能接口的调用代码。暂仅适用于CloudEC解决方案下的融合会议组网。

3.3 免费申请远程实验室

华为 eSDK 远程实验室简介

华为eSDK 远程实验室致力于为合作伙伴提供真实的华为ICT 产品能力的远程对接联调环境，通过在线申请相应ICT 产品的测试帐号与权限，您不需要购置华为产品便可基于远程实验室针对相关产品进行二次开发，并实现远程对接测试认证。借助华为远程实验室，您可以“零”出差构建解决方案，“零”设备构建演示环境，提高对接测试通过率，缩短产品二次开发周期。

华为eSDK远程实验室为开发者提供了7×24小时的免费云化实验室环境，提供真实的华为设备供开发者远程在线开发调试。借助远程实验室自助管理平台，开发者不需要购置华为产品便可基于远程实验室针对相关产品进行二次开发，并实现远程对接测试认证。

目前，华为远程实验室已发布Cloud Computing、Carrier Software、SDN、BYOD、Cloud VC、Agile Network、eLTE、Big Data、IoT、IES等多个生态圈的实验室环境。

相关信息请查询[华为开发者社区远程实验室](#)。

远程实验室有哪些优势

- 低门槛：官网注册用户即可申请使用，环境与预约时长、预约次数受限；
- 分级支持：环境按域划分，重点开发者、合作伙伴可访问特定环境并享受额外的预约环境；
- 全球资源高速互联：建设以苏州远程实验室为中心的实验室分布格局，依托IT全球100ms高性能骨干网络和IT全球端到端应用保障能力。

如何免费使用远程实验室

请访问[远程实验室操作指导](#)，查阅远程实验室预约申请及接入使用方法。

3.4 技术支持渠道

如果您在软件开发过程中碰到任何问题，开发者社区提供了以下技术支持渠道：

- 在[DevCenter](#)中提单跟踪。
- 在[华为开发者论坛](#)中发帖求助。

4 Hello World

- [4.1 概述](#)
- [4.2 环境准备](#)
- [4.3 HelloWorld](#)
- [4.4 运行HelloWorld](#)

4.1 概述

本示例基于JS SDK，通过登录EC服务器并加入视频会议的简单案例，向您展示如何使用CloudVC JS SDK快速进行二次集成开发。

4.2 环境准备

请确保满足以下开发环境要求:

环境和工具名称	版本要求	说明
操作系统	Windows 7专业版	硬件要求： CPU: i5-2400四核 3.1GHz及以上 内存: 4GB及以上
Visual Studio Code	VS Code 1.8	或者其他IDE工具
Chrome或Firefox或IE	Chrome 62及以上版本 Firefox 57及以上版本 IE11	webGL支持硬加速（若不是，请参考 更新驱动 升级到最新驱动） Native视频窗口仅支持32位浏览器。
webcomponentsjs	1.0.20	浏览器兼容性组件， 下载链接 ，文件为webcomponentsjs-1.0.20.zip（此组件仅用于Hello World支持多浏览器的演示使用，不作商用目的）。

环境和工具名称	版本要求	说明
Web服务器	推荐： Tomcat或Apache	在WEB服务器中配置JSSDK路径，将SDK和HTML网页设置到对应的WEB容器中
准备一个CloudVC的用户账号	CloudEC 6.1及以上版本	EC账号可来源于以下三种CloudEC解决方案环境： 1 华为远程实验室 2 华为公有云 3 企业入驻

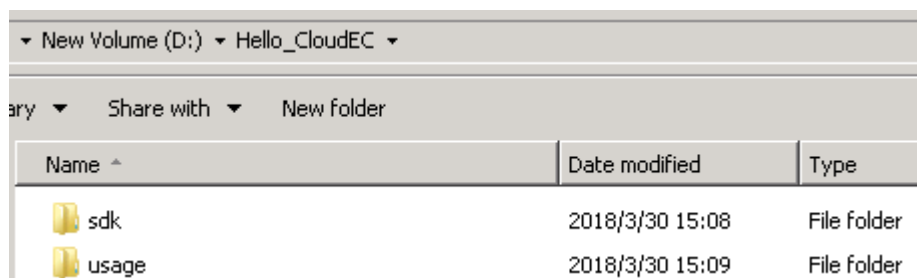
须知

Native视频的视频显示方式：视频显示最大支持720P，30fps。
数据会议不推荐使用，屏幕共享显示最大支持480P，15fps。

4.3 HelloWorld

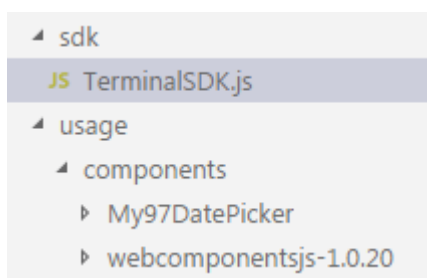
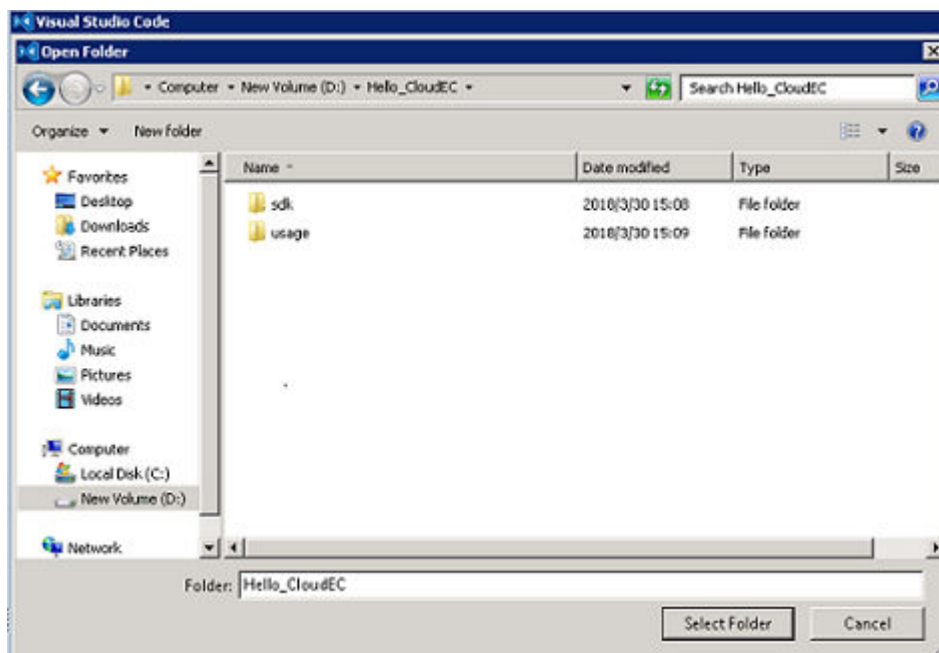
4.3.1 步骤 1 创建目录

在任意路径下创建Hello_CloudEC文件夹，解压CloudEC_Client_API_Demo_Windows_JS.zip，获取web_server_demo目录中的sdk和usage文件夹，并拷贝到刚才创建的Hello_CloudEC路径下。将下载的webcomponentsjs-1.0.20.zip解压到Hello_CloudEC的\usage\components目录中。

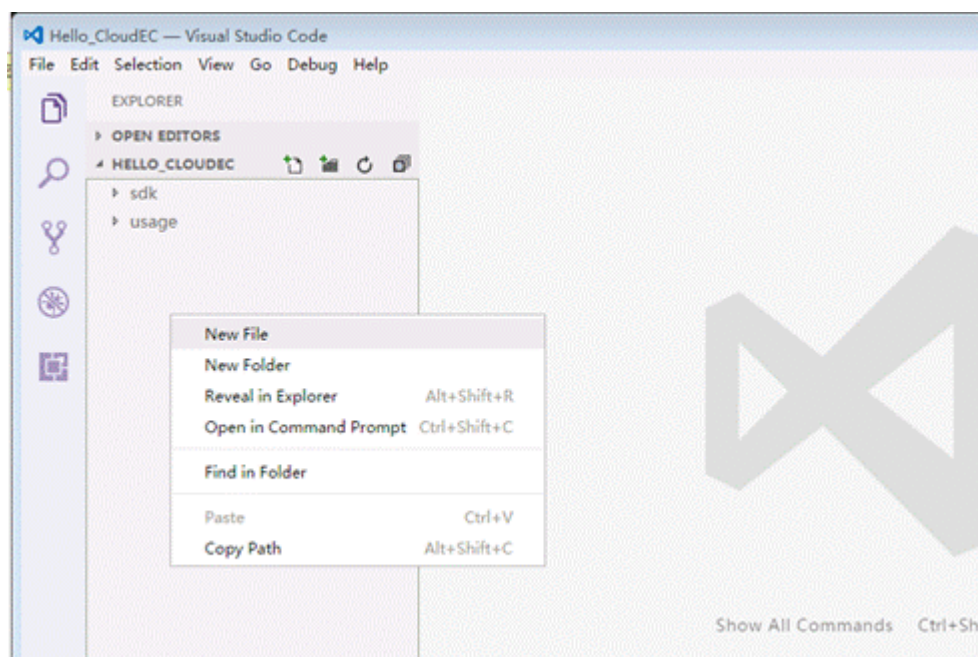


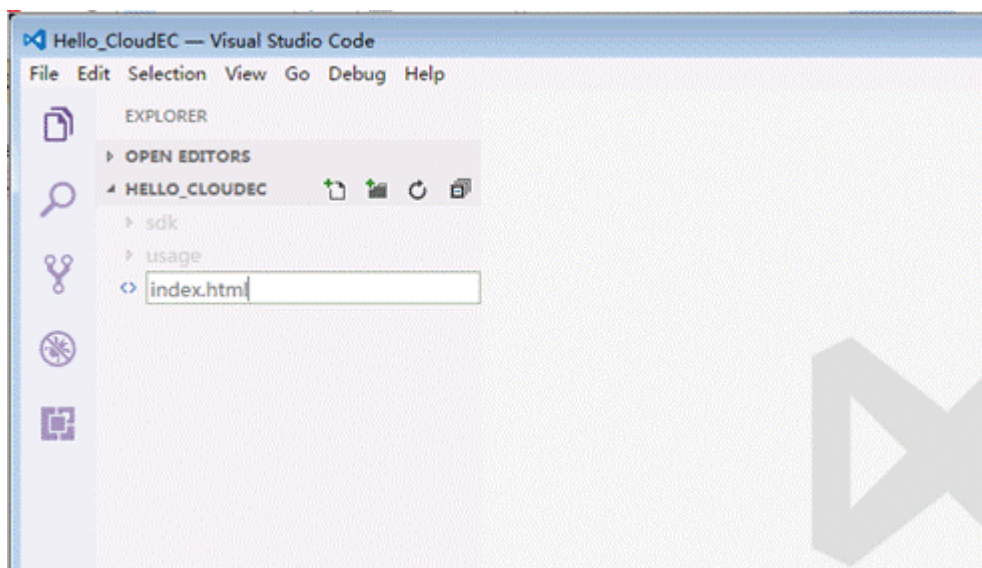
4.3.2 步骤 2 创建工程

打开Visual Studio Code，单击菜单栏的“文件 > 打开文件夹”，打开“选择文件夹窗口”窗口，进入Hello_CloudEC文件夹路径，单击“选择文件夹”，完成新工程的创建。



在 Visual Studio Code 工程导航栏右键打开菜单，选择“新建文件”选项，创建 index.html 文件。





须知

index.html必须创建在SDK文件夹同层次路径下。

4.3.3 步骤 3 引入依赖

在index.html的head标签开始处，导入JS SDK和UI控件。

```
<html>
<head>
  <meta charset="UTF-8">
  <script src="./sdk/TerminalSDK.js" /></script>
  <!--begin:webcomponentsts init-->
  <script>
    if (!window.customElements) { document.write('<!--'); } else { window.ShadyDOM = { force:
true }; window.customElements.----- = true; }
  </script>
  <script src="./usage/components/webcomponentsjs-1.0.20/custom-elements-es5-adapter.js"></
script>
  <!--! do not remove -->
  <script src="./usage/components/webcomponentsjs-1.0.20/webcomponents-loader.js"></script>
  <!--end:webcomponentsts init-->
```

4.3.4 步骤 4 创建 tsdkClient 对象

初始化系统配置，创建**listeners**对象，注册各个监听事件和对应的回调函数，将**listeners**对象作为参数，传入**createTsdkClient**，创建**tsdkClient**对象。可监听的事件请参考接口参考的事件类型小节。

```
"use strict";
(function (root) {
  var tsdkJsInitParam = {
    invokeMode: 1,
    svrAddr: "localhost.cloudec.huaweicloud.com",
    svrPort: "7684",
    ssl: 1
  };
  root.isNoLogin = false;
  var listeners = {
    OnEvtAuthSuccess: (ret) => {
```

```
        console.debug('auth success!');
        alert("auth success!" + JSON.stringify(ret))
    },
    OnEvtLoginSuccess: (ret) => {
        console.info('login success!' + JSON.stringify(ret));
        root.isNoLogin = true;
        alert("login success!")
    },
    OnError: function(ret) {
        if (3900000003 == ret.info.errorCode) {
            console.warn("Memory usage over 80%, please close the unrelated program.");
        } else {
            alert(JSON.stringify(ret));
        }
    },
};
terminalSDK.createTsdkClient(tsdkJsInitParam, listeners, (data)=>{
    root.tsdkClient = data
});
if (root.tsdkClient == null) {
    console.log("createTsdkClient");
}
})(this);
```

4.3.5 步骤 5 登录 CloudVC 服务器

创建login方法，并在其中调用tsdkClient.login()接口。

```
// 登录
function login() {
    // 设置业务配置参数
    var configParam = {
        logParam: 1,
        tlsParam: 1,
        proxyParam: 1,
        serviceSecurityParam: 1,
        iptServiceConfigParam: 1,
        localAddress: 1,
        filePathInfo: 1,
        dpiInfo: 1,
        networkInfo: 1,
        ipCallSwitch: 0,
        confCtrlParam: 1,
        sendDataSwitch: 1,
        baseInfoParam: 1,
        frameParam: 1,
        visibleInfo: 1
    }
    var callback = function() {}
    tsdkClient.setConfigParam(configParam, callback);
    // 初始化
    var tsdkAppInfoParam = {
        "clientType": 0,
        "productName": "SoftClient on Desktop",
        "deviceSn": "1",
        "supportAudioAndVideoCall": 1,
        "supportAudioAndVideoConf": 1,
        "supportDataConf": 1,
        "supportCtd": 0,
        "supportIm": 0,
        "supportRichMediaMessage": 0,
        "supportEnterpriseAddressBook": 0,
        "useUiPlugin": 1,
        "isWsInvokeMode": 1,
    };
    var callbacks = function (res) { }
    tsdkClient.init(tsdkAppInfoParam, (res)=>{
        if(res.result == 0) {
            alert("init success!")
        }
    })
}
```



```
    }  
  });  
  console.log(isNoLogin)  
  var userName = document.getElementById("name").value;  
  var password = document.getElementById("passwd").value;  
  var serverAddr = document.getElementById("svr_addr").value;  
  var serverPort = document.getElementById("svr_port").value;  
  var tsdkLoginParam = {  
    "userId": 1,  
    "authType": 0,  
    "userName": userName, //SC上分配的预定义节点  
    "password": password,  
    "userTiket": "1",  
    "serverType": 2, //SMC服务器  
    "serverVersion": "",  
    "serverAddr": serverAddr, //SMC服务器地址 192.160.54.207  
    "serverPort": parseInt(serverPort) // 5061  
  };  
  var callbacks = function (res) {  
    console.log(res)  
  }  
  console.debug('login begin');  
  tsdkClient.login(tsdkLoginParam, callbacks);  
  
  console.debug('login ret:' + callbacks);  
  passwd = "";  
  proxyPassword = "";  
  proxyParam = "";  
  
  document.getElementById("call").style.display = "block";  
}
```

4.3.6 步骤 6 加入会议

创建book_conf.html组件，在组件中调用tsdkClient.bookConference()接口，预约创建会议

创建创建joinInstanceConf函数，在其中调用tsdkClient.joinInstanceConf()接口，加入已开始会议。

```
<!-- 预定会议组件 -->  
<script>  
  function _classCallCheck(instance, Constructor) { if (!(instance instanceof Constructor))  
  { throw new TypeError("Cannot call a class as a function"); } }  
  function _possibleConstructorReturn(self, call) { if (!self) { throw new ReferenceError("this  
hasn't been initialised - super() hasn't been called"); } return call && (typeof call === "object"  
|| typeof call === "function") ? call : self; }  
  function _inherits(subClass, superClass) { if (typeof superClass !== "function" && superClass !  
== null) { throw new TypeError("Super expression must either be null or a function, not " + typeof  
superClass); } subClass.prototype = Object.create(superClass && superClass.prototype,  
{ constructor: { value: subClass, enumerable: false, writable: true, configurable: true } }); if  
(superClass) Object.setPrototypeOf ? Object.setPrototypeOf(subClass, superClass) :  
subClass.__proto__ = superClass; }  
  //Custom variables and functions  
  var roomTimeZone = [  
    ["-12", "(UTC-12:00)Marshall Islands"],  
    ["-11", "(UTC-11:00)Samoa"],  
    ["-10", "(UTC-10:00)Honolulu"],  
    ["-8", "(UTC-08:00)Anchorage"],  
    ["-7", "(UTC-07:00)Arizona"],  
    ["-7", "(UTC-07:00)San Francisco"],  
    ["-7", "(UTC-07:00)Tijuana"],  
    ["-7", "(UTC-07:00)Chihuahua"],  
    ["-6", "(UTC-06:00)Denver"],  
    ["-6", "(UTC-06:00)Saskatchewan"],  
    ["-6", "(UTC-06:00)Tegucigalpa"],  
    ["-6", "(UTC-06:00)Mexico City"],  
    ["-5", "(UTC-05:00)Bogota"],  
  ]  
}
```

```
[ "-5", "(UTC-05:00)Chicago"],
[ "-4.5", "(UTC-04:30)Caracas"],
[ "-4", "(UTC-04:00)Atlantic City"],
[ "-4", "(UTC-04:00)Indiana"],
[ "-4", "(UTC-04:00)La Paz"],
[ "-4", "(UTC-04:00)New York"],
[ "-3", "(UTC-03:00)Nuuk"],
[ "-3", "(UTC-03:00)Buenos Aires"],
[ "-3", "(UTC-03:00)Halifax"],
[ "-3", "(UTC-03:00)Recife"],
[ "-2.5", "(UTC-02:30)Newfoundland"],
[ "-2", "(UTC-02:00)Brasilia Time"],
[ "-1", "(UTC-01:00)Azores"],
[ "-1", "(UTC-01:00)Mid-Atlantic"],
[ "0", "(UTC00:00)London"],
[ "0", "(UTC+00:00)Casablanca"],
[ "0", "(UTC) Reykjavik"],
[ "0", "(UTC 00:00)Dakar"],
[ "1", "(UTC+01:00)Paris"],
[ "1", "(UTC+01:00)Madrid"],
[ "1", "(UTC+01:00)Berlin"],
[ "1", "(UTC+01:00)Amsterdam"],
[ "1", "(UTC+01:00)West Africa"],
[ "1", "(UTC+01:00)Warsaw"],
[ "1", "(UTC+01:00)Stockholm"],
[ "1", "(UTC+01:00)Rome"],
[ "2", "(UTC+02:00)Pretoria"],
[ "2", "(UTC+02:00)Helsinki, Kiev"],
[ "2", "(UTC+02:00)Athens"],
[ "2", "(UTC+02:00)Amman"],
[ "2", "(UTC+02:00)Cairo"],
[ "2", "(UTC+02:00)Tel Aviv"],
[ "3", "(UTC+03:00)Riyadh"],
[ "3", "(UTC+03:00)Nairobi"],
[ "3", "(UTC+03:00)Minsk"],
[ "3", "(UTC+03:00)Istanbul"],
[ "3", "(UTC+03:00)Moscow"],
[ "3.5", "(UTC+03:30)Tehran"],
[ "4", "(UTC+04:00)Baku"],
[ "4", "(UTC+04:00)Abu Dhabi, Muscat"],
[ "4.5", "(UTC+04:30)Kabul"],
[ "5", "(UTC+05:00)Islamabad"],
[ "5", "(UTC+05:00)Ekaterinburg"],
[ "5.5", "(UTC+05:30)Colombo"],
[ "5.5", "(UTC+05:30)Mumbai"],
[ "6", "(UTC+06:00)Yekaterinburg"],
[ "6", "(UTC+06:00)Almaty"],
[ "6.3", "(UTC+06:30)Yangon"],
[ "7", "(UTC+07:00)Bangkok"],
[ "7", "(UTC+07:00)Novosibirsk"],
[ "8", "(UTC+08:00)Taipei"],
[ "8", "(UTC+08:00)Singapore"],
[ "8", "(UTC+08:00)Perth"],
[ "8", "(UTC+08:00)Kuala Lumpur"],
[ "8", "(UTC+08:00)Beijing"],
[ "9", "(UTC+09:00)Tokyo"],
[ "9", "(UTC+09:00)Seoul"],
[ "9.5", "(UTC+09:30)Darwin"],
[ "10", "(UTC+10:00)Yakutsk"],
[ "10", "(UTC+10:00)Guam"],
[ "10", "(UTC+10:00)Brisbane"],
[ "10.5", "(UTC+10:30)Adelaide"],
[ "11", "(UTC+11:00)Vladivostok"],
[ "11", "(UTC+11:00)Sydney"],
[ "11", "(UTC+11:00)Solomon Is"],
[ "11", "(UTC+11:00)Hobart"],
[ "13", "(UTC+13:00)Wellington"],
[ "13", "(UTC+13:00)Fiji"]];
var confStartTime = function (evt) {
```

```
var evt = evt || window.event;
var e = evt.srcElement || evt.target;
if (e.value == "0") {
    document.getElementById('cloudec_input_datetime').style.display = 'none';
    document.getElementById('cloudec_room_timezone').style.display = 'none';
    document.getElementById('cloudec_td_chairmain').style.display = 'none';
    document.getElementById('cloudec_input_chairmain').style.display = 'none';
} else {
    document.getElementById('cloudec_input_datetime').style.display = 'block';
    document.getElementById('cloudec_room_timezone').style.display = 'block';
    document.getElementById('cloudec_td_chairmain').style.display = 'block';
    document.getElementById('cloudec_input_chairmain').style.display = 'block';
}
}
//Custom element
var CloudEC_BookConf = function (_HTMLElement) {
    _inherits(CloudEC_BookConf, _HTMLElement);

    function CloudEC_BookConf() {
        _classCallCheck(this, CloudEC_BookConf);
        return _possibleConstructorReturn(this, (CloudEC_BookConf.__proto__ ||
Object.getPrototypeOf(CloudEC_BookConf)).call(this));
    }
    CloudEC_BookConf.prototype.connectedCallback = function () {
        console.log("it's alive! connectedCallback");
        this.callback()
    }
    CloudEC_BookConf.prototype.createdCallback = function () {
        console.log("it's alive! createdCallback");
        this.callback()
    }
    CloudEC_BookConf.prototype.callback = function () {
        console.log("callback")
        var select_str = ""
        for (var m in roomTimeZone) {
            if (roomTimeZone[m][1] == "(UTC+08:00)Beijing") {
                select_str += "<option selected='selected' value='" + roomTimeZone[m][0] +
"">" + roomTimeZone[m][1] + "</option>"
            }
            else {
                select_str += "<option value='" + roomTimeZone[m][0] + "">" + roomTimeZone[m]
[1] + "</option>"
            }
        }
        var cloudec_confbook_div = document.createElement('div')
        cloudec_confbook_div.id = "cloudec_confbook_div"
        cloudec_confbook_div.innerHTML = "<table>"
            + "<tr>"
            + "<td>conference type:</td><td>"
            + "<input id='cloudec_conf_type' name='cloudec_conf_type' type='radio' value='0' /
>audio"
            + "<input id='cloudec_conf_type' name='cloudec_conf_type' type='radio' value='1' /
>video"
            + "<input id='cloudec_conf_type' name='cloudec_conf_type' type='radio' value='2' /
>audio+data"
            + "<input id='cloudec_conf_type' name='cloudec_conf_type' checked='checked'
type='radio' value='3' />video+data"
            + "</td></tr>"
            + "<tr>"
            + "<td/>starting time:</td><td>"
            + "<input id='cloudec_start_type' name='cloudec_start_type'
onclick='confStartType()' type='radio' value='0' />now"
            + "<input id='cloudec_start_type' name='cloudec_start_type'
onclick='confStartType()' checked='checked' type='radio' value='1' />later"
            + "</td></tr>"
            + "<tr>"
            + "<td/>HD conference:</td><td>"
            + "<input id='cloudec_is_HD_conf' name='cloudec_is_HD_conf' checked='checked'
type='radio' value='1' />yes"
```

```
+ "<input id='cloudec_is_HD_conf' name='cloudec_is_HD_conf' type='radio'
value='0' />no"
+ "</td>"
+ "</tr>"
+ "<tr><td></td><td>"
+ "<select id='cloudec_room_timezone' name='cloudec_room_timezone' style='width:
200px' >"
+ select_str
+ "</select>"
+ "</td></tr>"
+ "<tr>"
+ "<td></td>"
+ "<td>"
+ "<input type='text' id='cloudec_input_datetime' style='width:200px'
onclick='WdatePicker({dateFmt:'\\yyyy-MM-dd HH:mm:ss\\',lang:'\\en\\'})' />"
+ "</td>"
+ "</tr>"
+ "<tr>"
+ "<td>duration (minutes):</td>"
+ "<td><input id='cloudec_input_duration' type='text' value = '30' style='width:
200px' /></td>"
+ "</tr>"
+ "<tr>"
+ "<td>subject:</td>"
+ "<td><input id='cloudec_input_topic' type='text' value = 'CloudVC-Meeting'
style='width:200px' /></td>"
+ "</tr>"
+ "<tr>"
+ "<td id='cloudec_td_chairmain'>chairman:</td>"
+ "<td><input id='cloudec_input_chairmain' type='text' value='' style='width:
200px' /></td>"
+ "</tr>"
+ "<tr>"
+ "<td>attendee:</td>"
+ "<td><input id='cloudec_input_attendees' type='text' value='' style='width:
200px' /></td>"
+ "</tr>"
+ "</tr>"
+ "</table>"
var cloudec_confbook_btn = document.createElement('button')
cloudec_confbook_btn.innerHTML = "book conference"
cloudec_confbook_btn.addEventListener("mousedown", function (e) {
var startType =
parseInt(document.querySelector('input[id="cloudec_start_type"]:checked').value)
var isHDConf =
parseInt(document.querySelector('input[id="cloudec_is_HD_conf"]:checked').value)
var confMediaType =
parseInt(document.querySelector('input[id="cloudec_conf_type"]:checked').value)
var dateTime = document.getElementById("cloudec_input_datetime").value;
var topic = document.getElementById("cloudec_input_topic").value;
var duration = document.getElementById("cloudec_input_duration").value;
var chairmain = document.getElementById("cloudec_input_chairmain").value
var attendees_str = document.getElementById("cloudec_input_attendees").value
var attendees_array = attendees_str.split(",");
//The chairman is default self in instant conference. No need input.
if (startType == 1){
attendees_array.push(chairmain)
}
var attendeeList = [];
for (var m in attendees_array) {
var attendeeParam = {}
attendeeParam.name = attendees_array[m]
attendeeParam.number = attendees_array[m]
attendeeParam.role = 0
attendeeParam.smsPhone = ""
attendeeParam.email = ""
attendeeList[m] = attendeeParam
}
if (startType == 1){
```

```
        attendeeList[attendeeList.length - 1].role = 1;
    }
    //Get time and convert to UTC for the corresponding time zone
    var timezone = document.getElementById("cloudec_room_timezone");
    var offset = timezone.options[timezone.selectedIndex].value * 60 * 1000;

    //According to the selected time zone converted to the corresponding UTC time
    var utc_millisecond = Date.parse(dateTime)
    utc_millisecond = utc_millisecond - offset
    var newDate = new Date(utc_millisecond)
    var bookConfInfo = {
        welcomePrompt: 2,
        isMultiStreamConf: 0,
        language: 1,
        isHdConf: 1,
        isAutoMute: 1,
        attendeeNum: 2,
        confType: startType,
        recordMode: 0,
        isAutoRecord: 0,
        enterPrompt: 1,
        groupUri: "",
        attendeeList: attendeeList,
        startTime: startType == 0 ? null : dateTime,
        confEncryptMode: 0,
        duration: parseInt(duration),
        confMediaType: confMediaType,
        reminder: 0,
        size: 2,
        leavePrompt: 0,
        isAutoProlong: 0,
        subject: topic,
    }
    //Interface to call a book meeting
    tsdkClient.bookConference(bookConfInfo, function (ret) {
        console.log(bookConfInfo)
        alert("bookConference callback" + JSON.stringify(ret))
    })

    })
    this.appendChild(cloudec_confbook_div)
    this.appendChild(cloudec_confbook_btn)
    //var jsurl = './usage/components/My97DatePicker/WdatePicker.js'
    //this.loadJs(jsurl, null)
}

CloudEC_BookConf.prototype.loadJs = function (jsurl, callback) {
    var nodeHead = document.getElementsByTagName('head')[0];
    var nodeScript = null;
    nodeScript = document.createElement('script');
    nodeScript.setAttribute('charset', 'UTF-8');
    nodeScript.setAttribute('type', 'text/javascript');
    nodeScript.setAttribute('src', jsurl);
    if (callback != null) {
        nodeScript.onload = nodeScript.onreadystatechange = function () {
            if (nodeScript.readyState == 'complete' || nodeScript.readyState == 'loaded' ||
nodeScript.readyState == 'compvare') {
                nodeScript.readyState = 'complete';
                callback();
            }
        };
    }
    nodeHead.appendChild(nodeScript);
}
return CloudEC_BookConf;
}(HTMLDocument);
```

```
customElements.define("cloudec-bookconf", CloudEC_BookConf);
</script>

function joinInstanceConf() {
    var joinNumber = 0;
    var confTypeObj = document.getElementById("instance_conf_type");
    var isVideoJoin = parseInt(confTypeObj.options[confTypeObj.selectedIndex].value);
    var confId = document.getElementById("conferenceId").value;
    var accessNumber = document.getElementById("accessNumber").value;
    var confPasswd = document.getElementById("confPasswd").value;
    var TsdkConfJoinParam = {
        confId: confId,
        accessNumber: accessNumber,
        confPassword: confPasswd
    }

    tsdkClient.joinConference(joinNumber, isVideoJoin, TsdkConfJoinParam, function callback(ret) {
        console.info("join conference callback")
    });
    confPasswd = "";
    joinConfParam = "";
}
```

4.3.7 步骤 7 登出 CloudVC 服务器

创建logout方法，并在其中调用tsdkClient.logout()接口。

```
function logout() {
    var callbacks = function () { };
    tsdkClient.logout(callbacks);
    document.getElementById("login").style.display = "block";
    document.getElementById("call").style.display = "none";
}
```

4.3.8 步骤 8 创建页面主体

在body标签中，调用前述步骤中定义的登录、立即入会和预约入会方法，并且通过native视频插件显示会议视频。

```
<body>
    <h2>CloudVC JS SDK Hello World</h2>
    <div id="tab-content1" class="tab-content">
        <fieldset>
            <legend>User login</legend>
            <div id="login">
                server address: <input type="text" id="svr_addr" value="192.160.54.207" />
                port: <input type="text" id="svr_port" value="5061" />
                username: <input type="text" id="name" value="" />
                password: <input type="password" id="passwd" value="" />
                <button onclick="login()">login</button>
            </div>
        </fieldset>
    </div>
    <div id="usage">
        <fieldset>
            <legend>User logout</legend>
            <div id="userinfo"></div>
            <button onclick="logout()">logout</button>
        </fieldset>
        <fieldset>
            <legend>Instance conference</legend>
            conference type:<select id='instance_conf_type'>
                <option value=0>No video access</option>
                <option value=1>Video access</option>
            </select>
            conference ID:<input type="text" id="conferenceId"/>
            access code:<input type="text" id="accessNumber"/>
            conference password: <input type="password" id="confPasswd" value="" />
        </fieldset>
    </div>
</body>
```

```
<button onclick="joinInstanceConf()">instance conference</button>
</fieldset>
<fieldset>
  <legend>Book conference</legend>
  <cloudec-bookconf/>
</fieldset>
</div>
</body>
```

须知

Hello World的页面完整代码请参考[7.3 index.html完整代码](#)。

4.4 运行 HelloWorld

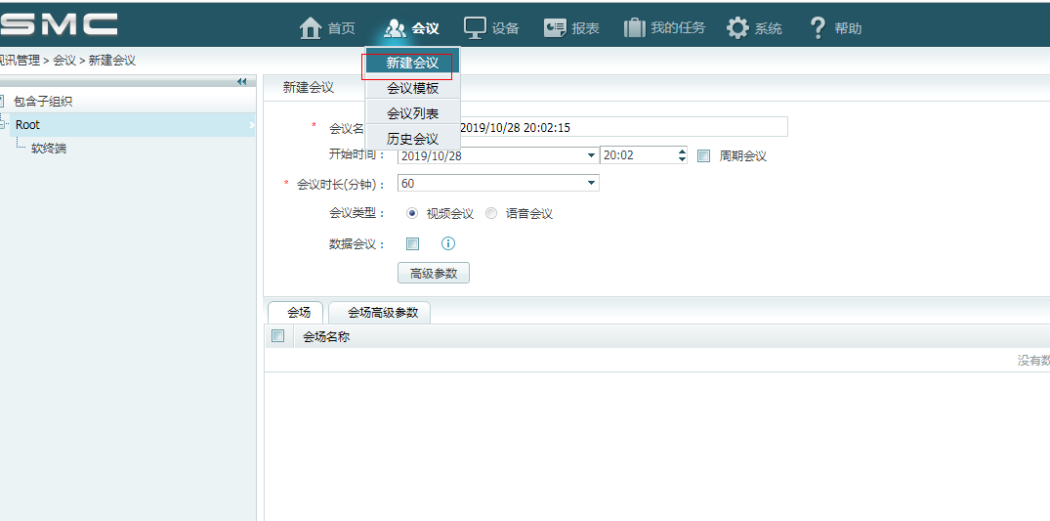
4.4.1 步骤 1 预约会议

如要测试Hello World的加入会议功能，可先登录到个人SMC账号创建一个会议，获取会议接入码登会议信息。

如要测试立即会议功能，可跳过此步骤。

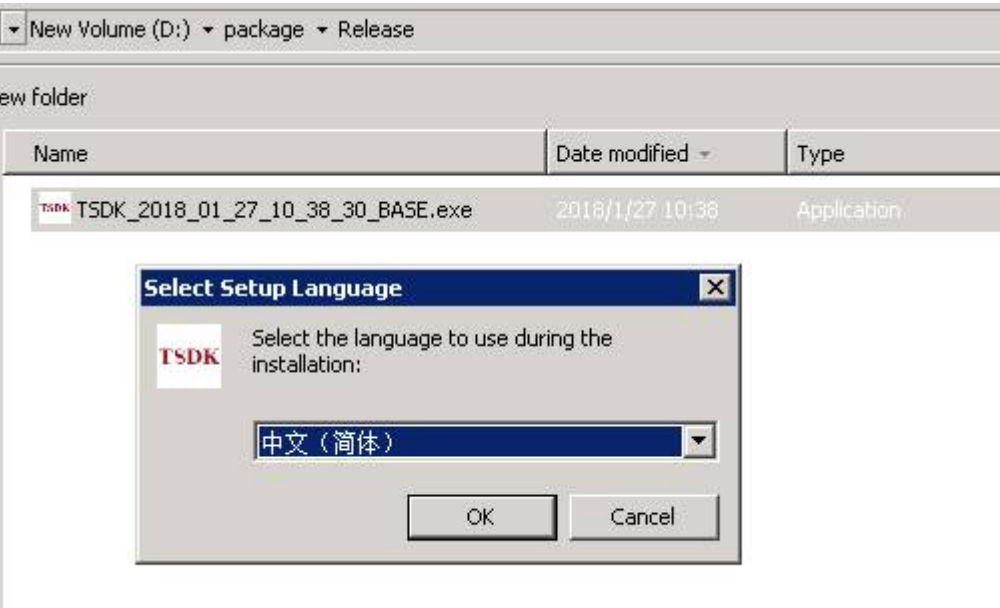


图 4-1



4.4.2 步骤 2 启动 TSDK 守护进程

安装并启动TSDK守护进程，双击TSDK_XXX_BASE.exe进入安装向导，根据引导完成安装。



说明

TSDK安装程序的制作请参考[TSDK安装包制作](#)。

4.4.3 步骤 3 WEB 服务器配置

在WEB服务器中配置Hello World路径，并重启WEB服务。以Tomcat 8.5.24为例，首先请确认Java运行时环境配置正确，tomcat可以正常启动。在tomcat的conf/server.xml中Host标签下新增Context配置，将路径设置为Hello_CloudEC目录所在位置，重启tomcat。


```
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
<Host name="localhost" appBase="webapps"
  unpackWARs="true" autoDeploy="true">

  <!-- <Context path="/" docBase="E:/EC_demo/test/ec_js_demo/web_sdk/Demo/web_server_demo/" debug="0"/> -->
  <Context path="/" docBase="E:/EC_demo/source/web_sdk/web_server_demo/" debug="0"/>

  <!-- SingleSignOn valve, share authentication between web applications
       Documentation at: /docs/config/valve.html -->
  <!--
  <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
  -->

  <!-- Access log processes all example.
       Documentation at: /docs/config/valve.html
       Note: The pattern used is equivalent to using pattern="common" -->
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
        prefix="localhost_access_log" suffix=".txt"
        pattern="%h %l %u %t &quot;%r&quot; %s %b" />

</Host>
```

4.4.4 步骤 4 访问 Hello_CloudEC 页面

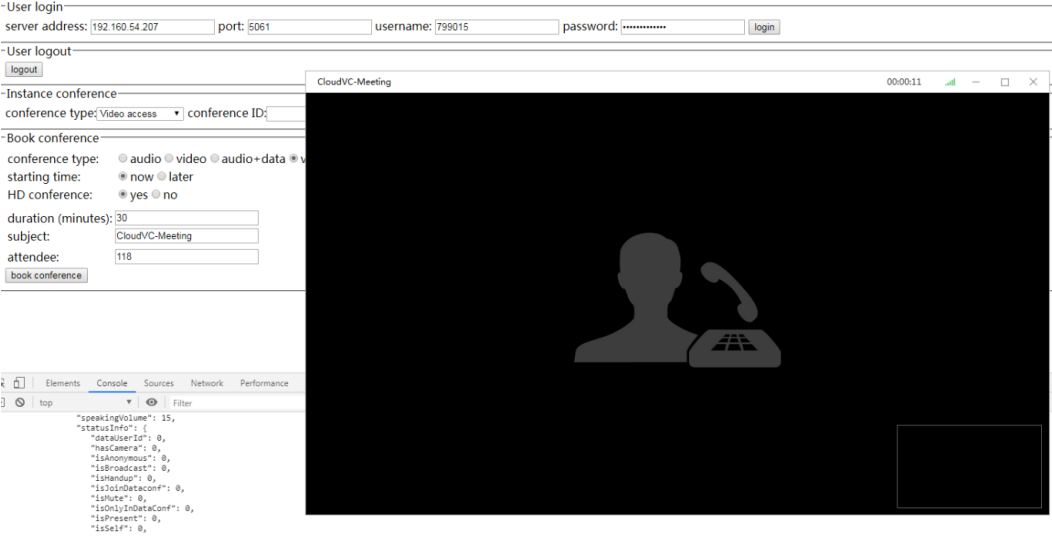
通过浏览器打开http://localhost:8080/, 注意tsdk后台进程同时仅支持一个浏览器连接, 不要同时打开多个浏览器窗口。

如需使用https访问, 请参考[https访问配置](#)。

4.4.5 步骤 5 加入会议

成功登录后, 输入入会信息, 加入会议; 或点击立即入会, 创建并加入会议。效果如图

图 4-2



5 业务开发指导

5.1 业务开发总体流程

5.2 初始化

5.3 登录鉴权

5.4 创建会议

5.5 加入会议

5.6 基本会控

5.7 视频会控

5.8 媒体设备

5.9 音视频呼叫

5.1 业务开发总体流程

在使用eSDK CloudVC提供的各类组件业务接口集成开发时，基本的流程步骤包含：

步骤1 初始化业务组件；

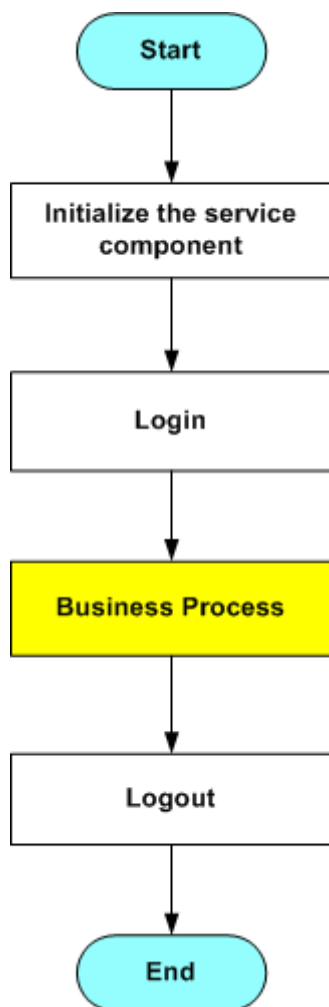
步骤2 登录；

步骤3 业务实现调用；

步骤4 登出；

----结束

图 5-1 总体流程图



流程说明

- 步骤1 初始化组件：**实现对业务组件进行资源初始化，设置第三方应用程序的全局业务配置参数。
- 步骤2 登录：**实现用户登录。
- 步骤3 业务实现调用：**根据实际业务开发需要，调用相应的业务接口实现相关业务，具体可参考后继典型业务场景描述的相关章节。
- 步骤4 登出：**实现用户退出登录，确保业务接口使用的安全性。

----结束

5.2 初始化

应用场景

在使用eSDK CloudEC系列业务组件，配套CloudEC解决方案实现各类业务前，需要先完成SDK的基础组件初始化。

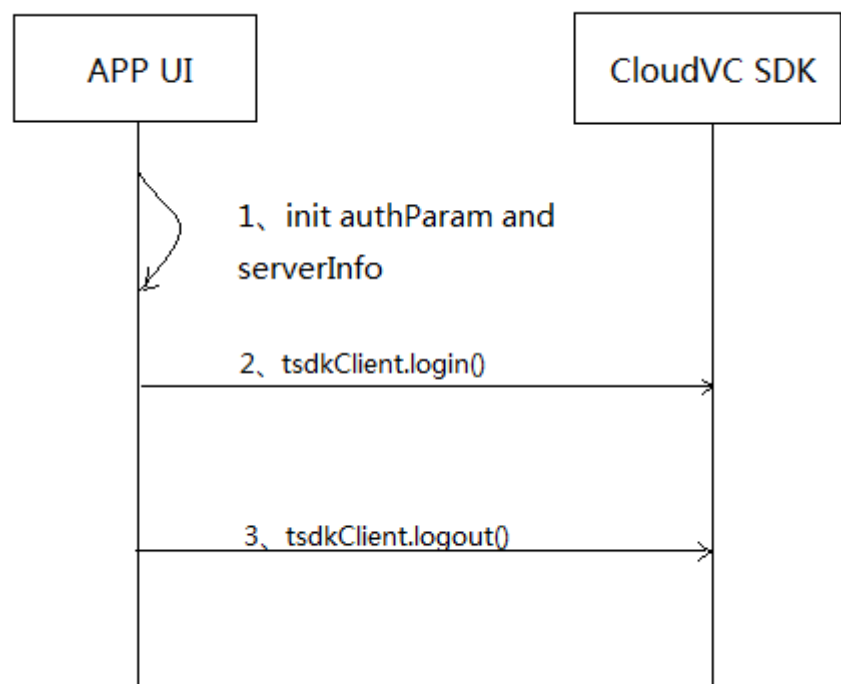
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

无。

流程说明

图 5-2 初始化配置基础组件流程图



步骤1 构建数据结构configParam

步骤2 调用tsdkClient.setConfigParam()接口，完成配置初始化，第1步中的configParam 作为参数。

代码示例：

```
// 设置业务配置参数
var configParam = {
  logParam: number,
  tlsParam: number,
  proxyParam: number,
  serviceSecurityParam: number,
  iptServiceConfigParam: number
  localAddress: number,
  filePathInfo: number,
  dpiInfo: number,
```

```
networkInfo: number,  
ipCallSwitch: number,  
confCtrlParam: number,  
sendDataSwitch: number,  
baseInfoParam: number,  
frameParam: number,  
visibleInfo: number  
}  
var callback = function() {}  
tsdkClient.setConfigParam(configParam, callback);
```

参考文件:

\usage\conference_usage.js

步骤3 构建数据结构listeners，为关心的事件添加回调函数

步骤4 调用terminalSDK.createTsdkClient()接口，完成tsdkClient对象创建，第3步中的listeners作为参数。

代码示例:

```
"use strict";  
(function (root) {  
    var tsdkJsInitParam = {  
        invokeMode: 1,  
        svrAddr: "localhost.cloudec.huaweicloud.com",  
        svrPort: "7684",  
        ssl: 1  
    };  
    root.isNoLogin = false;  
    var listeners = {  
        OnEvtAuthSuccess: (ret) => {  
            console.debug('auth success!');  
            alert("auth success!" + JSON.stringify(ret))  
        },  
        OnEvtLoginSuccess: (ret) => {  
            console.info('login success!' + JSON.stringify(ret));  
            root.isNoLogin = true;  
            alert("login success!")  
        },  
        OnEvtAuthFailed: (ret) => {  
            root.isNoLogin = false;  
            alert("OnEvtAuthFailed!")  
        },  
        OnEvtLogoutSuccess: (ret) => {  
            alert("logout Success!")  
        },  
        OnEvtLogoutFailed: (ret) => {  
            alert('logout failed!')  
        },  
        OnEvtCallOutgoing: (ret) => {  
            root.callOutGoingId = ret.param.callId  
            alert("Call out success!")  
        },  
        OnEvtCallIncoming: (ret) => {  
            root.callInComingId = ret.param.callId  
            alert("Please answer" + JSON.stringify(ret))  
        },  
        OnEvtOpenVideoReq: (ret) => {  
            alert("Audio to video")  
        },  
        OnEvtCloseVideoInd: (ret) => {  
            alert("video to audio")  
        },  
        OnEvtOpenVideoInd: (ret) => {  
            alert("video to audio")  
        },  
        OnEvtRefuseOpenVideoInd: (ret) => {
```

```
        alert("rejected!!!")
    },
    OnEvtCallEnded: (ret) => {
        alert("call end success!");
        root.callEndId = ret.param.callId
        console.log(JSON.stringify(ret))
    },
    OnEvtBookConfResult: (ret) => {
        console.log("book join successs" + JSON.stringify(ret))
    },
    OnEvtQueryConfListResult: (ret) => {
        root.confList = ret
    },
    OnEvtJoinConfResult: (ret) => {
        console.log("join sunccess" + JSON.stringify(ret))
        root.handle = ret.param.handle
    },
    OnEvtQueryConfListResult: (ret) => {
        root.callbackDatas = ret
    },
    OnError: function(ret) {
        if (390000003 == ret.info.errorCode) {
            console.warn("Memory usage over 80%, please close the unrelated program.");
        } else {
            alert(JSON.stringify(ret));
        }
    },
    });
terminalSDK.createTsdkClient(tsdkJsInitParam, listeners, (data)=>{
    root.tsdkClient = data
});
if (root.tsdkClient == null) {
    console.log("createTsdkClient");
}
})(this);
```

参考文件：

\\usage\\event_process.js

----结束

注意事项

- 1. 第1、2步可选，如不进行配置，则采用默认配置。
- 2. 第3步中，必须对OnError事件添加回调函数，否则可能无法收到必要的错误通知。

5.3 登录鉴权

应用场景

在使用CloudVC解决方案下的各类业务之前，需要向服务器完成登录鉴权。使用完各业务之后，需要向服务器完成登出注销。

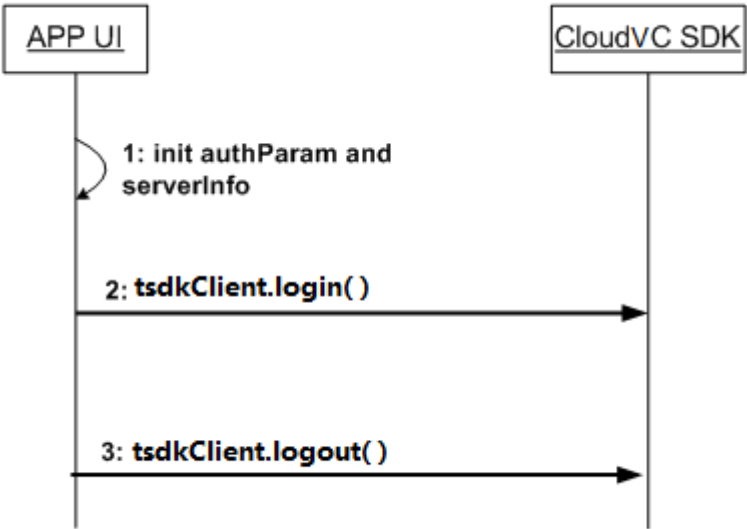
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

已初始化。

流程说明

图 5-3 登录登出流程图



- 步骤1** 构建数据结构tsdkLoginParam。
- 步骤2** 调用tsdkClient.login()接口进行登录，第1步中的tsdkLoginParam作为参数。

代码示例：

```
function login() {
    // 初始化
    var tsdkAppInfoParam = {
        "clientType": 0,
        "productName": "SoftClient on Desktop",
        "deviceSn": "1",
        "supportAudioAndVideoCall": 1,
        "supportAudioAndVideoConf": 1,
        "supportDataConf": 1,
        "supportCtd": 0,
        "supportIm": 0,
        "supportRichMediaMessage": 0,
        "supportEnterpriseAddressBook": 0,
        "useUiPlugin": 1,
        "isWsInvokeMode": 1,
    };
    var callbacks = function (res) { }
    tsdkClient.init(tsdkAppInfoParam, (res)=>{
        if(res.result == 0) {
            alert("init success!")
        }
    });
    console.log(isNoLogin)
    var userName = document.getElementById("name").value;
    var password = document.getElementById("passwd").value;
```

```
var serverAddr = document.getElementById("svr_addr").value;
var serverPort = document.getElementById("svr_port").value;
var tsdkLoginParam = {
    "userId": 1,
    "authType": 0,
    "userName": userName, //SC上分配的预定义节点 799015
    "password": password, // hw1334346871!
    "userTiket": "1",
    "serverType": 2, //SMC服务器
    "serverVersion": "",
    "serverAddr": serverAddr, //SMC服务器地址 192.160.54.207
    "serverPort": parseInt(serverPort) // 5061
};
var callbacks = function (res) {
    console.log(res)
}
console.debug('login begin');
tsdkClient.login(tsdkLoginParam, callbacks);

console.debug('login ret:' + callbacks);
passwd = "";
proxyPassword = "";
proxyParam = "";

document.getElementById("call").style.display = "block";
}
```

参考文件:

\\usage\\conference_usage.js

步骤3 调用tsdkClient.logout()接口，完成登出操作，此接口无需传入参数。

代码示例:

```
function logout() {
    var callbacks = function () {}
    tsdkClient.logout(callbacks);
    document.getElementById("login").style.display = "block";
    document.getElementById("call").style.display = "none";
}
```

参考文件:

\\usage\\conference_usage.js

----结束

注意事项

1. 调用login()接口前需要先初始化。
2. 调用logout()接口的前提是已登录。
3. 企业如果有自己内网，需要开启代理，否则有可能会sdp呼叫失败。

5.4 创建会议

应用场景

用户创建会议，查询与用户相关的会议列表，查询指定会议ID的会议详情。

功能接口	融合会议 Hosted	融合会议 On-premise	备注
bookConference()	Y	N	
getMyConfList()	Y	Y	
getMyConfInfo()	Y	Y	

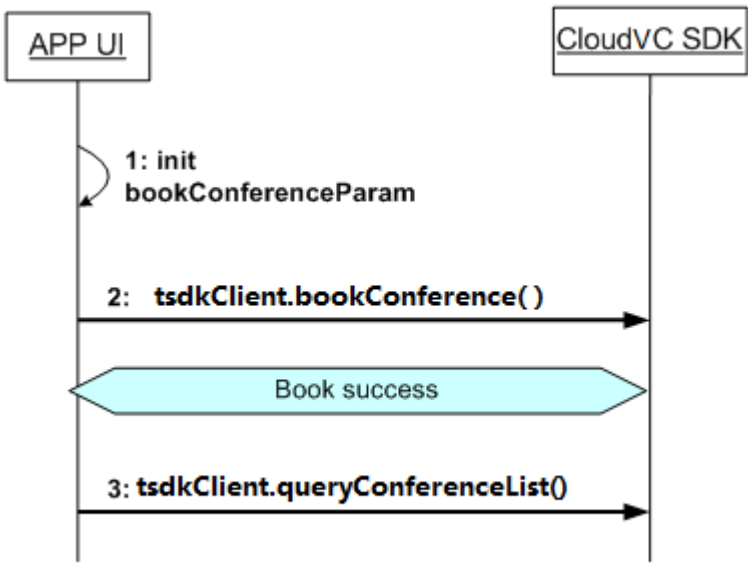
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

已登录。

流程说明

图 5-4 创建会议流程图



步骤1 构建预约会议参数的数据结构bookConfInfo 。

步骤2 调用bookConference()接口进行预约会议，第1步中的bookConfInfo 作为参数。

代码示例：

```
var bookConfInfo = {
    welcomePrompt: number,
    isMultiStreamConf: number,
    language: number,
    isHdConf: number,
    isAutoMute: number,
    attendeeNum: number,
    confType: number,
    recordMode: number,
    isAutoRecord: number,
    enterPrompt: number,
    groupUri: string,
    attendeeList: array,
    startTime: startType == 0 ? null : dateTime,
    confEncryptMode: number,
    duration: number,
    confMediaType: number,
    reminder: number,
    size: number,
    leavePrompt: number,
    isAutoProlong: number,
    subject: string,
}
//Interface to call a book meeting
tsdkClient.bookConference(bookConfInfo, function (ret) {
    console.log(bookConfInfo)
    alert("bookConference callback" + JSON.stringify(ret))
})
```

参考文件：

\usage\components\book_conf.html

步骤3 调用queryConferenceList()接口获取会议列表。

代码示例：

```
var queryReq = {
    confRight: 0,
    isIncludeEnd: 1,
    pageIndex: number
}
tsdkClient.queryConferenceList(queryReq, function (ret) {})
```

参考文件：

\usage\components\conf_list.html

----结束

注意事项

1. bookConference接口要求预约时间至少大于当前时间15分钟。

5.5 加入会议

应用场景

用户加入会议，加入匿名会议，拒绝会议邀请。

功能接口	融合会议 Hosted	融合会议 On-premise	备注
joinConference() ()	Y	Y	
joinAnonymousConf()	Y	N	

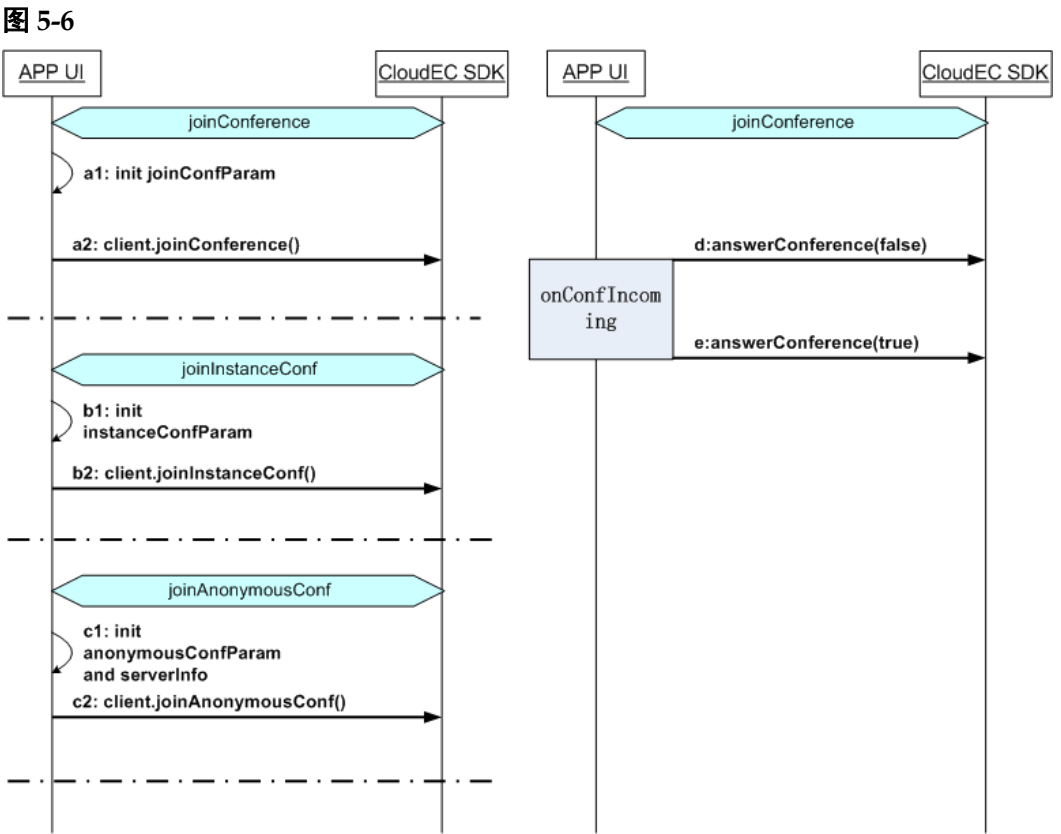
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

已登录。

流程说明

图 5-5 创建会议流程图



步骤1 加入会议流程

a1. 构建数据结构TsdkConfJoinParam

a2. 调用tsdkClient.joinConference()接口加入会议，a1中的TsdkConfJoinParam 作为参数。

代码示例：

```
function joinInstanceConf() {
    var joinNumber = 0;
    var confTypeObj = document.getElementById("instance_conf_type");
    var isVideoJoin = parseInt(confTypeObj.options[confTypeObj.selectedIndex].value);
    var confId = document.getElementById("conferenceId").value;
    var accessNumber = document.getElementById("accessNumber").value;
    var confPassword = document.getElementById("confPasswd").value;
    var TsdkConfJoinParam = {
        confId: confId,
        accessNumber: accessNumber,
        confPassword: confPassword
    }

    tsdkClient.joinConference(joinNumber, isVideoJoin, TsdkConfJoinParam, function callback(ret) {
        console.info("join conference callback")
    });
    confPasswd = "";
    joinConfParam = "";
}
```

参考文件：

\\usage\\components\\conference_usage.js

步骤2 匿名加入会议

b1. 构建数据结构anonymousConfParam

b2. 调用tsdkClient.joinConferenceByAnonymous()接口匿名加入会议，b1中的anonymousConfParam 作为参数。

代码示例：

```
function joinAnonymousConf() {
    var userName = document.getElementById("user_name").value;
    var confRandom = document.getElementById("random").value;
    var serverAddress = document.getElementById("site_url").value;
    var confPassword = document.getElementById("anony_passwd").value;
    var confId = document.getElementById("anony_conf_id").value;
    var userId = document.getElementById("user_ID").value;
    var serverPort = document.getElementById("anony_svr_port").value;
    var anonymousConfParam = {
        authType: 1,
        random: confRandom || 0,
        confPassword: confPassword,
        confId: confId,
        displayName: userName,
        serverAddr: serverAddress,
        userId: parseInt(userId),
        serverPort: parseInt(serverPort),
    }
    tsdkClient.joinConferenceByAnonymous(anonymousConfParam, function callback(ret) {
        alert("joinConferenceByAnonymous call back" + JSON.stringify(ret))
    });
    confPassword = "";
    anonymousConfParam = "";
}
```

参考文件：

\usage\components\conference_usage.js

步骤3 拒绝会议来电

```
function rejectConference() {
    var callId = callInComingId
    var callbacks = function () {}
    tsdkClient.endCall(callId, callbacks);
    call_Id = 0;
}
```

参考文件：

\usage\components\conference_usage.js

----结束

注意事项

- 1. 匿名入会无需登录，匿名入会不能管理会议，仅能作为成员进行静音、举手操作。

5.6 基本会控

须知

相对于CloudVC 19.1.0版本，当前版本及后继版本SDK支持更高效实时的会议控制能力（iDo）：

- 1. 集成时，可通过初始化系统设置接口设置会控协议confCtrlProtocol参数，设置为1时启用此能力（iDo）；
- 2. 为保持对历史版本的兼容，此能力默认关闭，即使用老版本会议控制协议rest。
- 3. 相对老版本会议控制协议rest，新版本会议控制协议iDo的会议控制能力存在部分差异，具体参见”应用场景“相关章节描述。

应用场景

增加/删除与会者成员，静音/取消静音指定与会者，申请/释放主席权限，举手，离开会议，结束会议。

功能接口	融合会议 Hosted	融合会议 On-premise	备注
addAttendee()	Y	Y	
removeAttendee()	Y	Y	
muteAttendee()	Y	Y	
releaseChairman()	Y	Y	

功能接口	融合会议 Hosted	融合会议 On-premise	备注
requestChairman()	Y	Y	
setHandup()	Y	N	Hosted 使用iDo会控协议时暂不支持举手
leaveConference()	Y	Y	
endConference()	Y	Y	

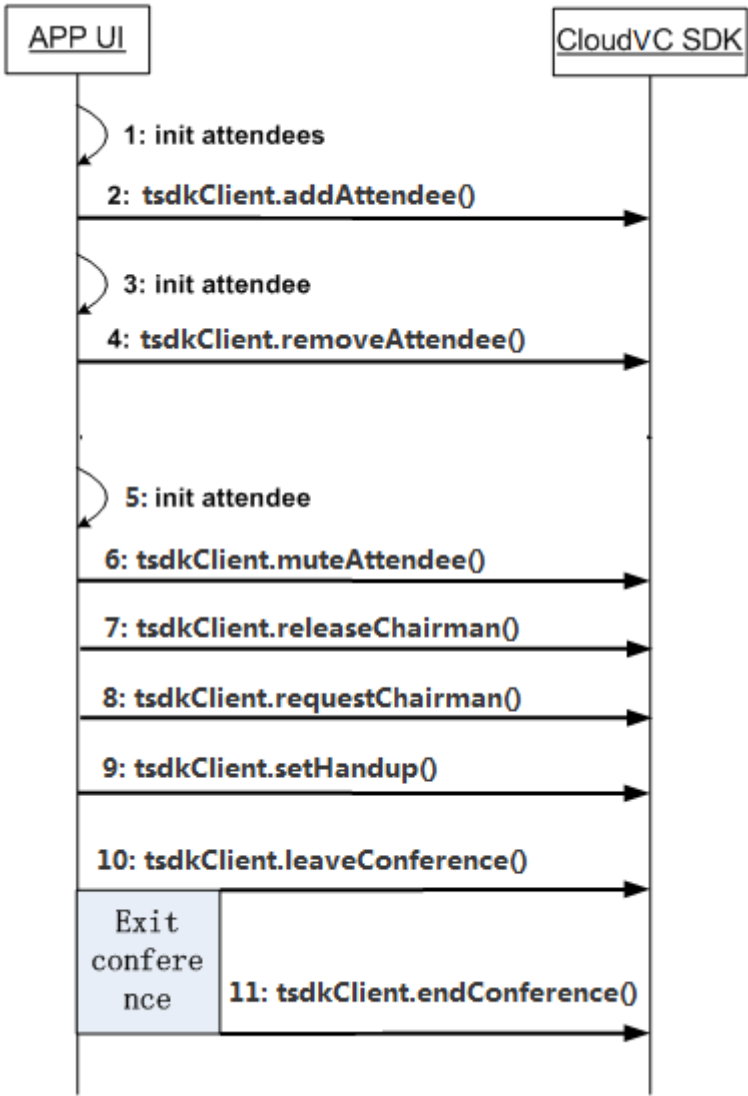
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

已加入会议。

流程说明

图 5-7 基本会控流程图



- 步骤1** 构建要加入会议的成员列表的数据结构addAttendeesInfo
- 步骤2** 调用tsdkClient.addAttendee()接口增加与会成员，第1步中的addAttendeesInfo作为参数。
- 代码示例：

```
function addAttendee() {
    var confHandle = handle
    var AttendeeList = document.getElementById("addAttendee_list").value;
    if (handle == undefined) {
        alert("Meeting has not yet started");
        return;
    } else if (AttendeeList == null || AttendeeList == "") {
        alert("attendee number is empty");
        return;
    }

    var AttendeeListArray = AttendeeList.split(",");
    var AttendeeLists = new Array();
```

```
for (var i = 0; i < AttendeeListArray.length; i++) {  
    AttendeeLists[i] = { number: AttendeeListArray[i], name: AttendeeListArray[i], role: 0 };  
}  
var addAttendeesInfo = {  
    attendeeNum: parseInt(document.getElementById("addAttendee_num").value),  
    attendeeList: AttendeeLists  
}  
tsdkClient.addAttendee(confHandle, addAttendeesInfo, (ret) => {});  
}
```

参考文件:

\\usage\\components\\conference_usage.js

步骤3 构建要移除会议成员的数据结构attendee

步骤4 调用tsdkClient.removeAttendee()接口删除指定的与会成员，第3步中的attendee作为参数。

代码示例:

```
function removeAttendee() {  
    var confHandle = handle;  
    var attendee = document.getElementById("removeAttendee").value  
    tsdkClient.removeAttendee(confHandle, attendee, (ret) => {});  
}
```

参考文件:

\\usage\\components\\conference_usage.js

步骤5 构建要静音的与会成员的数据结构attendee

步骤6 调用tsdkClient.muteAttendee()接口静音或者取消静音指定的与会成员，第6步中的attendee作为参数。

代码示例:

```
function muteAttendee() {  
    var confHandle = handle  
    var attendee = document.getElementById("muteAttendee").value  
    var isMuteObj = document.getElementById("is_Mute");  
    var isMute = parseInt(isMuteObj.options[isMuteObj.selectedIndex].value);  
    tsdkClient.muteAttendee(confHandle, attendee, isMute, function callback(ret) {});  
}
```

参考文件:

\\usage\\components\\conference_usage.js

步骤7 调用tsdkClient.releaseChairman()接口释放主席权限。

代码示例:

```
function releaseChairman() {  
    var confHandle = handle;  
    tsdkClient.releaseChairman(confHandle, (ret) => {});  
}
```

参考文件:

\\usage\\components\\conference_usage.js

步骤8 调用tsdkClient.requestChairman()接口申请主席权限。

代码示例:


```
function requestChairman() {  
    var confHandle = handle;  
    var password = document.getElementById("requestChairman").value  
    tsdkClient.requestChairman(confHandle, password, (ret) => {});  
}
```

参考文件:

\\usage\\components\\conference_usage.js

步骤9 调用tsdkClient.setHandup()接口将指定的与会成员举手或取消举手,普通与会者可以调用此方法设置或取消自己的举手。

代码示例:

```
function setHandup() {  
    var confHandle = handle  
    var isHandupObj = document.getElementById("set_hand_up");  
    var isHandup = parseInt(isHandupObj.options[isHandupObj.selectedIndex].value);  
    var attendee = document.getElementById("setHandup").value  
    tsdkClient.setHandup(confHandle, isHandup, attendee, function callback(ret) {});  
}
```

参考文件:

\\usage\\components\\conference_usage.js

步骤10 离开或退出会议

a. 调用tsdkClient.leaveConference()接口将离开会议,会议仍然在继续。

代码示例:

```
function leaveConference() {  
    if(handle) {  
        var confHandle = handle  
        tsdkClient.leaveConference(confHandle, function callback(ret) {});  
    }else{  
        alert("The conference handle does not exist")  
    }  
}
```

参考文件:

\\usage\\components\\conference_usage.js

b. 调用tsdkClient.endConference()接口将结束会议。

代码示例:

```
function endConference() {  
    if(handle) {  
        var confHandle = handle  
        tsdkClient.endConference(confHandle, function callback(ret) {});  
    }else{  
        alert("The conference handle does not exist")  
    }  
}
```

参考文件:

\\usage\\components\\conference_usage.js

----结束

注意事项

- 1. 调用addAttendee()、removeAttendee()、releaseChairman()和endConference()接口要求调用者必须为主席。
- 2. setHandup()接口用于被禁言的场景。
- 3. muteAttendee()接口，主席可对所有与会者设置或取消静音，普通与会者只可对自己设置或取消静音。

5.7 视频会控

应用场景

广播或取消广播指定与会者，选看与会者，本地麦克关闭/开启切换。

功能接口	融合会议 Hosted	融合会议 On-premise	备注
broadcastAttendee()	Y	Y	
watchAttendee()	Y	Y	IDo会控下所有人均可选看，取消选看，rest会控仅支持主席选看
muteMic()	Y	Y	

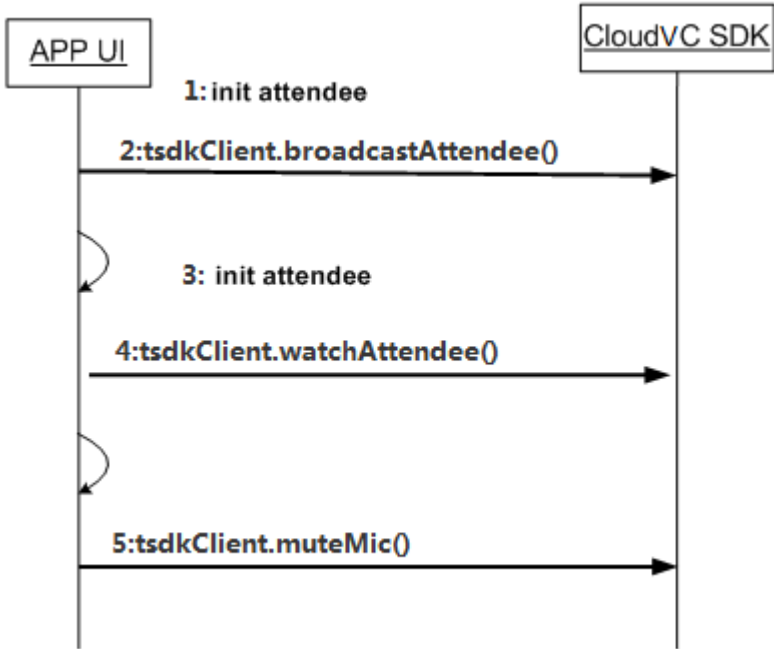
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

已加入会议。

流程说明

图 5-8 视频会控流程图



- 步骤1** 构建指定与会者的数据结构attendee
- 步骤2** 调用broadcastAttendee()接口广播或取消广播指定与会者，第1步中的attendee作为参数。

代码示例：

```
function broadcastAttendee() {
    var confHandle = handle
    var attendee = document.getElementById("broadcastAttendee").value
    var isBroadcastObj = document.getElementById("is_Broadcast");
    var isBroadcast = parseInt(isBroadcastObj.options[isBroadcastObj.selectedIndex].value);
    tsdkClient.broadcastAttendee(confHandle, attendee, isBroadcast, function callback(ret) {});
}
```

参考文件：

\usage\components\conference_usage.js

- 步骤3** 构建指定与会者的数据结构watchAttendeeInfo
- 步骤4** 调用watchAttendee()接口选看指定与会者，第3步中的watchAttendeeInfo作为参数。

代码示例：

```
function watchAttendee() {
    var confHandle = handle
    var watchAttendeeList = document.getElementById("watch_Attendee_list").value;
    if (handle == undefined) {
        alert("Meeting has not yet started");
        return;
    } else if (watchAttendeeList == null || watchAttendeeList == "") {
        alert("attendee number is empty");
        return;
    }
}
```

```
    }  
  
    var watchAttendeeListArray = watchAttendeeList.split(",");  
    var watchAttendeeLists = new Array();  
    for (var i = 0; i < watchAttendeeListArray.length; i++) {  
        watchAttendeeLists[i] = { number: watchAttendeeListArray[i], name:  
watchAttendeeListArray[i], role: 0 };  
    }  
    var watchAttendeeInfo = {  
        watchAttendeeNum: parseInt(document.getElementById("watch_Attendee_Num").value),  
        watchAttendeeList: watchAttendeeLists  
    }  
    tsdkClient.watchAttendee(confHandle, watchAttendeeInfo, (ret) => {});  
}
```

参考文件:

\\usage\\components\\conference_usage.js

步骤5 调用micMute()接口进行本地麦克关闭/开启切换。

代码示例:

```
function muteMic(bMute){  
    var callId = call_Id;  
    var isMute = parseInt(bMute);  
    var callbacks = function () {};  
    tsdkClient.muteMic(callId, isMute, callbacks)  
}
```

参考文件:

\\usage\\conference_usage.js

----结束

注意事项

1. 调用broadcastAttendee()接口时，在广播和声控模式下，主席可指定会场，取消广播时不需要指定与会者。

5.8 媒体设备

应用场景

获取和设置麦克风序号，获取和设置扬声器序号，获取和设置视频序号、设置麦克或扬声器的音量，获取麦克或扬声器音量，播放媒体铃声，停止播放媒体铃声。

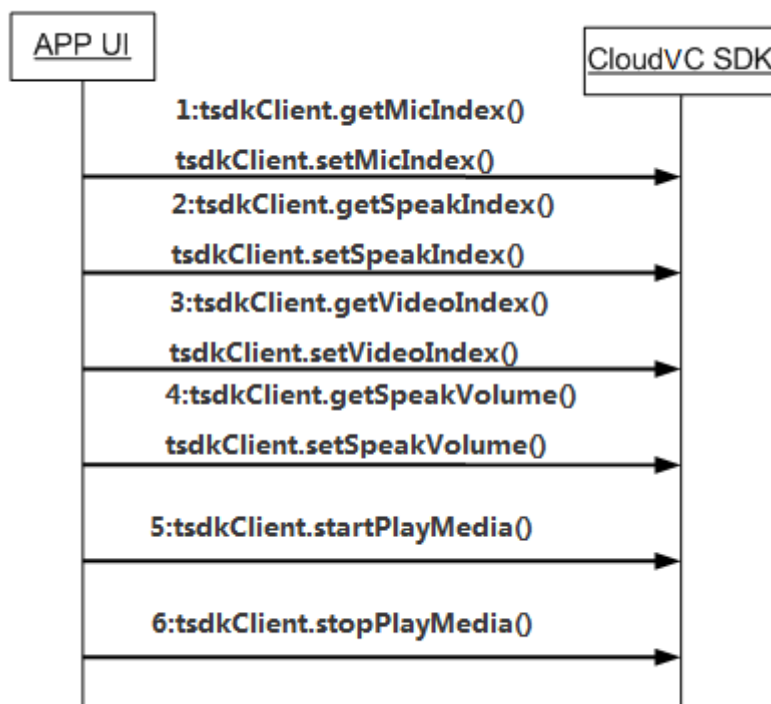
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

已登录。

流程说明

图 5-9 媒体设备



步骤1 调用tsdkClient.getMicIndex()接口获取麦克风序号，调用tsdkClient.setMicIndex()设置麦克风序号。

代码示例：

```
function setMicIndex() {
    var index;
    tsdkClient.getMicIndex((ret) => {
        index = ret.param.index
    })
    tsdkClient.setMicIndex(index, (ret) => {})
}
```

参考文件：

\\usage\\components\\media_device.html

步骤2 调用tsdkClient.getSpeakIndex()接口获取扬声器序号，调用tsdkClient.setSpeakIndex()接口设置扬声器序号。

代码示例：

```
function setSpeakIndex() {
    var index;
    tsdkClient.getSpeakIndex((ret) => {
        index = ret.param.index
    })
    tsdkClient.setSpeakIndex(index, (ret) => { })
}
```

参考文件：

\usage\components\media_device.html

步骤3 调用tsdkClient.getVideoIndex()接口获取视频序号，调用tsdkClient.setVideoIndex()接口设置视频序号。

代码示例：

```
function setVideoIndex() {
    var index;
    tsdkClient.getVideoIndex((ret) => {
        index = ret.param.index
    })
    setTimeout(() => {
        tsdkClient.setVideoIndex(index, (ret) => { })
    }, 0);
}
```

参考文件：

\usage\components\media_device.html

步骤4 调用tsdkClient.getSpeakVolume()接口获取输出音量，调用tsdkClient.setSpeakVolume()接口设置输出音量。

代码示例：

```
function getSpeakVolume() {
    tsdkClient.getSpeakVolume((ret) => {
        console.log(ret)
    })
}

function setSpeakVolume() {
    var speakervol = parseInt(document.getElementById("SpeakerVol"));
    tsdkClient.setSpeakVolume(speakervol, (ret) => { })
}
```

参考文件：

\usage\components\media_device.html

步骤5 调用startPlayMedia()接口播放媒体铃声。

代码示例：

```
var playHandle;
function startPlayMedia() {
    var mediaFilePath = document.getElementById("media_file_path").value;
    if(playHandle != null) {
        alert("Please do not repeat clicks!!")
    }else {
        tsdkClient.startPlayMedia(0, mediaFilePath, function(data) {
            if(data.result == 0){
                playHandle = data.param.playHandle;
            }
        });
    }
}
```

参考文件：

\usage\conference_usage.js

步骤6 调用stopPlayMedia()接口停止播放媒体铃声。

代码示例：

```
function stopPlayMedia() {
    if(playHandle == null) {
```

```
        alert("Audio file not playing!")
    }else{
        var callback = function () { };
        tsdkClient.stopPlayMedia(playHandle, callback);
        playHandle = null;
    }
}
```

参考文件：

\\usage\\conference_usage.js

----结束

注意事项

调用stopPlayMedia()接口时要有铃声正在播放才可调用此接口，否则报错。

5.9 音视频呼叫

应用场景

用户发起音视频呼叫，应答或拒绝呼叫，主动挂断呼叫，发送二次拨号信息，音视频通话相互转换，接受或拒绝音频转视频请求，两方通话转多方会议。

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

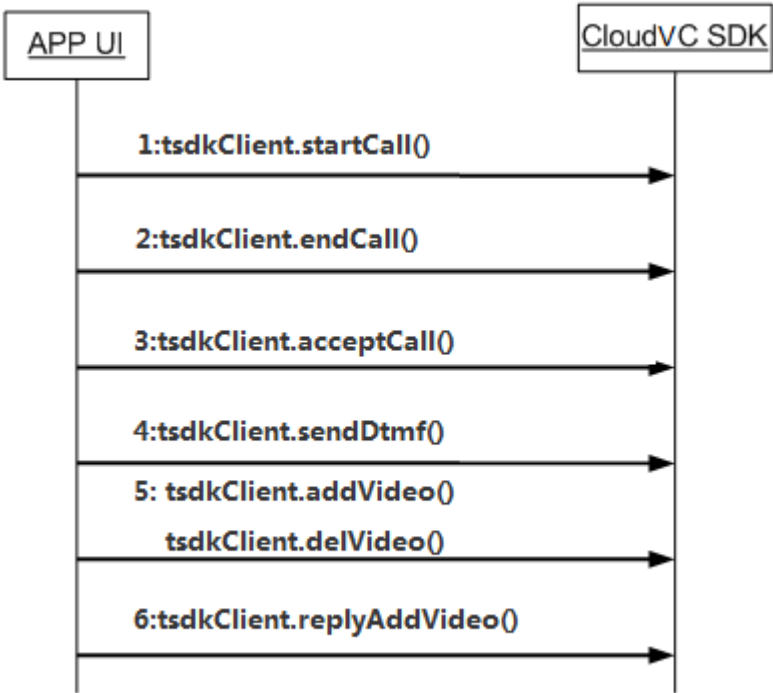
前提条件

已登录。

流程说明

图 5-10 音视频呼叫流程图

图 5-11



- 步骤1** a:主叫方调用tsdkClient.startCall()接口发起音频或视频呼叫。
b:被叫方调用tsdkClient.endCall()接口拒绝音频或视频呼叫。
c:被叫方调用tsdkClient.acceptCall()接口接听音频或视频呼叫。

代码示例：

```
function startCall() {
    console.log(isNoLogin)
    if (isNoLogin) {
        var isVideoCall = 0;
        var calleeNum = document.getElementById("callee_num").value;
        if (document.getElementById("isVideoCall").checked) {
            isVideoCall = 1;
        }
        tsdkClient.startCall(calleeNum, isVideoCall, function (data) {
            call_Id = data.param.callId
            if (data.result) {
                document.getElementById("callState").innerHTML = "call state: make call";
            }
        });
    } else {
        alert("please login first!")
    }
}

function endCall() {
    var callId = call_Id;
    if (call_Id == 0) {
        callId = callInComingId
    }
}
```



```
    }  
    var callbacks = function () {};  
    tsdkClient.endCall(callId, callbacks);  
    call_Id = 0;  
}  
function acceptCall() {  
    var callId = callInComingId;  
    call_Id = callInComingId;  
    console.log(callId)  
    var isVideo = document.getElementById("isVideoCall").checked;  
    var callbacks = function () {};  
    tsdkClient.acceptCall(callId, isVideo, callbacks);  
}
```

参考文件:

\\usage\\conference_usage.js

步骤2 调用tsdkClient.sendDtmf()接口发送DTMF信号。

代码示例:

```
function DTMF(dtmfNo) {  
    console.log(call_Id)  
    var callId = call_Id;  
    var tone = parseInt(dtmfNo);  
    var callback = function () {};  
    tsdkClient.sendDtmf(callId, tone, callback);  
}
```

参考文件:

\\usage\\conference_usage.js

步骤3 a:通话双方任意一方调用tsdkClient.addVideo()接口发起音频呼叫转视频呼叫请求。

b:通话双方任意一方调用tsdkClient.delVideo()接口发起视频呼叫转音频呼叫请求。

c:调用tsdkClient.replyAddVideo()接口响应添加视频请求

代码示例:

```
function addVideo() {  
    var callId = call_Id;  
    var callbacks = function () {};  
  
    tsdkClient.addVideo(callId, callbacks);  
}  
  
function delVideo() {  
    var callId = call_Id;  
    var callbacks = function () {};  
  
    tsdkClient.delVideo(callId, callbacks);  
}  
  
function replyAddVideo(accept) {  
    var callId = call_Id;  
    var callbacks = function () {};  
  
    tsdkClient.replyAddVideo(accept, callId, callbacks);  
}
```

参考文件:

\\usage\\conference_usage.js

----结束

注意事项

1. 调用tsdkClient.acceptCall()接口需在来电事件之后调用，否则失败。
2. 通话双方均可以调用endCall()接口挂断呼叫。
3. 处于通话中才可以调用addVideo()接口、sendDTMF()接口、delVideo()、replyAddVideo()接口。
4. 为了实现更友好的最终用户体验，UI应同步调用SDK提供的媒体播放接口或系统提供的播放接口，实现播放DTMF按键音。

6 问题定位

- 6.1 错误码列表
- 6.2 在线分析
- 6.3 日志分析

6.1 错误码列表

表 6-1 登录模块

模块	错误码（十进制值表示）	含义说明
登录模块	100000001	一般错误
	100000002	参数错误
	100000003	请求超时
	100000007	DNS解析异常
	100000008	网络异常
	100000009	鉴权失败
	100000010	sn匹配失败
	100000011	服务器异常，或网络不通
	100000012	账号被锁定
	100000017	查询服务器地址失败
	100000018	启动Token刷新失败
	100000019	系统不支持修改密码
	100000020	老密码错误
	100000021	新密码长度非法

模块	错误码（十进制值表示）	含义说明
	100000022	新密码复杂度不满足要求
	100000023	新密码不能与最近旧密码相同
	100000024	新密码不能包含3个以上重复字符
	100000029	用户名或者密码错误
	100000030	用户已被锁定
	100000031	nonce接入码错误
	100000032	服务器ca证书校验失败
	100000033	网络异常
	100000034	随机数错误
	100000035	缺少密码，请输入密码
	100000036	会议已经结束
	100000037	ip被锁定
	100000038	服务器返回重定向地址
	100000039	会议被锁定
	100000040	会议没有召开

表 6-2 呼叫模块

模块	错误码（十进制值表示）	含义说明
呼叫模块	200000001	一般错误
	200000002	参数错误
	200000003	分配内存错误
	200000004	系统错误
	200000005	发送消息错误
	200000006	获取系统配置错误
	200000007	物理网络错误
	200000008	网络接入错误
	200000009	创建定时器错误
	200000010	呼叫状态错误

模块	错误码（十进制值表示）	含义说明
	200000011	正在进行其他操作错误
	200000012	请求主控进行呼叫开始错误
	200000013	申请音频资源错误
	200000014	正在进行其他补充业务
	200000015	记录通话记录错误
	200000016	媒体进程返回的错误
	200000017	超过最大呼叫路数
	200000018	SIP账户ID不存在
	200000019	呼叫ID不存在
	200000020	正在注册中导致失败
	200000021	注册失败
	200000022	注销失败
	200000023	设置帐号信息错误
	200000024	设置SIP帐号失败
	200000025	上报SIP帐号信息失败
	200000026	服务器信息错误
	200000027	账户信息错误
	200000028	SIPC执行错误
	200000029	未注册错误
	200000030	调用Sip接口订阅失败
	200000031	注销订阅中错误
	200000032	订阅中错误
	200000033	只允许一个本地会议
	200000034	本地会议未创建
	200000035	与会者线路与主席账户不匹配
	200000036	本地会议状态错误
	200000037	IP Phone端联动状态上报失败
	200000038	会议ID不存在

模块	错误码（十进制值表示）	含义说明
	200000039	视频保持失败
	200000040	视频恢复失败
	200000041	服务器会议个数超出
	200000042	获取AA随机数失败
	200000043	AA登录失败
	200000044	会议类型不匹配
	200000045	视频会议开启失败
	200000046	AA无主服务器
	200000047	密码错误
	200000048	用户名错误
	200000049	用户已登录
	200000050	账户已锁定
	200000051	终端类型不匹配
	200000052	解析XML错误
	200000053	连接服务器错误
	200000054	获取媒体配置失败
	200000055	获取业务权限失败
	200000056	业务权限不足失败
	200000057	网络环境错误
	200000058	业务冲突
	200000059	连接超时
	200000060	未知错误
	200000061	添加振铃信息失败
	200000062	删除加振铃信息失败
	200000063	创建振铃号码失败
	200000064	VVM参数错误
	200000065	获取登记业务错误
	200000066	获取语音邮箱错误
	200000067	与会者已经存在
	200000068	与会者不存在

模块	错误码（十进制值表示）	含义说明
	200000069	创建服务器视频会议窗口失败
	200000070	视频窗口已存在
	200000071	获取会议列表失败
	200000072	需要主持人权限才能操作
	200000073	没有视频设备可以操作
	200000074	没有关闭刷新注册
	200000075	在线状态上报失败
	200000076	网络地址本订阅notify上报失败
	200000077	智真帐号被踢,info上报失败
	200000078	TLS根证书错误
	200000079	AD鉴权失败
	200000080	会议列表正在获取中
	200000081	禁止呼叫
	200000082	呼叫数达上限
	200000083	加密呼叫数达上限
	200000084	超出视频呼叫数，需要降为音频
	200000085	H323帐号ID不存在
	200000086	H323帐号信息错误
	200000087	H323帐号信息上报失败
	200000088	本端资源不足
	200000089	不支持该呼叫协议
	200000090	设置H323帐号信息错误
	200000091	mediax会议接入号上报界面失败
	200000092	上报Mediavx VMR信息到界面失败
	200000093	上报获取IMS会议列表信息到界面失败

模块	错误码（十进制值表示）	含义说明
	200000400	服务器返回"错误请求",响应码:400
	200000401	服务器返回"未经授权",响应码:401
	200000402	服务器返回"付费要求",响应码:402
	200000403	服务器返回"禁止",响应码:403
	200000404	服务器返回"未发现",响应码:404
	200000405	服务器返回"方法不允许",响应码:405
	200000406	服务器返回"不可接受",响应码:406
	200000407	服务器返回"需要代理授权",响应码:407
	200000408	服务器返回"请求超时",响应码:408
	200000410	服务器返回"离开",响应码:410
	200000413	服务器返回"请求实体太大",响应码:413
	200000414	服务器返回"请求URL太长",响应码:414
	200000415	服务器返回"不支持的媒体类型",响应码:415
	200000416	服务器返回"不支持的URL计划",响应码:416
	200000420	服务器返回"不良扩展",响应码:420
	200000421	服务器返回"需要扩展",响应码:421
	200000480	服务器返回"临时失效",响应码:480
	200000481	服务器返回"呼叫/事务不存在",响应码:481
	200000482	服务器返回"发现环路",响应码:482

模块	错误码（十进制值表示）	含义说明
	200000483	服务器返回"跳数太多",响应码:483
	200000484	服务器返回"地址不完整",响应码:484
	200000485	服务器返回"不明朗",响应码:485
	200000486	服务器返回"这里忙",响应码:486
	200000487	服务器返回"请求终止",响应码:487
	200000488	服务器返回"这里请求不可接受",响应码:488
	200000491	服务器返回"未决请求",响应码:491
	200000493	服务器返回"不可辨识",响应码:493
	200000500	服务器返回"服务器内部错误",响应码:500
	200000501	服务器返回"不可执行",响应码:501
	200000502	服务器返回"坏网关",响应码:502
	200000503	服务器返回"服务无效",响应码:503
	200000504	服务器返回"服务器超时",响应码:504
	200000505	服务器返回"版本不支持",响应码:505
	200000513	服务器返回"消息太大",响应码:513
	200000600	服务器返回"全忙",响应码:600
	200000603	服务器返回"丢弃",响应码:603
	200000604	服务器返回"不存在",响应码:604
	200000606	服务器返回"不可接受",响应码:606

模块	错误码（十进制值表示）	含义说明
	200000801	证书错误
	200000802	接收证书失败
	200000803	注册信息，需要重发(UI不需要处理)
	200000804	注册信息错误
	200000805	注册表过期
	200000806	GK注销
	200000807	GK注销要求重新注册
	200000808	注册认证失败
	200000809	被叫不在线
	200000810	本端未注册
	200000811	注册过期
	200000812	安全原因
	200000813	非法输入
	200000814	响应超时
	200000815	GK 路由呼叫
	200000816	主从确定失败
	200000817	MCU媒体加密资源不足而导致失败
	200000818	非标信令原因
	200000819	其他错误
	200000820	H323会场号码重名
	200000821	H323找不到GK
	200000822	H323对端没有响应
	200000823	媒体加密协商失败
	200000824	带宽不够导致呼叫失败
	200000825	带宽能力不足导致视频协商失败
	200000826	编解码不匹配导致视频协商失败
	200000827	分辨率能力低导致视频协商失败

模块	错误码（十进制值表示）	含义说明
	200000828	视频协商失败
	200000829	H323ARQ超时
	200000830	H323注册码无效
	200000831	H323注册序列号重复
	200000832	H323断开呼叫超时
	200000833	服务器地址错误
	200000834	SIP TCP建立失败
	200000835	SIP TLS建立失败
	200000836	SIP 会话心跳超时挂断
	200000837	无码流被挂断
	200000838	IP地址变化导致的挂断
	290000001	SIP注册处理超时
	290000002	被抢登录，强制注销
	290000003	播放音频文件错误
	290000004	停止播放的音频文件不存在
	290000005	视频会议无法盲转

表 6-3 会控模块

模块	错误码（十进制值表示）	含义说明
会控模块	300000001	一般错误
	300000002	参数错误
	300000003	超时
	300000006	请求消息异常
	300000007	鉴权失败
	300000008	服务器异常
	300000012	无效的URL
	300000150	鉴权信息验证失败，token 设置错误

模块	错误码（十进制值表示）	含义说明
	300011002	调用接口不符合系统限制（如：音频会议中调用广播与会者接口）
	300011004	权限不足
	300011013	用户不存在/账号或会议ID不存在
	300011030	会议资源不足
	300011033	没有会议主席
	300011038	会议时长错误
	300011040	会议开始时间早于当前时间
	300011097	条件错误
	390000001	入会失败，会议接入码或会议ID或会议密码错误
	390000002	视频通道连接失败
	390000003	内存使用过高，请释放内存

表 6-4 会议模块

模块	错误码（十进制值表示）	含义说明
会议模块	400000001	警告
	400000002	错误
	400000003	失败
	400000004	空指针
	400000005	调用超时
	400000006	参数错误
	400000009	没有权限
	400000012	调用线程错误
	400000101	拒绝用户请求
	400000102	网络错误
	400000105	服务器超出最大连接数
	400000106	license过期

模块	错误码（十进制值表示）	含义说明
	400000107	会议已结束
	400000110	会议状态错误
	400000111	鉴权时发现该用户不存在
	400000112	会议中channel达到阈值
	400000124	Audio打开MIC数量已超过设置的最大打开MIC数
	400000126	鉴权失败
	400000127	服务器返回未知错误
	400000130	会议锁定
	400000131	鉴权超时
	400000132	会议密码错误
	400000133	会议数超过限制
	400000134	会议成员超过限制
	400000135	没有足够的数据License，不允许入会
	400000136	鉴权License超时，不允许入会
	400000138	用户重复入会，不允许入会
	400000301	用户被踢
	400000302	清除token值
	400000303	系统异常
	400000304	网络错误
	400000305	超时
	400000306	未知错误
	400000307	最大License范围
	400000308	系统错误
	400000309	用户离线
	400000400	未初始化
	400000401	重复初始化
	400000402	重复入会
	400000403	未入会

模块	错误码（十进制值表示）	含义说明
	400000407	会议已结束
	400000408	会议已离开
	400000412	参数错误
	400000413	会议环境尚未初始化完成
	400000414	自己已经是主讲人
	400000415	电话未初始化
	400000416	加入电话会议失败
	400000418	参数无效
	400000419	相同的会议已经存在
	400000421	证书生效时间大于当前时间，代表证书还未生效
	400000422	证书过期时间小于当前时间，代表证书已过期
	400001002	错误的用户ID
	400001202	参数异常
	400001203	角色异常
	400001206	共享状态异常
	400001208	创建共享进程异常
	490000001	参数无效
	490000002	参数类型无效

表 6-5 即时消息模块

模块	错误码（十进制值表示）	含义说明
即时消息模块	500000000	登录成功
	500000001	登录失败
	500000002	密码错误
	500000003	帐号不存在
	500000004	用户已登录（登录依然成功，其他已经登录的终端提示被踢下线）
	590000001	参数错误

模块	错误码（十进制值表示）	含义说明
	590000002	操作失败

表 6-6 通讯录模块

模块	错误码（十进制值表示）	含义说明
通讯录模块	600000001	一般错误
	600000002	参数错误

表 6-7 通用错误模块

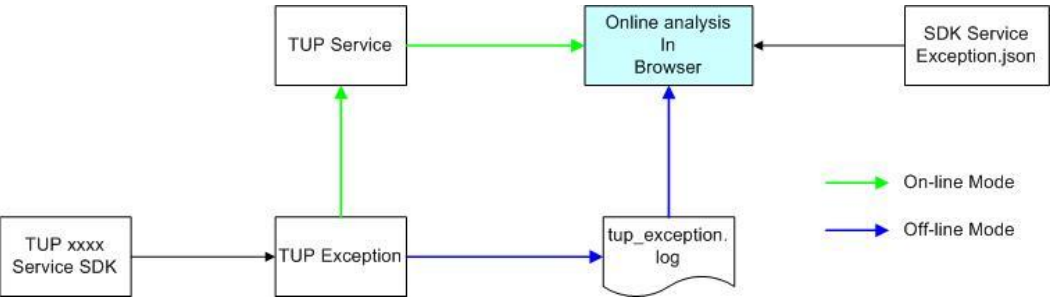
模块	错误码（十进制值表示）	含义说明
	900000001	websocket断开，请重新连接
	900000002	初始化失败
	900000003	参数类型错误
	900000004	参数错误
	900000005	登录状态错误，请重新登录
	900000006	对象不存在
	900000007	参数在给定值范围之外
	900000008	已经在会议或呼叫中
	900000009	权限错误
	900000010	会议状态错误

6.2 在线分析

介绍

SDK提供问题“在线分析”机制，以协助开发者在使用SDK开发过程中更方便直观的分析定位并解决问题。

图 6-1 原理流程



说明

- 1、业务SDK向TUP Exception抛出“异常”，由TUP Exception向TUP Service推送“异常”信息，并将“异常”信息写入tup_exception.log日志。
- 2、在线模式下，“在线分析”呈现页面接受TUP Service推送的“异常”信息，离线模式下，“在线分析”呈现页面读取日志路径下的“tup_exception.log”中记录的“异常”信息，结合“SDK Service exception”各种类型异常的详细描述文件，进行实时的“异常分析”，给开发者呈现关心的、可理解的问题原因分析和方案建议。
- 3、tupException.html在CloudEC_Client_API_Demo_Windows_JS.zip的web_server_demo\tool目录中，tup_exception.log在TSDK后台进程目录中生成，默认路径为C:\Program Files\TSDK。

在线实时分析

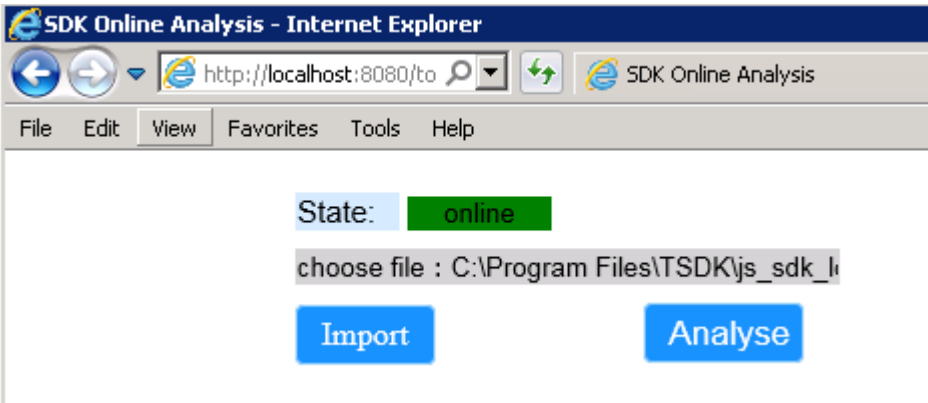
说明

前提条件：应用程序已开启并运行“TUP Service”。

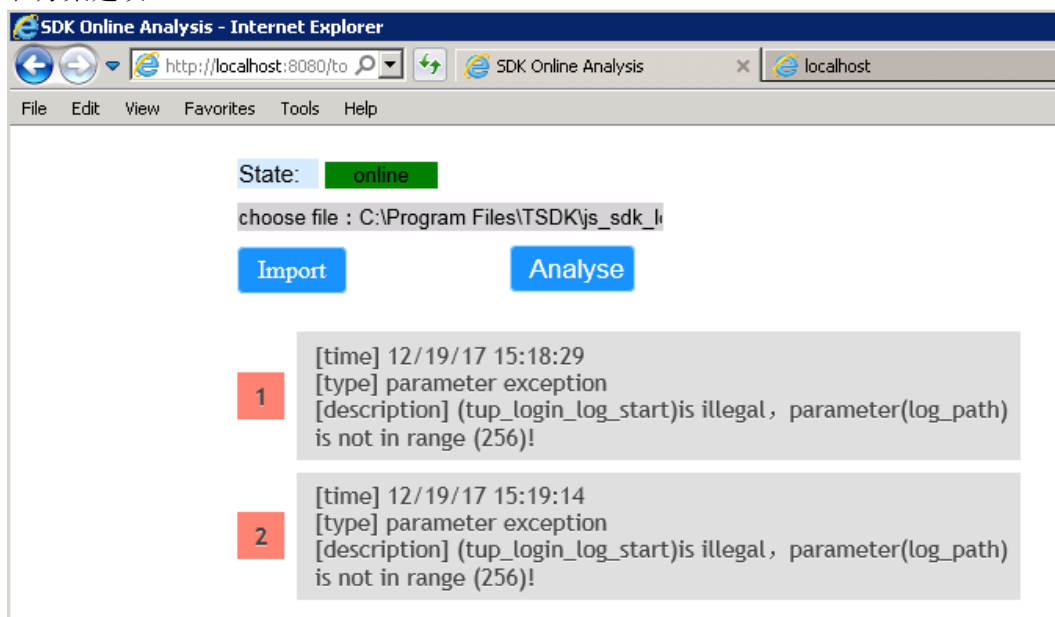
步骤1 浏览器中打开tupException.html，“SDK Online Analysis”呈现页面连接“TUP Service”，并显示连接状态正常。

说明

若“TUP Service”未启动或其他原因，则显示连接状态异常。



步骤2 当TUP运行过程中产生“异常”时，页面将实时显示关心的、可理解的问题原因分析和方案建议。



----结束

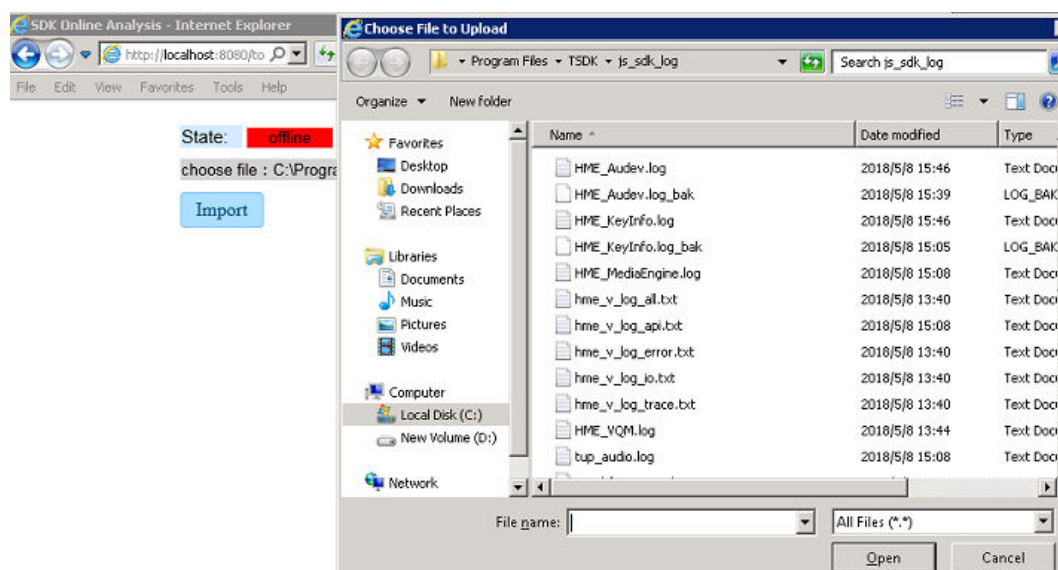
离线导入分析

说明

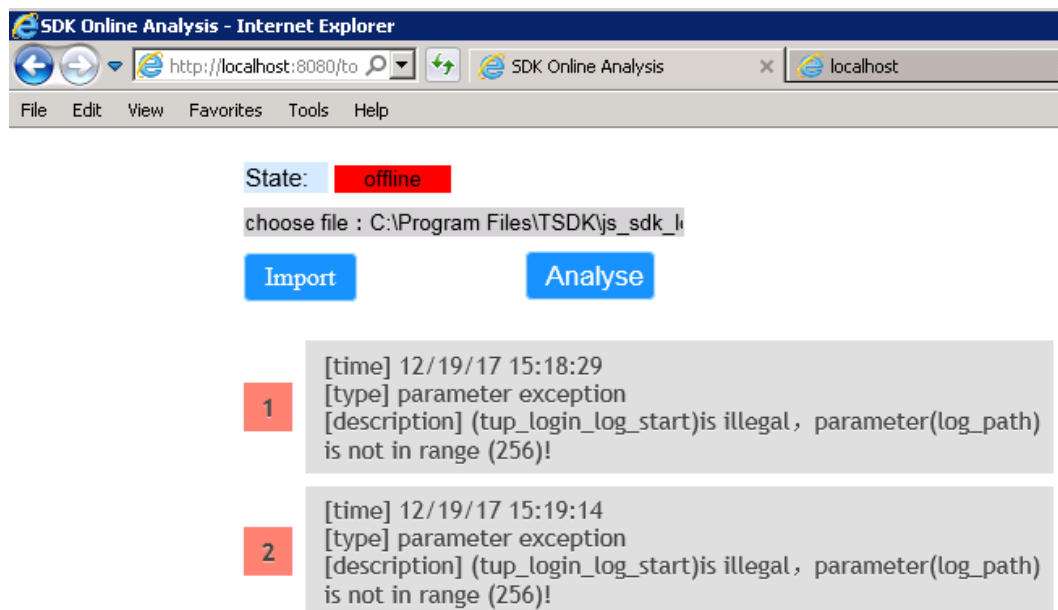
在移动端或其他应用程序无法运行“TUP Service”的情况下，开发者可以通过导入日志的方式进行“异常”分析。

步骤1 浏览器中打开tupException.html，"SDK Online Analysis"呈现页面显示连接状态异常。

步骤2 开发者在"SDK Online Analysis"呈现页面中，点击“Import”按钮，选取异常日志“tup_exception.log”。



步骤3 点击“Analyse”按钮后，页面即可显示关心的、可理解的问题原因分析和方案建议。



----结束

6.3 日志分析

获取日志

TSDK的日志在TSDK的安装目录中js_sdk_log文件夹中，native视频窗口日志在用户的\AppData\Roaming\TSDK目录中。

日志分析

1. SDK日志的一般格式为：

[YYYY-MM-DD HH:MM:SS.MS][P:xxxx/T:xxxx][level][filename:line function]log
info string

其中：

[YYYY-MM-DD HH:MM:SS.MS] 表示日志打印时间，精确到ms；

[P:xxxx/T:xxxx] 表示日志所在的进程ID和线程ID；

[level] 表示日志级别，包括错误(Err)、警告(War)、信息(Inf)和调试(Debug)级别；

[filename:line function]表示日志所在的文件名，行号和函数；

log info string 为日志信息描述。

建议开发者根据时间顺序进行相应的业务调用分析。

2. 对于业务的关键逻辑位置，会打印一条相应的关键信息日志。

```
[2017-05-05 16:29:35.077][P:7920/T:8776][Inf][call][call_msg.c:881  
CALLMPROC_MSG_AsynSend]ulMsgID = 0x00000008[CALL_E_EVT_CALL_DESTROY], ulParam123 =  
[0x7bdfb600, 1, 0], from [P7920] to [P7920]  
[2017-05-05 16:29:35.077][P:7920/T:8776][Dbg][call][call_basic.c:969  
callbasicDestroyBasicCall]MEDIA_SetAudioPlayDelay(0) 0  
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][call][call_basic.c:15740  
CallBasicSetVideoMute]callID:0x7bdfb600, bIsMute=0  
[2017-05-05 16:29:35.079][P:7920/T:8776][Inf][call][call_basic.c:15762  
CallBasicSetVideoMute]3. ulCallID = 2078258688, bIsMute=0  
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][call][call_conf.c:5110
```

```
CallConfOnEndConfCall]CallConfOnEndConfCall ulConfID=0x0
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][call][call_basic.c:11348
CallBasicGetConfID]CallBasicGetConfID(0x7bdfb600)
[2017-05-05 16:29:35.079][P:7920/T:8776][Err][call][call_basic.c:11354
CallBasicGetConfID]Get Call ID(0x7bdfb600) Error=0x8002113
[2017-05-05 16:29:35.079][P:7920/T:8776][Err][call][call_conf.c:5123
CallConfOnEndConfCall]con't find ServerConf by ID=0xffffffff!
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][call][call_main.c:2128
callmainMsgProcModTsp]call process msg leave, msgId: 0x2904
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][SipTptd][sstpthiglu.c:292
SipStackTptDToTptMsgProc]start process tpt to tptd msg: TPTD_SENDRESULT_SUCC
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][SipTptd][sstpthiglu.c:394
SipStackTptDToTptMsgProc]end process tpt to tptd msg: TPTD_SENDRESULT_SUCC
```

7 附录

[7.1 更新驱动](#)

[7.2 https访问配置](#)

[7.3 index.html完整代码](#)

[7.4 TSDK安装包制作](#)

7.1 更新驱动

查看 WebGL

在Chrome浏览器中输入chrome://gpu，查看WebGL是否是Hardware accelerated，如果是Software only，请升级驱动。



Note: To properly save this page, select the "Webpage, Complete" option in the Save File dialog.

Graphics Feature Status

- Canvas: **Hardware accelerated**
- CheckerImaging: **Disabled**
- Flash: **Hardware accelerated**
- Flash Stage3D: **Hardware accelerated**
- Flash Stage3D Baseline profile: **Hardware accelerated**
- Compositing: **Hardware accelerated**
- Multiple Raster Threads: **Enabled**
- Native GpuMemoryBuffers: **Software only. Hardware acceleration disabled**
- Rasterization: **Hardware accelerated**
- Video Decode: **Hardware accelerated**
- Video Encode: **Hardware accelerated**
- WebGL: **Hardware accelerated**
- WebGL2: **Hardware accelerated**

升级驱动

请根据CPU类型进行选择：

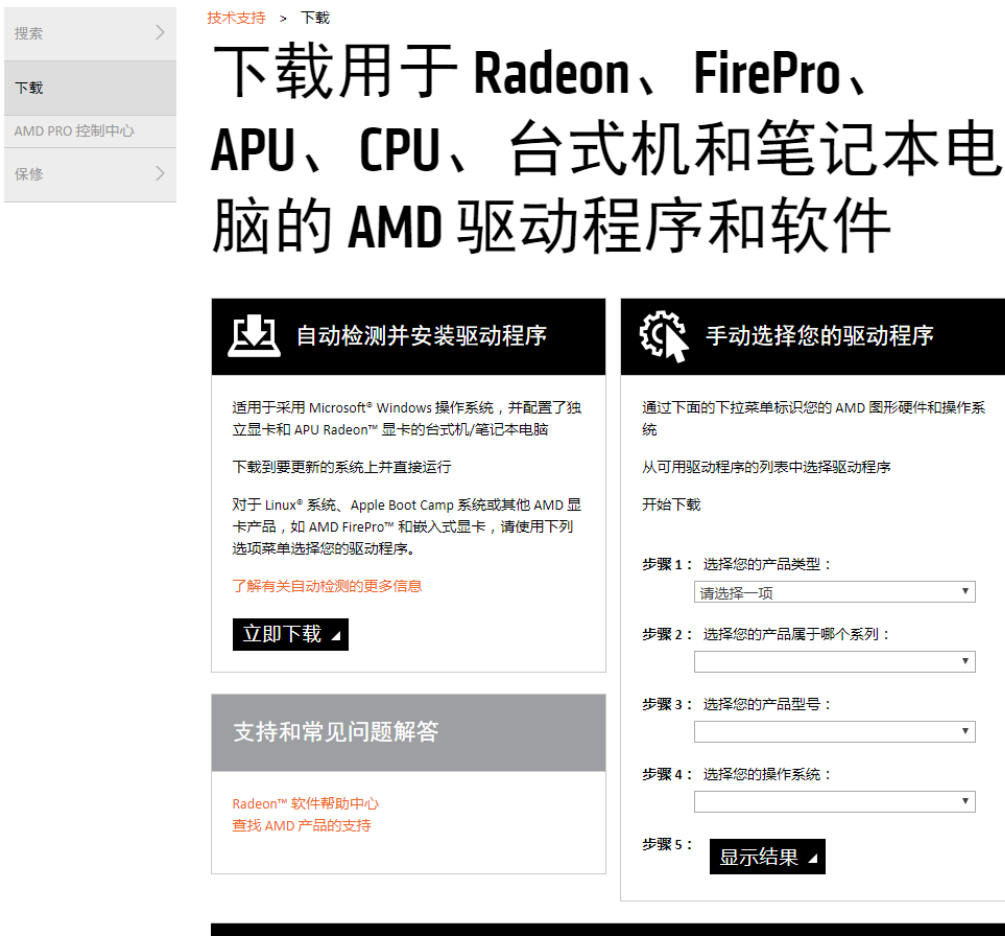
● intel

请到intel官网<https://downloadcenter.intel.com/zh-cn/>，根据您机器的硬件升级到相应最新驱动，安装后重启机器即可。



● AMD

请到AMD官网<http://support.amd.com/zh-cn/download/>，根据您机器的硬件升级到相应最新驱动，安装后重启机器即可。



7.2 https 访问配置

目的：浏览器基于安全考虑，通过https访问时需要配套websocket加密使用wss而不能使用ws，本章节主要说明这种场景下如何设置开启wss连接，其中涉及localhost域名和对应域名的证书。

步骤1 确认部署的服务器是否已开启https服务，若已开启请忽略此步骤。以tomcat为例：

```
</Connector>
-->

<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="D:\server.keystore"
    keystorePass="*****">
    <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
</Connector>
```

步骤2 在本地浏览器中安装安全证书，您可以用自己生成配套域名的证书，也可以使用出厂的默认证书，默认证书路径在TSDK的安装目录下(server.crt)。

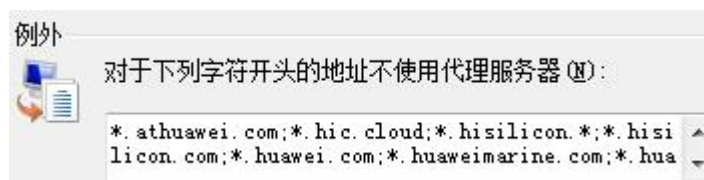
步骤3 在demo中开启WSS配置。

```
var options = {
    domain: "localhost.cloudec.huaweicloud.com",
    isWSS: 1,
    isTlsSupport: 0,
    isWithSBC: 0,
    dropFrame: 2,
}
cloudEC.configure(options)
```

须知

如果企业内无法访问外网，需要部署域名服务器，并设置localhost.cloudec.yourcompany.com绑定127.0.0.1，域名可以自定义，对应域名的证书也需要配套生成。

如果企业内可以访问外网，但开启了代理；需要把域名解析加代理例外，如下图所示，否则访问任何网站都是证书不可信。



----结束

7.3 index.html 完整代码

```
<html>
<head>
  <meta charset="UTF-8">
  <script src="/sdk/TerminalSDK.js" /></script>
  <!--begin:webcomponentsts init-->
  <script>
    if (!window.customElements) {
      document.write('<!--');
    } else {
      window.ShadyDOM = {
        force: true
      };
      window.customElements.forcePolyfill = true;
    }
  </script>
  <script src="/usage/components/webcomponentsjs-1.0.20/custom-elements-es5-adapter.js"></script>
</head>
<!--! do not remove -->
<script src="/usage/components/webcomponentsjs-1.0.20/webcomponents-loader.js"></script>
<!--end:webcomponentsts init-->
<link rel="import" href="/usage/components/book_conf.html">
<script>
  var root = window
  var tsdkJsInitParam = {
    invokeMode: 1,
    svrAddr: "localhost.cloudec.huaweicloud.com",
    svrPort: "7684",
    ssl: 1
  };
  root.isNoLogin = false;
  var listeners = {
    OnEvtAuthSuccess: (ret) => {
      console.debug('auth success!');
      alert("auth success!" + JSON.stringify(ret))
    },
    OnEvtLoginSuccess: (ret) => {
      console.info('login success!' + JSON.stringify(ret));
      root.isNoLogin = true;
      alert("login success!")
    },
    OnEvtAuthFailed: (ret) => {
      root.isNoLogin = false;
      alert("OnEvtAuthFailed!")
    },
    OnEvtLogoutSuccess: (ret) => {
      alert("logout Success!")
    },
    OnEvtLogoutFailed: (ret) => {
      alert('logout failed!')
    },
    OnEvtCallOutgoing: (ret) => {
      root.callOutGoingId = ret.param.callId
      alert("Call out success!")
    },
    OnEvtCallIncoming: (ret) => {
      root.callInComingId = ret.param.callId
      alert("Please answer" + JSON.stringify(ret))
    },
    OnEvtOpenVideoReq: (ret) => {
      alert("Audio to video")
    },
    OnEvtCloseVideoInd: (ret) => {
      alert("video to audio")
    },
    OnEvtOpenVideoInd: (ret) => {
```

```
        alert("video to audio")
    },
    OnEvtRefuseOpenVideoInd: (ret) => {
        alert("rejected!!!")
    },
    OnEvtCallEnded: (ret) => {
        alert("call end success!");
        root.callEndId = ret.param.callId
        console.log(JSON.stringify(ret))
    },
    OnEvtBookConfResult: (ret) => {
        console.log("book join successs" + JSON.stringify(ret))
    },
    OnEvtQueryConfListResult: (ret) => {
        root.confList = ret
    },
    OnEvtJoinConfResult: (ret) => {
        console.log("join sunccess" + JSON.stringify(ret))
        root.handle = ret.param.handle
    },
    OnEvtQueryConfListResult: (ret) => {
        root.callbackDatas = ret
    },
    OnError: function(ret) {
        if (390000003 == ret.info.errorCode) {
            console.warn("Memory usage over 80%, please close the unrelated program.");
        } else {
            alert(JSON.stringify(ret));
        }
    },
};
terminalSDK.createTsdkClient(tsdkJsInitParam, listeners, (data)=>{
    root.tsdkClient = data
});
if (root.tsdkClient == null) {
    console.log("createTsdkClient");
}
function login() {
    var configParam = {
        logParam: 1,
        tlsParam: 1,
        proxyParam: 1,
        serviceSecurityParam: 1,
        iptServiceConfigParam: 1,
        localAddress: 1,
        filePathInfo: 1,
        dpiInfo: 1,
        networkInfo: 1,
        ipCallSwitch: 0,
        confCtrlParam: 1,
        sendDataSwitch: 1,
        baseInfoParam: 1,
        frameParam: 1,
        visibleInfo: 1
    }
    var callback = function () { }
    tsdkClient.setConfigParam(configParam, callback);
    // 初始化
    var tsdkAppInfoParam = {
        "clientType": 0,
        "productName": "SoftClient on Desktop",
        "deviceSn": "1",
        "supportAudioAndVideoCall": 1,
        "supportAudioAndVideoConf": 1,
        "supportDataConf": 1,
        "supportCtd": 0,
        "supportIm": 0,
        "supportRichMediaMessage": 0,
        "supportEnterpriseAddressBook": 0,
```



```
        "useUiPlugin": 1,
        "isWsInvokeMode": 1,
    };
    var callbacks = function (res) { }
    tsdkClient.init(tsdAppInfoParam, (res) => {
        if (res.result == 0) {
            alert("init success!")
        }
    });
    console.log(isNoLogin)
    var userName = document.getElementById("name").value;
    var password = document.getElementById("passwd").value;
    var serverAddr = document.getElementById("svr_addr").value;
    var serverPort = document.getElementById("svr_port").value;
    var tsdkLoginParam = {
        "userId": 1,
        "authType": 0,
        "userName": userName,
        "password": password,
        "userTiket": "1",
        "serverType": 2, //SMC
        "serverVersion": "",
        "serverAddr": serverAddr, //SMC 192.160.54.207
        "serverPort": parseInt(serverPort) // 5061
    };
    var callbacks = function (res) {
        console.log(res)
    }
    console.debug('login begin');
    tsdkClient.login(tsdLoginParam, callbacks);

    console.debug('login ret:' + callbacks);
    passwd = "";
    proxyPassword = "";
    proxyParam = "";
}
//end login
function joinInstanceConf() {
    var joinNumber = 0;
    var confTypeObj = document.getElementById("instance_conf_type");
    var isVideoJoin = parseInt(confTypeObj.options[confTypeObj.selectedIndex].value);
    var confId = document.getElementById("conferenceId").value;
    var accessNumber = document.getElementById("accessNumber").value;
    var confPassword = document.getElementById("confPasswd").value;
    var TsdConfJoinParam = {
        confId: confId,
        accessNumber: accessNumber,
        confPassword: confPassword
    }
    tsdkClient.joinConference(joinNumber, isVideoJoin, TsdConfJoinParam, function
callback(ret) {
        console.info("join conference callback")
    });
    confPasswd = "";
    joinConfParam = "";
}
function logout() {
    var callbacks = function () { };
    tsdkClient.logout(callbacks);
    // tsdkClient.uninit(callbacks);
    //change UI to login
    document.getElementById("login").style.display = "block";
    document.getElementById("call").style.display = "none";
}
</script>
</head>
<body>
    <h2>CloudVC JS SDK Hello World</h2>
    <div id="tab-content1" class="tab-content">
```

```
<fieldset>
  <legend>User login</legend>
  <div id="login">
    server address: <input type="text" id="svr_addr" value="192.160.54.207" />
    port: <input type="text" id="svr_port" value="5061" />
    username: <input type="text" id="name" value="" />
    password: <input type="password" id="passwd" value="" />
    <button onclick="login()">login</button>
  </div>
</fieldset>
</div>
<div id="usage">
  <fieldset>
    <legend>User logout</legend>
    <div id="userinfo"></div>
    <button onclick="logout()">logout</button>
  </fieldset>
  <fieldset>
    <legend>Instance conference</legend>
    conference type:<select id='instance_conf_type'>
      <option value=0>No video access</option>
      <option value=1>Video access</option>
    </select>
    conference ID:<input type="text" id="conferenceId"/>
    access code:<input type="text" id="accessNumber"/>
    conference password: <input type="password" id="confPasswd" value="" />
    <button onclick="joinInstanceConf()">instance conference</button>
  </fieldset>
  <fieldset>
    <legend>Book conference</legend>
    <cloudec-bookconf/>
  </fieldset>
</div>
</body>
</html>
```

7.4 TSDK 安装包制作

向您展示TSDK安装包的制作，安装及卸载以及指导您如何自定义修改参数。

7.4.1 环境准备

请确保满足以下开发环境要求:

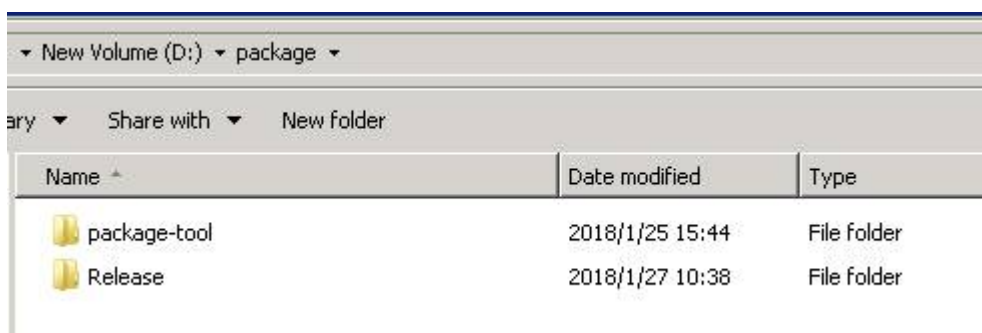
环境和工具名称	版本要求	说明
操作系统	Windows 7专业版	
Inno setup	5.5.9	下载链接 ，文件为innosetup-5.5.9.exe
Visual Studio	2012	用于编译源码，生成tsdk_service_tray.exe 若使用默认发布的tsdk_service_tray.exe，则不需要此工具。
Microsoft Visual C++ 2008 Redistributable	9.0.30729.17	下载链接 ，文件为vcredist_x86.exe

说明

语言文件（ChineseSimplified.isl和EnglishBritish.isl）在
CloudEC_Client_API_Demo_Windows_JS.zip中的make_package_demo目录中。

7.4.2 TSDK 安装包快速制作步骤

步骤1 在D盘创建打包目录package，解压CloudEC_Client_API_Demo_Windows_JS.zip，将make_package_demo目录中package-tool和Release文件夹整体拷贝到package目录中。

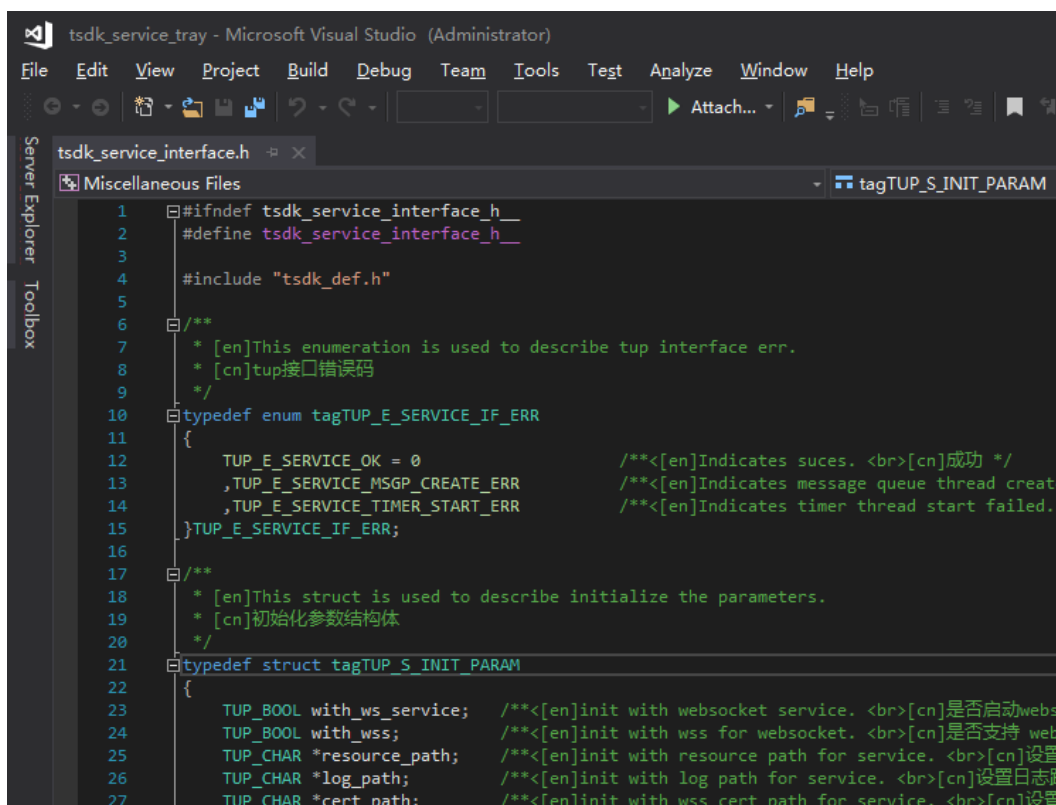


步骤2 将下载的vcredist_x86.exe拷贝到D:\package\package-tool目录中。

步骤3 进入CloudEC_Client_API_Demo_Windows_JS的client_deamon_demo目录中。

a. 若使用默认的tsdk_service_tray.exe，将bin目录中的tsdk_service_tray.exe拷贝到D:\package\package-tool\sdk目录中。

b. 将eSDK_EC_API_XXX_windows_c.zip中lib目录中的tupService.lib文件拷贝到tsdk_service_tray\lib中，用Visual Studio 2012打开tsdk_service_tray.sln文件，点击图示按钮编译生成tsdk_service_tray.exe，将Release目录中的tsdk_service_tray.exe拷贝到D:\package\package-tool\sdk目录中。

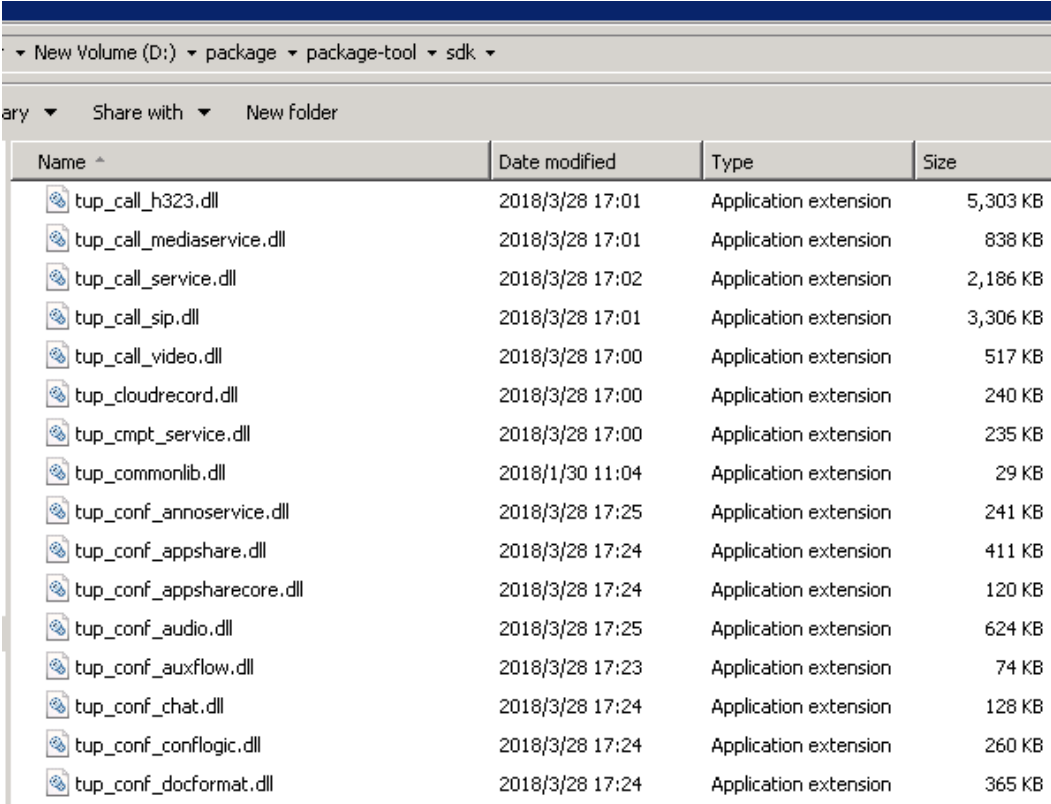


说明

XXX为版本号。

tsdk_service_tray.exe仅首次制作TSDK安装包时才需要编译生成，后续制作可以复用已生成的tsdk_service_tray.exe。

步骤4 准备依赖的库文件，将eSDK_EC_API_XXX_windows_c.zip中dll和plugin目录的文件拷贝到D:\package\package-tool\sdk目录中。



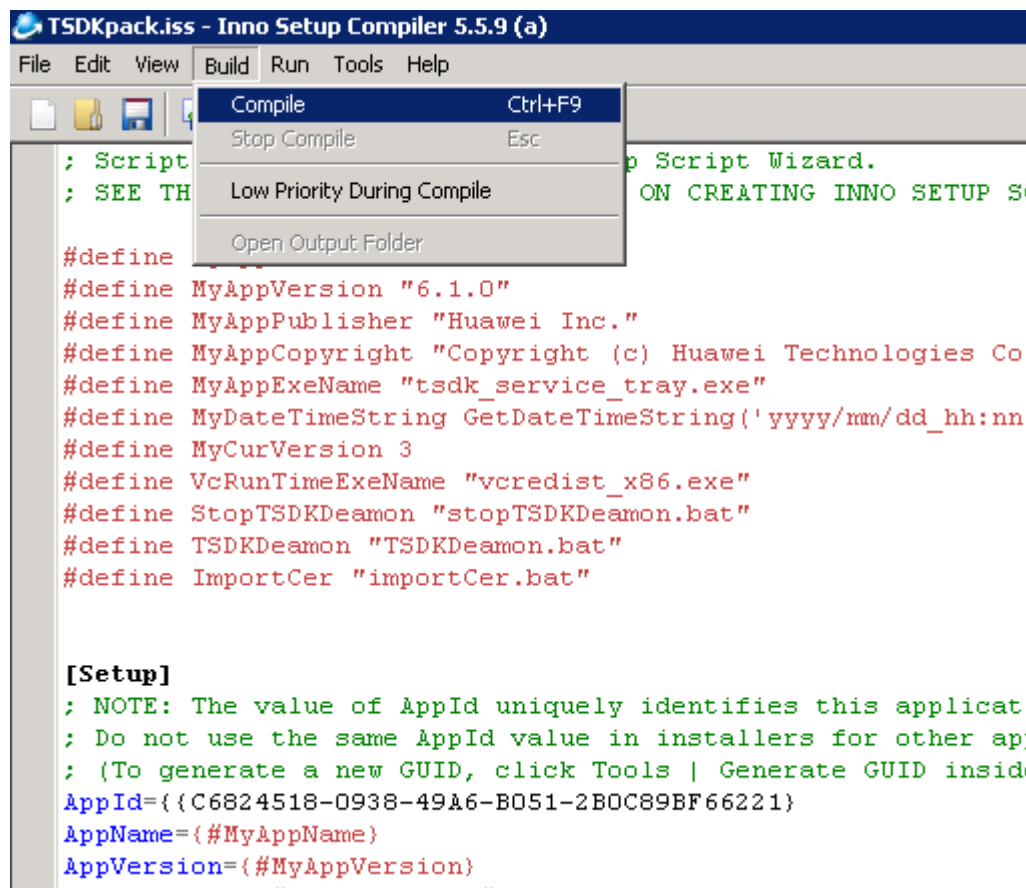
Name	Date modified	Type	Size
tup_call_h323.dll	2018/3/28 17:01	Application extension	5,303 KB
tup_call_mediaservice.dll	2018/3/28 17:01	Application extension	838 KB
tup_call_service.dll	2018/3/28 17:02	Application extension	2,186 KB
tup_call_sip.dll	2018/3/28 17:01	Application extension	3,306 KB
tup_call_video.dll	2018/3/28 17:00	Application extension	517 KB
tup_cloudrecord.dll	2018/3/28 17:00	Application extension	240 KB
tup_cmpt_service.dll	2018/3/28 17:00	Application extension	235 KB
tup_commonlib.dll	2018/1/30 11:04	Application extension	29 KB
tup_conf_annoservice.dll	2018/3/28 17:25	Application extension	241 KB
tup_conf_appshare.dll	2018/3/28 17:24	Application extension	411 KB
tup_conf_appsharecore.dll	2018/3/28 17:24	Application extension	120 KB
tup_conf_audio.dll	2018/3/28 17:25	Application extension	624 KB
tup_conf_auxflow.dll	2018/3/28 17:23	Application extension	74 KB
tup_conf_chat.dll	2018/3/28 17:24	Application extension	128 KB
tup_conf_conflogic.dll	2018/3/28 17:24	Application extension	260 KB
tup_conf_docformat.dll	2018/3/28 17:24	Application extension	365 KB

说明

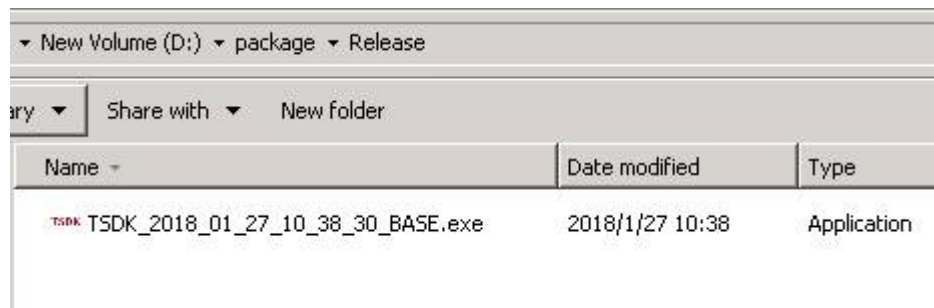
XXX为版本号。

步骤5 将D:\package\package-tool\sdk目录中的tup_service_daemon.exe文件复制并粘贴到当前目录，重命名为tup_service_s.exe。

步骤6 安装Inno setup，安装完成后，将ChineseSimplified.isl和EnglishBritish.isl拷贝到Inno Setup 5\Languages目录下。启动安装包编译，双击D:\package\package-tool目录下的TSDKpack.iss，点击Compile按钮或者使用快捷键Ctrl+F9启动安装包编译。



步骤7 获取TSDK安装包，编译完成后在D:\package\Release目录下获取TSDK安装包。

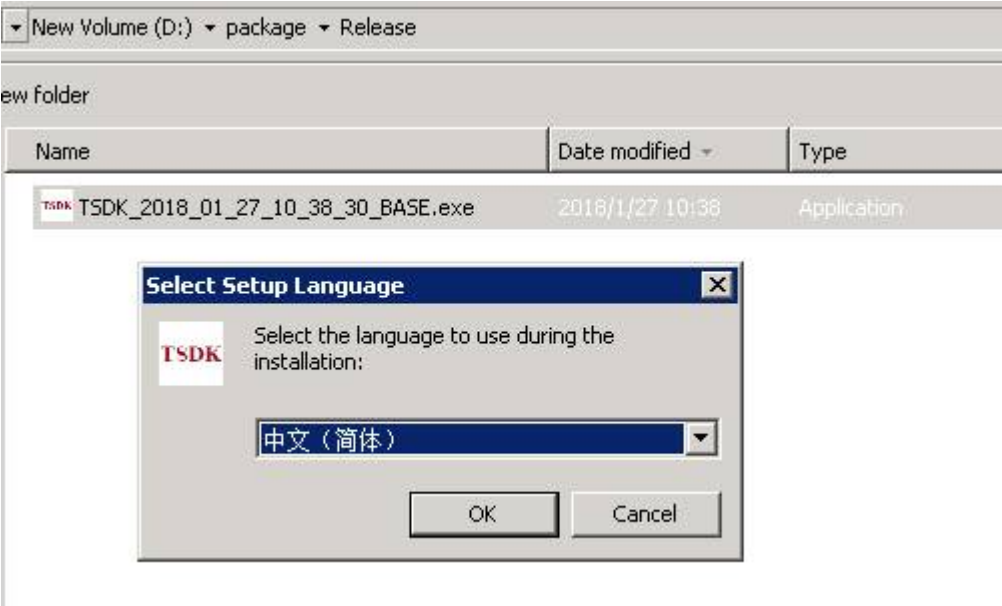


----结束

7.4.3 TSDK 安装及卸载

- 安装

双击安装包TSDK_XXX_BASE.exe进入安装向导，根据引导完成安装。

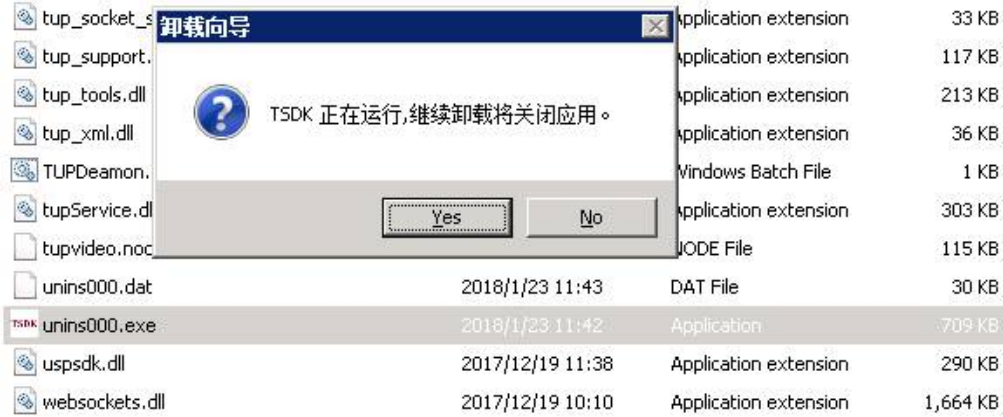


须知

在Windows 10系统上，应用程序以非“管理员身份”运行时，默认无写系统盘权限（AppData目录除外），若将TSDK服务安装在系统盘（AppData目录除外），将会导致日志文件无权限生成。建议安装TSDK服务安装至默认路径，或非系统盘。

● 卸载

在TSDK安装目录下，双击unins000.exe进入卸载向导，根据引导完成卸载。

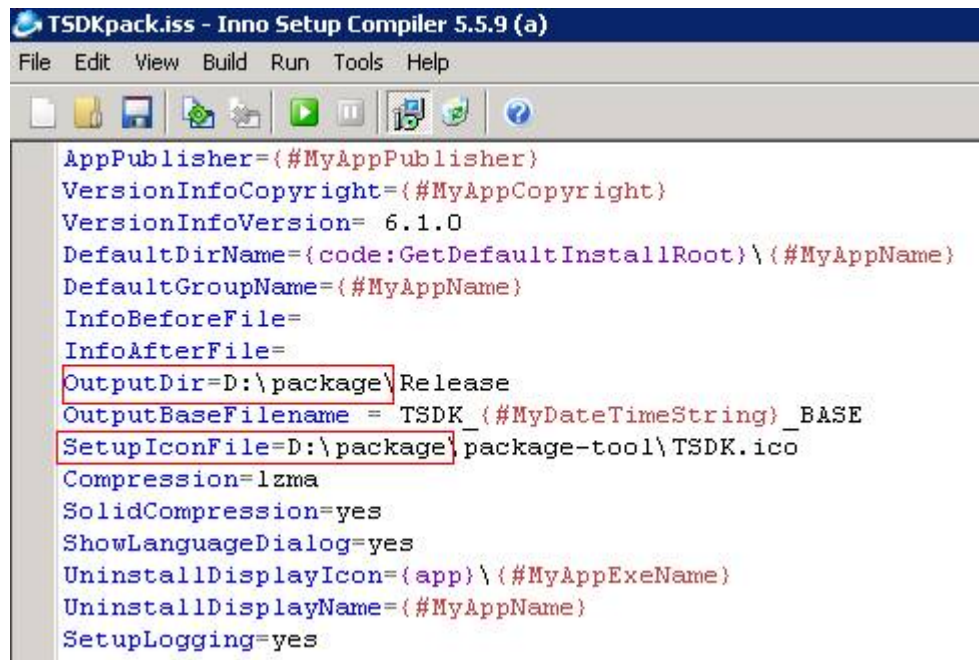


7.4.4 进阶指南

7.4.4.1 打包目录路径自定义修改

若不想将打包目录放在D盘根目录，可根据需要自定义修改。

将整个package目录放在您需要的路径，再打开\package\package-tool目录下的TSDKpack.iss，修改目录相关的内容：



```
TSDKpack.iss - Inno Setup Compiler 5.5.9 (a)
File Edit View Build Run Tools Help

AppPublisher={#MyAppPublisher}
VersionInfoCopyright={#MyAppCopyright}
VersionInfoVersion= 6.1.0
DefaultDirName={code:GetDefaultInstallRoot}\{#MyAppName}
DefaultGroupName={#MyAppName}
InfoBeforeFile=
InfoAfterFile=
OutputDir=D:\package\Release
OutputBaseFilename = TSDK_{#MyDateTimeString}_BASE
SetupIconFile=D:\package\package-tool\TSDK.ico
Compression=lzma
SolidCompression=yes
ShowLanguageDialog=yes
UninstallDisplayIcon={app}\{#MyAppExeName}
UninstallDisplayName={#MyAppName}
SetupLogging=yes
```




```
TSDKpack.iss - Inno Setup Compiler 5.5.9 (a)
File Edit View Build Run Tools Help

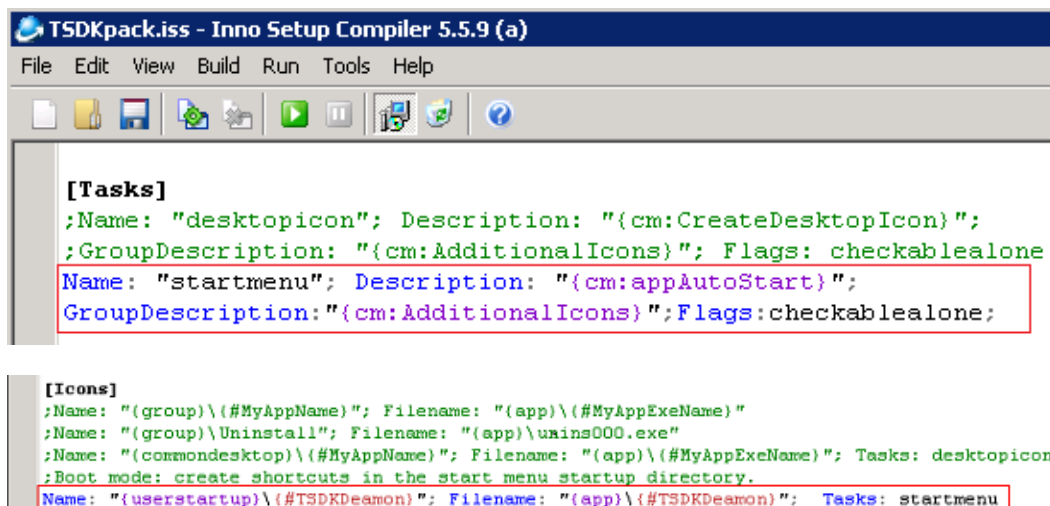
Root:HKLM; Subkey:"SOFTWARE\Microsoft\Windows\CurrentVersion\Uni
Root:HKLM; Subkey:"SOFTWARE\Microsoft\Windows\CurrentVersion\Uni
Root:HKCU; Subkey:"Software\Microsoft\Windows\CurrentVersion\Uni
Root:HKCU; Subkey:"Software\Microsoft\Windows\CurrentVersion\Uni
Root:HKLM; Subkey:"SOFTWARE\Microsoft\Windows\CurrentVersion\Uni
Root:HKLM; Subkey:"SOFTWARE\Microsoft\Windows\CurrentVersion\Uni
Root:HKCU; Subkey:"Software\Microsoft\Windows\CurrentVersion\Uni
Root:HKCU; Subkey:"Software\Microsoft\Windows\CurrentVersion\Uni
Root:HKCU; Subkey:"Software\Microsoft\Windows\CurrentVersion\Uni
Root:HKCU; Subkey:"Software\Microsoft\Windows\Windows Error Repo
Root:HKCU; Subkey:"Software\Microsoft\Windows\Windows Error Repo
Root:HKCU; Subkey:"Software\Classes\TSDKLaunch\DefaultIcon"; Val
Root:HKCU; Subkey:"Software\Classes\TSDKLaunch\shell\open\comman
Root:HKCR; SubKey:"TSDK"; ValueType: string; ValueData: "TSDK pr
Root:HKCR; SubKey:"TSDK"; ValueType: string; ValueName: "URL Pro
Root:HKCR; SubKey:"TSDK\DefaultIcon"; ValueType: string; ValueDa
Root:HKCR; SubKey:"TSDK\shell\open\command"; ValueType: string;

[Files]
Source: "D:\package\package-tool\sdk\*"; DestDir: "{app}"; Flags
; NOTE: Don't use "Flags: ignoreversion" on any shared system fi
; VC Redistribute
Source: "D:\package\package-tool\vcredist_x86.exe"; DestDir: "{a
Source: "D:\package\package-tool\TSDKDeamon.bat"; DestDir: "{app
Source: "D:\package\package-tool\stopTSDKDeamon.bat"; DestDir: "
Source: "D:\package\package-tool\importCer.bat"; DestDir: "{app}
Source: "D:\package\package-tool\cert.pfx"; DestDir: "{app}";
Source: "D:\package\package-tool\server.crt"; DestDir: "{app}";
Source: "D:\package\package-tool\root_cert.pem"; DestDir: "{app}
Source: "D:\package\package-tool\root_cert_huawei.pem"; DestDir:
Source: "D:\package\package-tool\server.pem"; DestDir: "{app}";
Source: "D:\package\package-tool\server.key"; DestDir: "{app}";
Source: "D:\package\package-tool\map_tree.xml"; DestDir: "{app}"

[Icons]
;Name: "{group}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"
;Name: "{group}\Uninstall"; Filename: "{app}\unins000.exe"
;Name: "{commondesktop}\{#MyAppName}"; Filename: "{app}\{#MyAppE
```

7.4.4.2 开机自启动

开机自启动的设置如需修改，请打开\package\package-tool目录下的TSDKpack.iss，目前默认是开机自启动，如需关闭开机自启动，请将下面图示代码注释掉即可。



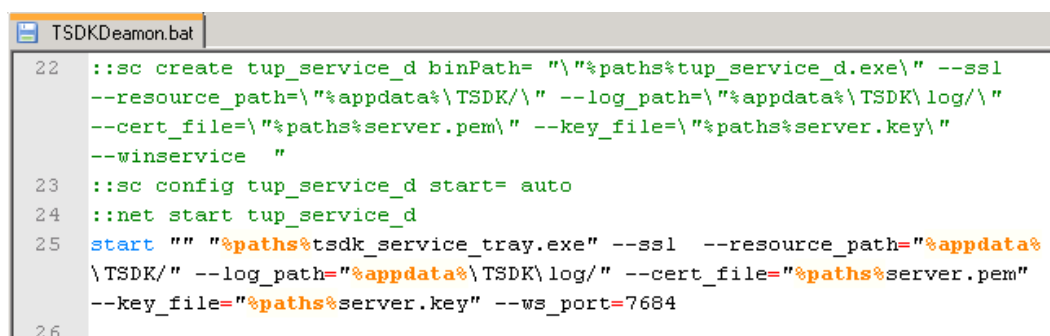
说明

如需手动启动TSDK，请在TSDK安装目录下，双击TSDKDaemon.bat启动TSDK。

tscsvn.dll	2017/12/19 11:38	Application extension	577 KB
TSDK tsdk_service_tray.exe	2018/5/7 13:33	Application	19 KB
TSDKDaemon.bat	2018/5/7 15:11	Windows Batch File	1 KB
tup_call_audio.dll	2018/1/30 15:14	Application extension	361 KB
tup_call_bfcpl.dll	2018/1/30 15:14	Application extension	271 KB
tup_call_mediaservice.dll	2018/1/30 15:15	Application extension	811 KB
tup_call_service.dll	2018/1/30 15:15	Application extension	2,108 KB
tup_call_sip.dll	2018/1/30 15:15	Application extension	3,260 KB
tup_call_video.dll	2018/1/30 15:14	Application extension	466 KB
tup_cmpt_service.dll	2018/1/30 15:14	Application extension	235 KB

7.4.4.3 修改证书及业务端口

如果需要修改TSDK的证书及业务端口等参数，请用编辑器打开\package\package-tool目录下的TSDKDaemon.bat，根据需要修改相关的内容：



8

修订记录

发布日期	文档版本	修订说明
2019-03-15	05	配套19.1.RC1版本，文档刷新发布
2019-01-02	04	配套19.0.0版本，文档刷新发布。
2018-10-12	03	配套19.0.RC1版本，文档刷新发布，主要更新包括： 1、新增IM模块的群组管理、个人通讯录、状态与提醒和即时消息功能； 2、错误码列表章节增加即时消息模块错误码。 3、优化会议控制能力，提升会议控制实时性，提升会议用户体验。
2018-07-07	02	配套6.1.0版本，文档刷新发布，主要更新包括： 1. 新增对融合会议On-premise组网支持； 2. 视频会议章节新增设置远端多画面功能； 3. 媒体设备章节新增播放/停止播放媒体铃声功能； 4. 新增音视频呼叫章节； 5. 新增IPT业务章节； 6. 新增Native视频窗口设置章节； 7. 错误码列表章节增加呼叫模块错误码。
2018-05-10	01	配套6.1.0.RC1版本(暂仅支持融合会议Hosted组网)，文档首次发布。