

## Ideashare SDK 开发指南

# Ideashare SDK 开发指南

文档版本

01

发布日期

2022-03-04



**版权所有 © 华为技术有限公司 2022。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暗示的声明或保证。

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://e.huawei.com>

# 目 录

<b>1 SDK 下载</b>	<b>1</b>
<b>2 Android SDK</b>	<b>2</b>
2.1 概述	2
2.2 变更记录	2
2.3 快速入门	2
2.3.1 预置条件	2
2.3.2 开发环境准备	2
2.3.3 SDK 快速集成	3
2.4 典型场景	6
2.4.1 初始化	6
2.4.2 查询投影码	6
2.4.3 连接/断开连接	7
2.4.4 共享/停止共享	7
2.4.5 遥控器	8
2.5 常见问题	9
2.5.1 获取日志	9
2.5.2 混淆文件	9
2.6 Android 接口参考	9
2.6.1 概述	9
2.6.2 变更记录	10
2.6.3 接口参考	10
2.6.3.1 基础配置	10
2.6.3.1.1 初始化	10
2.6.3.1.2 日志设置	11
2.6.3.2 业务接口（主动调用）	12
2.6.3.2.1 设置设备类型	12
2.6.3.2.2 设备发现	13
2.6.3.2.3 连接	14
2.6.3.2.4 断开连接	15
2.6.3.2.5 投屏前权限确认	15
2.6.3.2.6 开始投屏	16
2.6.3.2.7 停止投屏	17
2.6.3.2.8 获取投屏状态	18

2.6.3.2.9 获取 SDK 版本.....	18
2.6.3.2.10 获取智慧屏最新版本信息.....	18
2.6.3.2.11 获取远程服务状态信息.....	19
2.6.3.2.12 打开/关闭麦克风.....	20
2.6.3.2.13 调节扬声器音量.....	21
2.6.3.2.14 打开/关闭扬声器.....	22
2.6.3.2.15 打开/关闭摄像机.....	22
2.6.3.2.16 离开会议.....	23
2.6.3.2.17 使用证书鉴权连接.....	24
2.6.3.2.18 开启/关闭 音频投放.....	24
2.6.3.2.19 自定义通知.....	25
2.6.3.2.20 开启兼容性.....	26
2.6.3.3 业务接口（全局推送 IShareEventHandler 方法）.....	27
2.6.3.3.1 远程服务状态信息变化通知.....	27
2.6.3.3.2 共享状态变化通知.....	28
2.6.4 说明.....	29
2.6.4.1 公网地址说明.....	29
2.6.4.2 权限说明.....	29
2.7 错误码参考.....	29
2.7.1 错误码.....	29

## 3 IOS SDK..... 31

3.1 概述.....	31
3.2 变更记录.....	31
3.3 快速入门.....	31
3.3.1 预置条件.....	31
3.3.2 开发环境准备.....	31
3.3.3 SDK 快速集成.....	32
3.4 典型场景.....	37
3.4.1 场景 1： 初始化.....	37
3.4.2 场景 2： 查询投屏码.....	38
3.4.3 场景 3： 设置 TLS 证书验证.....	39
3.4.4 场景 4： 连接/断开连接.....	40
3.4.5 场景 5： 开始投屏.....	41
3.4.6 场景 6： 停止投屏.....	43
3.4.7 场景 7： 投放音频.....	44
3.4.8 场景 8： 遥控器.....	45
3.4.9 场景 9： 设备推送.....	47
3.4.10 场景 10： 错误上报.....	48
3.4.11 场景 11： 去初始化.....	48
3.5 常见问题.....	49
3.5.1 获取日志.....	49
3.6 IOS 接口参考.....	50

3.6.1 概述.....	51
3.6.2 变更记录.....	51
3.6.3 接口参考.....	51
3.6.3.1 启动 APP.....	51
3.6.3.1.1 IdeaShareServiceController : 投屏控制类，主要是提供业务接口.....	51
3.6.3.1.2 初始化日志.....	51
3.6.3.1.3 初始化 SDK.....	52
3.6.3.1.4 去初始化 SDK.....	52
3.6.3.1.5 投屏策略配置.....	53
3.6.3.1.6 投屏码解析.....	53
3.6.3.2 连接设备.....	54
3.6.3.2.1 设置 TLS 验证.....	54
3.6.3.2.2 设备连接.....	54
3.6.3.2.3 断开连接.....	55
3.6.3.2.4 开始投屏.....	55
3.6.3.2.5 结束投屏.....	56
3.6.3.2.6 远程遥控功能.....	56
3.6.3.2.7 打开/关闭麦克风.....	57
3.6.3.2.8 调节扬声器音量.....	57
3.6.3.2.9 打开/关闭扬声器.....	58
3.6.3.2.10 打开/关闭摄像头.....	58
3.6.3.2.11 离开会议.....	59
3.6.3.2.12 结束会议.....	59
3.6.3.2.13 获取客户端 SDK 版本.....	60
3.6.3.2.14 服务端版本查询.....	60
3.6.3.2.15 辅助进程发送视频数据.....	60
3.6.3.2.16 辅助进程发送音频数据.....	61
3.6.3.2.17 辅助进程投屏初始化.....	61
3.6.3.2.18 接口调用结果通知.....	61
3.6.3.2.19 discoverCallback.....	61
3.6.3.2.20 connectCallback.....	62
3.6.3.2.21 disconnectCallback.....	62
3.6.3.2.22 startSharingCallback.....	63
3.6.3.2.23 stopSharingCallback.....	63
3.6.3.2.24 remoteServiceStatusCallback.....	63
3.6.3.2.25 micMuteCallback.....	64
3.6.3.2.26 speakerVolumeCallback.....	64
3.6.3.2.27 speakerMuteCallback.....	65
3.6.3.2.28 cameraMuteCallback.....	65
3.6.3.2.29 leaveConfCallback.....	66
3.6.3.2.30 finishConfCallback.....	66
3.6.3.2.31 sdkLatestVersionCallback.....	67

3.6.3.2.32 tlsCallback.....	67
3.6.3.3 设备推送通知.....	67
3.6.3.3.1 onRemoteServiceStatusChangedNotify.....	68
3.6.3.3.2 OnDevConfStatusNotify.....	68
3.6.3.3.3 OnShareStatusChangeNotify.....	69
3.6.3.4 错误通知.....	70
3.6.3.4.1 OnErrorNotify.....	70
3.6.4 附录.....	70
3.6.4.1 枚举定义.....	70
3.6.4.1.1 LOG_LEVEL.....	70
3.6.4.1.2 IDEA_SHARE_APP_DEFINAION_QUAITY.....	71
3.6.4.1.3 IDEA_SHARE_APP_CONF_STATUS.....	71
3.6.4.1.4 IDEA_SHARE_APP_AUX_STATUS.....	71
3.6.4.1.5 IDEA_SHARE_APP_SERVER_STATUS.....	71
3.6.4.1.6 IDEA_SHARE_APP_ERR.....	72
3.6.4.1.7 IDEA_SHARE_APP_TLS_VERIFY_MODE.....	73
3.6.4.2 OC 类定义.....	73
3.6.4.2.1 IdeaShareServiceAppInfo.....	74
3.6.4.2.2 IdeaShareServiceLogInfo.....	74
3.6.4.2.3 SharePolicy.....	74
3.6.4.2.4 DiscoverResult.....	74
3.6.4.2.5 DeviceInfo.....	74
3.6.4.2.6 RemoteServiceStatus.....	75
3.6.4.2.7 IdeaShareServiceTlsInfo.....	75
3.6.4.3 SDK 接口头文件定义.....	75
3.6.4.3.1 IdeaShareEventHandler.h.....	76
3.6.4.3.2 IdeaShareServiceController.h.....	76
3.6.4.4 权限说明.....	76
3.7 错误码参考.....	76
3.7.1 错误码.....	76
<b>4 MAC SDK.....</b>	<b>77</b>
4.1 概述.....	77
4.2 变更记录.....	77
4.3 快速入门.....	77
4.3.1 预置条件.....	77
4.3.2 开发环境准备.....	78
4.3.3 SDK 快速集成.....	78
4.4 典型场景.....	86
4.4.1 典型场景接口调用概览.....	87
4.4.2 场景 1： 初始化.....	88
4.4.3 场景 2： 查询投影码.....	89
4.4.4 场景 4： TLS 验证.....	90

4.4.5 场景 5：连接/断开连接.....	90
4.4.6 场景 6：共享/停止共享.....	91
4.4.7 场景 7：投放音频.....	91
4.4.8 场景 8：遥控器.....	92
4.4.9 场景 9：设备推送.....	93
4.4.10 场景 10：错误上报.....	94
4.4.11 场景 11：去初始化.....	95
4.4.12 场景 12：鼠标样式更改.....	95
4.5 常见问题.....	95
4.5.1 获取日志.....	95
4.5.2 声卡驱动安装.....	96
4.6 Mac 接口参考.....	97
4.6.1 概述.....	97
4.6.2 变更记录.....	97
4.6.3 接口参考.....	97
4.6.3.1 启动 APP.....	97
4.6.3.1.1 IdeaShareServiceController : 投屏控制类，主要是提供业务接口.....	97
4.6.3.1.2 初始化日志.....	98
4.6.3.1.3 初始化 SDK.....	98
4.6.3.1.4 去初始化 SDK.....	99
4.6.3.1.5 投屏策略配置.....	99
4.6.3.1.6 投屏码解析.....	100
4.6.3.2 连接设备.....	100
4.6.3.2.1 证书验证.....	100
4.6.3.2.2 设备连接.....	101
4.6.3.2.3 断开连接.....	101
4.6.3.2.4 开始投屏.....	102
4.6.3.2.5 结束投屏.....	102
4.6.3.2.6 远程遥控功能.....	103
4.6.3.2.7 打开/关闭麦克风.....	103
4.6.3.2.8 调节扬声器音量.....	104
4.6.3.2.9 打开/关闭扬声器.....	104
4.6.3.2.10 打开/关闭摄像头.....	105
4.6.3.2.11 离开会议.....	106
4.6.3.2.12 结束会议.....	106
4.6.3.2.13 版本查询.....	106
4.6.3.2.14 打开权限窗口.....	107
4.6.3.2.15 服务端版本查询.....	107
4.6.3.2.16 服务端鼠标样式设置.....	107
4.6.3.3 接口调用结果通知.....	108
4.6.3.3.1 tlsCallback.....	108
4.6.3.3.2 discoverCallback.....	108

4.6.3.3.3 connectCallback.....	108
4.6.3.3.4 disconnectCallback.....	109
4.6.3.3.5 startSharingCallback.....	109
4.6.3.3.6 stopSharingCallback.....	110
4.6.3.3.7 remoteServiceStatusCallback.....	110
4.6.3.3.8 micMuteCallback.....	111
4.6.3.3.9 speakerVolumeCallback.....	111
4.6.3.3.10 speakerMuteCallback.....	112
4.6.3.3.11 cameraMuteCallback.....	112
4.6.3.3.12 leaveConfCallback.....	112
4.6.3.3.13 finishConfCallback.....	113
4.6.3.3.14 sdkLatestVersionCallback.....	113
4.6.3.4 设备推送通知.....	114
4.6.3.4.1 onRemoteServiceStatusChangedNotify.....	114
4.6.3.4.2 onDevConfStatusNotify.....	115
4.6.3.4.3 onShareStatusChangeNotify.....	115
4.6.3.4.4 onServerMouseFeatureNotify.....	116
4.6.3.5 错误通知.....	116
4.6.3.5.1 onErrorNotify.....	117
4.6.4 附录.....	117
4.6.4.1 枚举定义.....	117
4.6.4.1.1 LOG_LEVEL.....	117
4.6.4.1.2 IDEA_SHARE_APP_DEFINAION_QUAITY.....	118
4.6.4.1.3 IDEA_SHARE_APP_AUTH.....	118
4.6.4.1.4 IDEA_SHARE_APP_CONF_STATUS.....	118
4.6.4.1.5 IDEA_SHARE_APP_AUX_STATUS.....	118
4.6.4.1.6 IDEA_SHARE_APP_SERVER_STATUS.....	119
4.6.4.1.7 IDEA_SHARE_APP_ERR.....	119
4.6.4.1.8 IDEA_SHARE_APP_TLS_VERIFY_MODE.....	120
4.6.4.1.9 IDEA_SHARE_MOUSE_SHAPE.....	121
4.6.4.2 OC 类定义.....	121
4.6.4.2.1 IdeaShareServiceAppInfo.....	121
4.6.4.2.2 IdeaShareServiceLogInfo.....	121
4.6.4.2.3 SharePolicy.....	121
4.6.4.2.4 DiscoverResult.....	122
4.6.4.2.5 DeviceInfo.....	122
4.6.4.2.6 RemoteServiceStatus.....	122
4.6.4.2.7 IdeaShareServiceTlsInfo.....	123
4.6.4.3 SDK 接口头文件定义.....	123
4.6.4.3.1 IdeaShareEventHandler.h.....	123
4.6.4.3.2 IdeaShareServiceController.h.....	123
4.6.4.4 公网地址说明.....	123

4.7 错误码参考.....	123
4.7.1 错误码.....	123
<b>5 Windows SDK.....</b>	<b>124</b>
5.1 概述.....	124
5.2 变更记录.....	124
5.3 快速入门.....	124
5.3.1 预置条件.....	124
5.3.2 开发环境准备.....	124
5.3.3 SDK 快速集成.....	125
5.4 典型场景.....	128
5.4.1 典型场景接口调用概览.....	128
5.4.2 场景 1： 初始化.....	130
5.4.3 场景 2： 查询投屏码.....	131
5.4.4 场景 3： 连接/断开连接.....	132
5.4.5 场景 4： 共享/停止投屏.....	133
5.4.6 场景 5： 投放音频.....	134
5.4.7 场景 6： 设置投屏对象.....	134
5.4.8 场景 7： 遥控器.....	135
5.4.9 场景 8： 设备推送.....	136
5.4.10 场景 9： 错误上报.....	137
5.4.11 场景 10： 去初始化.....	137
5.5 常见问题.....	137
5.5.1 获取日志.....	137
5.6 Windows 接口参考（ C++ ） .....	139
5.6.1 概述.....	139
5.6.2 变更记录.....	139
5.6.3 接口参考.....	139
5.6.3.1 基础配置.....	139
5.6.3.1.1 获得 ShareServiceController 单例.....	139
5.6.3.1.2 初始化日志.....	139
5.6.3.1.3 自定义通知回调类.....	140
5.6.3.1.4 初始化 SDK.....	141
5.6.3.1.5 去初始化 SDK.....	141
5.6.3.2 连接设备.....	141
5.6.3.2.1 设备发现.....	141
5.6.3.2.2 TLS 证书校验设置.....	142
5.6.3.2.3 连接.....	143
5.6.3.2.4 断开连接.....	143
5.6.3.3 投屏.....	144
5.6.3.3.1 开始投屏.....	144
5.6.3.3.2 停止投屏.....	144
5.6.3.3.3 设置投屏策略.....	145

5.6.3.3.4 设置投屏对象.....	145
5.6.3.3.5 设置投屏设备类型.....	146
5.6.3.3.6 获取投屏状态.....	147
5.6.3.3.7 设置鼠标样式.....	147
5.6.3.3.8 获取窗口句柄信息.....	148
5.6.3.4 遥控器.....	148
5.6.3.4.1 查询远端设备状态.....	148
5.6.3.4.2 打开/关闭麦克风.....	148
5.6.3.4.3 打开/关闭扬声器.....	149
5.6.3.4.4 打开/关闭摄像头.....	149
5.6.3.4.5 调节扬声器音量.....	150
5.6.3.4.6 离开会议.....	151
5.6.3.4.7 结束会议.....	151
5.6.3.5 版本信息查询.....	152
5.6.3.5.1 获取 SDK 版本信息.....	152
5.6.3.5.2 获取服务端版本信息.....	152
5.6.4 通知参考.....	153
5.6.4.1 接口调用结果通知.....	153
5.6.4.1.1 OnDiscoverResult.....	153
5.6.4.1.2 OnConnectResult.....	154
5.6.4.1.3 OnDisconnectResult.....	154
5.6.4.1.4 OnSharePlayResult.....	154
5.6.4.1.5 OnShareStopResult.....	155
5.6.4.1.6 OnGetRemoteServiceStatusResult.....	155
5.6.4.1.7 OnSetRemoteMicMuteResult.....	156
5.6.4.1.8 OnSetRemoteSpeakerVolumeResult.....	156
5.6.4.1.9 OnSetRemoteSpeakerMuteResult.....	156
5.6.4.1.10 OnSetRemoteCameraMuteResult.....	157
5.6.4.1.11 OnSetLeaveConferenceResult.....	157
5.6.4.1.12 OnSetEndConferenceResult.....	158
5.6.4.1.13 OnGetSDKLatestVersionResult.....	158
5.6.4.1.14 OnSetMouseShapeResult.....	158
5.6.4.2 设备推送通知.....	159
5.6.4.2.1 OnRemoteServiceStatusChangedNotify.....	159
5.6.4.2.2 OnDevConfStateNotify.....	160
5.6.4.2.3 OnShareStatusChangedNotify.....	161
5.6.4.3 错误通知.....	161
5.6.4.3.1 OnError.....	161
5.6.4.3.2 OnDiscoverFailed 参考样例.....	162
5.6.5 附录.....	162
5.6.5.1 枚举定义.....	162
5.6.5.1.1 LogLevel.....	162

5.6.5.1.2 ShareAppDefinationQuality.....	163
5.6.5.1.3 ShareAppScreenType.....	163
5.6.5.1.4 ConfState.....	163
5.6.5.1.5 AuxState.....	164
5.6.5.1.6 ShareAppServerStatus.....	164
5.6.5.1.7 ShareAppErr.....	164
5.6.5.1.8 ShareClientTlsVerifyMode.....	166
5.6.5.1.9 ShareDeviceType.....	166
5.6.5.1.10 IdeaShareStatus.....	166
5.6.5.1.11 ShareMouseShape.....	167
5.6.5.2 结构体定义.....	167
5.6.5.2.1 AppInfo.....	167
5.6.5.2.2 LogInfo.....	167
5.6.5.2.3 SharePolicy.....	167
5.6.5.2.4 DiscoverResult.....	168
5.6.5.2.5 DeviceInfo.....	168
5.6.5.2.6 RemoteServiceStatus.....	168
5.6.5.2.7 ShareTlsInfo.....	169
5.6.5.3 SDK 接口头文件定义.....	169
5.6.5.3.1 IEventHandler.h.....	169
5.6.5.3.2 IShareServiceController.h.....	170
5.6.5.3.3 IdeaShareDef.h.....	172
5.6.5.3.4 IServiceController.h.....	175
5.6.5.4 二进制公网地址说明.....	176
5.7 Windows 接口参考 ( JavaScript ) .....	176
5.7.1 概述.....	176
5.7.2 变更记录.....	176
5.7.3 前言.....	176
5.7.4 使用方法.....	176
5.7.5 接口参考.....	176
5.7.5.1 基础配置.....	176
5.7.5.1.1 初始化日志.....	176
5.7.5.1.2 初始化 SDK.....	177
5.7.5.1.3 去初始化 SDK.....	178
5.7.5.2 连接设备.....	179
5.7.5.2.1 设备发现.....	179
5.7.5.2.2 TLS 证书校验设置.....	180
5.7.5.2.3 连接.....	181
5.7.5.2.4 断开连接.....	182
5.7.5.3 投屏.....	183
5.7.5.3.1 开始投屏.....	183
5.7.5.3.2 停止投屏.....	184

5.7.5.3.3 设置投屏对象.....	184
5.7.5.3.4 设置投屏策略.....	185
5.7.5.3.5 获取投屏状态.....	186
5.7.5.4 遥控器.....	187
5.7.5.5 查询远端设备状态.....	187
5.7.5.6 打开/关闭麦克风.....	188
5.7.5.7 调节扬声器音量.....	189
5.7.5.8 打开/关闭扬声器.....	190
5.7.5.9 打开/关闭摄像头.....	191
5.7.5.10 离开会议.....	191
5.7.5.11 结束会议.....	192
5.7.5.12 版本信息查询.....	193
5.7.5.12.1 获取 SDK 版本信息.....	193
5.7.5.12.2 获取大屏侧服务端版本信息.....	194
5.7.5.13 事件上报/接口回调结果.....	195
5.7.5.13.1 投屏状态发生变化事件上报.....	195
5.7.5.14 大屏状态信息变化通知事件.....	195
5.7.5.15 设备推送辅流和会议状态通知.....	196
5.7.5.16 接口调用错误回调.....	197
5.7.5.16.1 OnError 错误事件上报.....	197
5.7.6 附录.....	197
5.7.6.1 错误码一对应接口表.....	197
5.7.6.2 结构体描述.....	199
5.7.6.2.1 LogInfo 结构体.....	199
5.7.6.2.2 ShareTlsInfo 结构体.....	199
5.7.6.2.3 SharePolicy 结构体.....	199
5.7.6.2.4 LogLevel 结构体.....	199
5.7.6.2.5 ShareClientTlsVerifyMode 结构体.....	200
5.7.6.2.6 ShareAppScreenType 结构体.....	200
5.7.6.2.7 IdeaShareStatus 结构体.....	200
5.8 错误码参考.....	200
5.8.1 错误码.....	201
<b>6 个人数据处理说明.....</b>	<b>202</b>

# 1 SDK 下载

表 1-1

平台	资源	SDK包大小	下载链接	发布日期	说明
Android	SDK	6.16 MB	下载IdeaHub Engine Suite_21.1.0_IdeaShareSDK_Android.zip 指导文档: <a href="#">2 Android SDK</a>	2021-08-27	适用于IdeaHub Enterprise/Pro/S/B2/、Borad Pro/EDU、Board2、IdeaHub S2/S2 Pro/ES2/ES2 Pro
IOS	SDK	23.10MB	下载IdeaHub Engine Suite_21.1.0_IdeaShareSDK_iOS.zip 指导文档: <a href="#">3 IOS SDK</a>	2021-08-27	适用于IdeaHub Enterprise/Pro/S/B2/、Borad Pro/EDU、Board2、IdeaHub S2/S2 Pro/ES2/ES2 Pro
MAC	SDK	3.32 MB	下载IdeaHub Engine Suite_21.1.0_IdeaShareSDK_macOS.zip 指导文档: <a href="#">4 MAC SDK</a>	2021-08-27	适用于IdeaHub Enterprise/Pro/S/B2/、Borad Pro/EDU、Board2、IdeaHub S2/S2 Pro/ES2/ES2 Pro
Windows	SDK	3.42 MB	下载IdeaHub Engine Suite_21.1.0_IdeaShareSDK_Windows.zip 指导文档: <a href="#">5 Windows SDK</a>	2021-08-27	适用于IdeaHub Enterprise/Pro/S/B2/、Borad Pro/EDU、Board2、IdeaHub S2/S2 Pro/ES2/ES2 Pro

# 2 Android SDK

## 2.1 概述

投屏二次开发Android SDK 提供了一套完整的接口集合，开发者可以通过调用 ideaShare SDK ( 以下简称SDK ) 打开的API，快速集成投屏能力，包括启动投屏、遥控器等。

## 2.2 变更记录

表 2-1 变更记录

日期	版本	变更内容
2021-02-09	60.13.2	初稿完成

## 2.3 快速入门

本文面向有一定Java和Android开发能力的开发者，Demo编译流程为例，介绍如何使用SDK 进行二次开发。

### 2.3.1 预置条件

支持SMC 2.0 , SMC 3.0, 云会议组网。

### 2.3.2 开发环境准备

表 2-2 环境要求

名称	要求
Java版本	Java8
Android Studio	推荐使用版本 4.0+

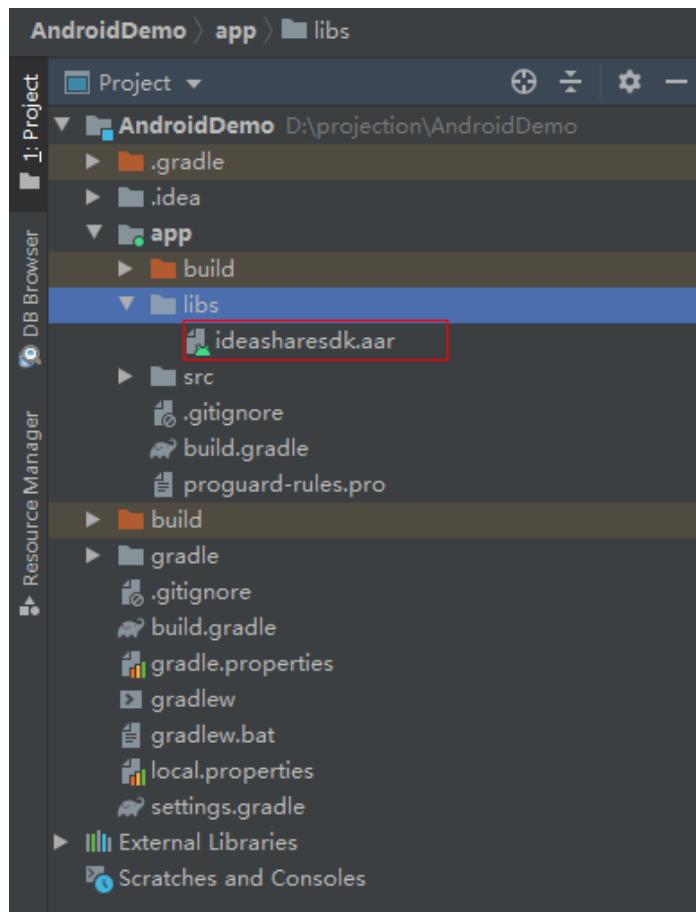
名称	要求
Android 最低运行环境	安卓7
CPU	ARM32 ARM64
CompileSdkVersion	29
TargetSdkVersion	28

### 2.3.3 SDK 快速集成

#### 1. 在工程中导入AAR包

安卓SDK是以aar包的形式提供集成使用的，首先需要将ideasharesdk.aar文件导入到项目工程当中，放在目录app/libs目录下。如图1-1所示：

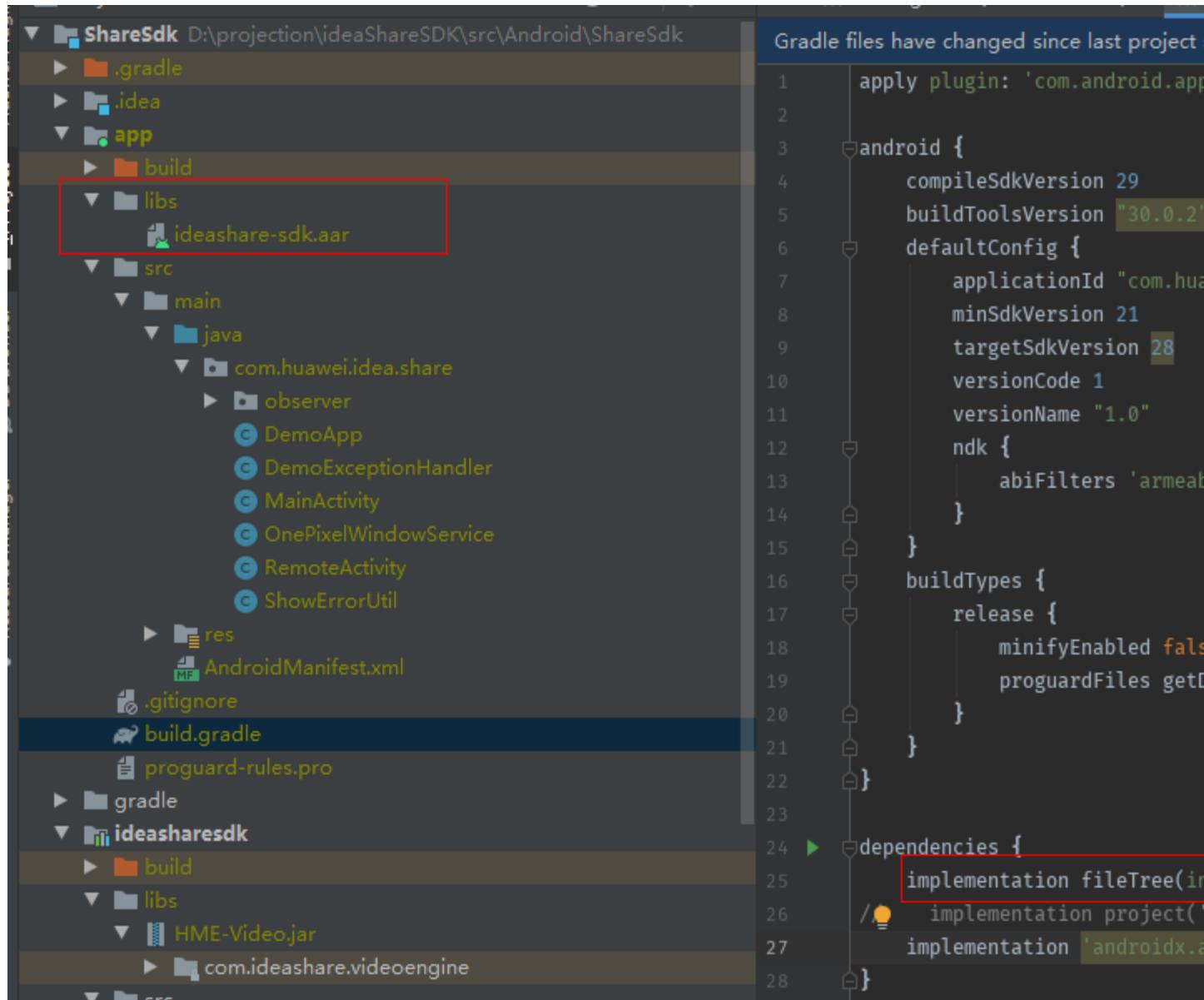
图 2-1 文件目录



#### 2. 修改项目配置文件 build.gradle

在build.gradle文件中引用sdk，然后点击右上角大象图标，同步gradle配置。如图1-2所示：

图 2-2 导入 SDK



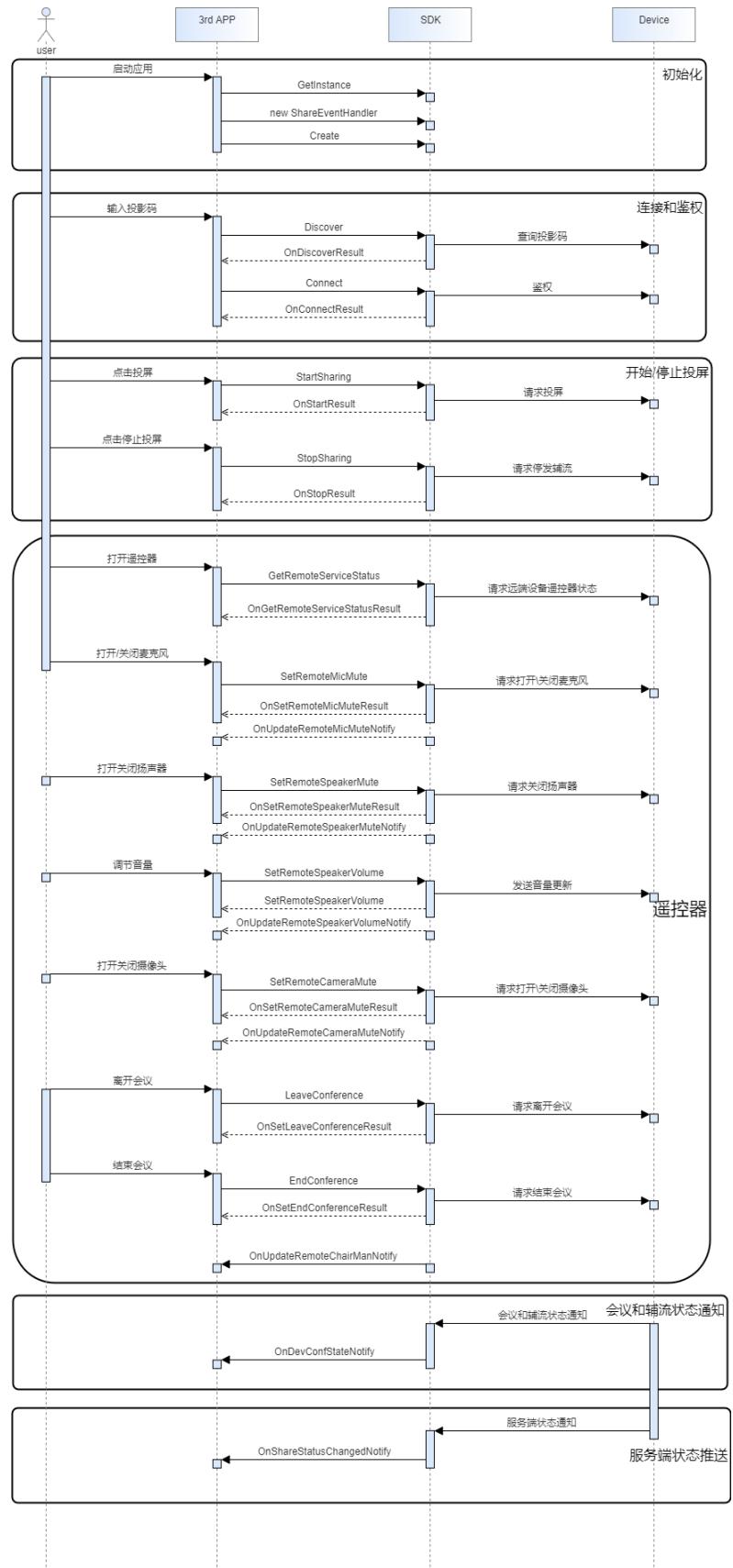
### 3.项目中使用API

完成以上两部之后就可以正常使用SDK 的API了（请参考Android SDK API文档）。

图 2-3 导入依赖项，使用 API

```
import com.huawei.idea.ideasharesdk.sdk.IShareServiceController;
import com.huawei.idea.ideasharesdk.sdk.ShareServiceController;
```

图 2-4 接口调用流程



## 2.4 典型场景

### 2.4.1 初始化

#### 描述

投屏SDK的所有接口都定义在 IShareServiceController 接口中，需要通过实例对象 ShareServiceController 对象来使用，第一步应该创建该对象：调用该对象的静态方法 create 获取实例。

#### 注意事项

该方法将返回一个单列对象，只需执行一次。

#### 示例代码

```
IShareServiceController shareServiceController = ShareServiceController.create (this, applInfo, new  
IShareServiceController.IShareEventHandler() {  
    @Override  
    public void onRemoteServiceStatusChangedNotify(RemoteServiceStatus remoteServiceStatus)  
{  
    }  
    @Override  
    public void onShareStatusChangedNotify(ShareStatus shareStatus, int i) {  
    }  
});
```

### 2.4.2 查询投影码

#### 描述

用户输入投影码后，返回连接设备IP和密码。

#### 业务流程

1. 获取EUA地址和端口号，EUA地址为空则传入空字符串，端口号0。
2. 调用 ShareServiceController 的 Discover 接口。
3. 处理回调  
    处理回调中的结果。

#### 注意事项

初始化SDK之后才能调用查询投影码。

#### 示例代码

```
shareServiceController.discover("1234567", "", 144, new Callback<DiscoverResult>() {  
    @Override  
    public void onSuccess(DiscoverResult discoverResult) {  
        //进行connect  
    }  
    @Override  
    public void onFailed(int retCode, String desc) {  
    }  
});
```

```
        }  
    }  
}
```

## 2.4.3 连接/断开连接

### 描述

连接设备（例如IdeaHub）

### 业务流程

**连接接口调用：**

1. 调用Connect接口；
2. 处理连接回调结果。

**断开接口调用：**

1. 调用Disconnect接口；
2. 处理断开连接回调结果。

### 注意事项

必须在调用初始化和查询投影码后才能调用此接口。

### 示例代码

```
shareServiceController.connect(connectInfo, new Callback<ConnectResult>() {  
    @Override  
    public void onSuccess(ConnectResult connectResult) {  
        if(connectResult.getDevName() != null){  
            app.setDeviceName(connectResult.getDevName());  
        }  
    }  
    @Override  
    public void onFailed(int i, String s) {  
    }  
});  
  
shareServiceController.disconnect(new Callback<Void>() {  
    @Override  
    public void onSuccess(Void aVoid) {  
        LogUtil.d(TAG, "Disconnect success!!!!");  
    }  
    @Override  
    public void onFailed(int retCode, String desc) {  
        ErrorHandler.showErrorMsg(me, retCode);  
    }  
});
```

## 2.4.4 共享/停止共享

### 描述

连接设备后，调用共享接口发起共享或调用停止共享接口停止共享。

## 业务流程

1. 投屏前需首先获得用户授权屏幕截取权限，调用 confirmPermissionBeforeSharing 接口。
2. 在当前 Activity 中重写 onActivityResult 方法，在此调用投屏接口 startSharing。

## 注意事项

连接设备之后才能调用这两个接口。

## 示例代码

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // 确认权限后开始投屏
    shareServiceController.startSharing(requestCode, resultCode, data, new Callback<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
        }
        @Override
        public void onFailure(int i, String s) {
        }
    });
}

shareServiceController.stopSharing(new Callback<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
    }
    @Override
    public void onFailure(int i, String s) {
    }
});
```

## 2.4.5 遥控器

### 描述

在连接设备后，客户端可以查询设备状态，包括：麦克风状态、扬声器状态、摄像头状态、音量大小、是否在会议中，并且操作这些设备状态，如调节音量大小、离开会议（前提是已经入会）。

## 业务流程

1. 调用getRemoteServiceStatus查询连接设备状态。
2. 处理回调结果事件OnGetRemoteServiceStatusResult。
3. 调用遥控器接口（如setRemoteMicMute）请求遥控远端设备。
4. 收到服务端主动推送过来的通知事件OnRemoteServiceStatusChangedNotify 事件，更新客户端遥控器状态。
5. 在会议中时，调用leaveConference 离开会议，处理回调事件结果。

## 注意事项

连接设备后遥控器功能才可用。

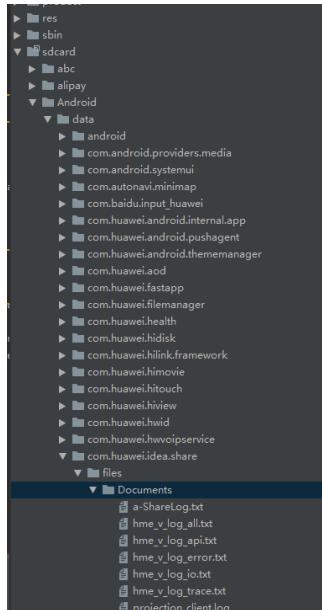
## 示例代码

```
shareServiceController.getRemoteServiceStatus(new Callback<RemoteServiceStatus>() {  
    @Override  
    public void onSuccess(RemoteServiceStatus remoteServiceStatus) {  
    }  
    @Override  
    public void onFailure(int i, String s) {  
    }  
});
```

## 2.5 常见问题

### 2.5.1 获取日志

1. 可以使用 setLog 接口指定日志路径，如果没有指定，默认日志路径将放在/sdcard/Android/data/‘应用名’/files/Documents 下：



### 2.5.2 混淆文件

1. 使用apk release发布时，如使用混淆文件，需在里面添加如下内容：  
-keep class com.ideashare.videoengine.\*\* {\*;}  
-keep class com.huawei.idea.ideasharesdk.\*\*{\*;}

## 2.6 Android 接口参考

### 2.6.1 概述

本文主要介绍投屏服务 Java 接口参考。主要包含文档修订记录、接口函数定义、参数定义说明、告警类型附录、事件上报通知说明以及错误码等信息。

## 2.6.2 变更记录

日期	版本	变更内容
2021-02-02	1.0.0	1. 初稿-已实现接口参考
2021-02-09	1.0.0.1	1. 接口优化

## 2.6.3 接口参考

### 2.6.3.1 基础配置

#### 2.6.3.1.1 初始化

##### 接口描述

该接口用于获取ShareServiceController 单例对象。

##### 注意事项

需在create方法调用之后使用才能获得对象，否则返回null。

##### 方法定义

```
/**  
 * 获得接口单例对象  
 * @param context 应用上下文  
 * @param applInfo app相关信息  
 * @param iShareEventHandler 上报事件通知函数  
 */  
public static ShareServiceController getInstance();
```

##### 参数描述

无。

##### 示例代码

```
ShareServiceController instance = ShareServiceController.getInstance();
```

##### 接口描述

该接口用于初始化SDK，进行一些环境准备工作，调用其他接口之前必须先调用这个接口，只需要调用一次，建议放在Application的onCreate里。

##### 注意事项

属于ShareServiceController类的静态方法，是调用SDK其它接口的前置条件。

##### 方法定义

```
/**  
 * 初始化接口  
 * @param context 应用上下文  
 * @param applInfo app相关信息  
 * @param iShareEventHandler 上报事件通知函数  
 */  
public static IShareServiceController create(Context context, ApplInfo applInfo, IShareEventHandler iShareEventHandler);
```

## 参数描述

参数	是否必须	类型	描述
context	是	Context	应用上下文，建议使用getApplicationContext()
applInfo	否	AppInfo	app相关信息
	appId	String	appId 建议取值企业域名+应用名 长度<=128
	exePath	String	应用执行路径 长度<=128
	logInfo	LogInfo	日志配置（可选，null则使用默认日志路径 /sdcard/Android/data/{appName}/files/Documents/）
iShareEventHandler	是	IShareEventHandler	注册上报事件通知函数类

## 示例代码

```
LogInfo logInfo = new LogInfo(path, LogLevel.DEBUG, true);
AppInfo applInfo = new AppInfo("appId", "log");
applInfo.setLogInfo(logInfo);
// 配置默认日志
IShareServiceController shareServiceController = ShareServiceController.create(this, applInfo, new
IShareServiceController.IShareEventHandler() {
    @Override
    public void onRemoteServiceStatusChangedNotify(RemoteServiceStatus remoteServiceStatus) {
    }
    @Override
    public void onShareStatusChangedNotify(ShareStatus shareStatus, int i) {
    }
});
```

### 说明

返回值IShareServiceController为SDK对外提供的所有方法的interface。LogInfo参数建议总是传入，以便查看初始化日志。

## 2.6.3.1.2 日志设置

### 接口描述

日志设置接口。

### 注意事项

如果需要自定义日志路径，在调用create方法之后尽可能早的调用该方法，以便查看完整日志。

### 方法定义

```
/*
 * 日志设置接口
 * @param logInfo 日志信息类
 * @return 0表示设置成功
 */
```

```
int setLog(LogInfo logInfo)
```

#### 参数描述

参数	是否必须	类型	描述
logInfo	是	LogInfo	日志信息
path	是	String	日志路径，长度225
logLevel	是	LogLevel	日志级别： ERROR、WARNING、 INFO、DEBUG
enable	是	boolean	是否开启日志

#### 示例代码

```
String path = getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS).getAbsolutePath();
LogInfo logInfo = new LogInfo(path, LogLevel.DEBUG, true);
shareServiceController.setLog(logInfo);
```

#### 说明

同步接口，返回值0标识成功，-1标识失败；初始化日志。

### 2.6.3.2 业务接口（主动调用）

#### 2.6.3.2.1 设置设备类型

##### 接口描述

该接口仅用于对投屏器进行开发时设置设备类型。

##### 注意事项

- 普通android设备无需调用本接口，仅适用于投屏器。
- 初始化后调用。

##### 方法定义

```
/** * 设置设备类型 * @param deviceType 设备类型 * @return 0表示设置成功 */
int setDeviceType(DeviceType deviceType);
```

#### 参数描述

参数	是否必须	类型	描述
deviceType	是	DeviceType	设备类型： PROJECTOR //投屏器 NON_PROJECTOR //非投屏器

## 示例代码

```
ShareServiceController.setDeviceType(DeviceType.PROJECTOR);
```

### 2.6.3.2.2 设备发现

#### 接口描述

该接口用于设备发现（投屏码解析）。

#### 注意事项

- 初始化后调用。
- 投屏码分为在线模式和离线解析两种场景，请根据实际环境选择。
- 请确保大屏与终端在同一网络环境下(网络互通)。
- 投屏码为6/8位数字或者字母，不支持组合形式。
- 设备发现和设备连接接口一般需组合使用。

#### 方法定义

```
/**  
 * 设备发现（投屏码解析）  
 * @param castCode 投屏码  
 * @param address 请求服务器的地址  
 * @param port 请求服务器的端口号  
 * @param callback 回调函数  
 * @return 0表示接口调用成功  
 */  
int discover(String castCode, String address, int port, Callback<DiscoverResult> callback);
```

#### 参数描述

参数	是否必须	类型	描述
castCode	是	String	投影码，以大屏显示为准
address	否	String	Eua服务IP，本地解析请填”空字符串”
port	否	int	在线模式下EUA 服务器端口，离线模式下可填入任意数字
callback	是	Callback	回调函数，异步回调结果

#### DiscoverResult参数说明

参数	是否必须	类型	描述
addressNum	/	int	设备查询数量
password	/	String	连接码
addressArray	/	String[]	查询到的IdeaHub地址集合，多个

## 示例代码

```
shareServiceController.discover("01234567", "", 144, new Callback<DiscoverResult>() {  
    @Override  
    public void onSuccess(DiscoverResult discoverResult) {  
        //进行connect  
    }  
    @Override  
    public void onFailure(int retCode, String desc) {  
    }  
});  
}
```

### 2.6.3.2.3 连接

#### 接口描述

该接口用于设备发现后进行连接设备。

#### 注意事项

连接一般需与设备发现组合使用，此时大多数入参来源于设备发现的返回值。

#### 方法定义

```
/**  
 * 连接  
 * @param connectInfo 连接信息  
 * @param callback 回调函数  
 * @return 0表示接口调用成功  
 */  
int connect(ConnectInfo connectInfo, Callback<ConnectResult> callback);
```

#### 参数描述

参数	是否必须	类型	描述
connectInfo	是	ConnectInfo	连接信息
password	是	String	连接码，来自DiscoverResult
addressNum	是	int	设备查询数量，来自DiscoverResult
addressArray	是	String[]	查询到的IdeaHub地址集合，多个，来自DiscoverResult
port	否	int	保留字段，可不填
callback	是	Callback	回调函数，异步回调结果

#### ConnectResult参数说明

参数	是否必须	类型	描述
devAddr	/	String	连接智慧屏的IP

参数	是否必须	类型	描述
devName	/	String	智慧屏的名字

### 示例代码

```
shareServiceController.connect(connectInfo, new Callback<ConnectResult>() {
    @Override
    public void onSuccess(ConnectResult connectResult) {
        if(connectResult.getDevName() != null){
            app.setDeviceName(connectResult.getDevName());
        }
    }
    @Override
    public void onFailed(int i, String s) {
    }
});
```

### 2.6.3.2.4 断开连接

#### 接口描述

该接口用于断开与智慧屏的连接。

#### 注意事项

无。

#### 方法定义

```
/**
 * 断开连接
 * @param callback 回调函数
 * @return 0表示接口调用成功
 */
int disconnect(Callback<Void> callback)
```

#### 参数描述

参数	是否必须	类型	描述
callback	是	Callback	回调函数，异步回调结果

### 示例代码

```
shareServiceController.disconnect(new Callback<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        LogUtil.d(TAG, "Disconnect success!!!!");
    }
    @Override
    public void onFailed(int retCode, String desc) {
        ErrorHandler.showErrMsg(me, retCode);
    }
});
```

### 2.6.3.2.5 投屏前权限确认

#### 接口描述

该接口用于获取投屏所需权限。

### 注意事项

与用户有交互，建议通过重写activity的onActivityResult获取交互结果。

### 方法定义

```
/**  
 * 投屏前权限确认  
 * @param activity 用于接收onActivityResult的页面对象  
 * @return 0表示接口调用成功  
 */  
int confirmPermissionBeforeSharing(Activity activity)
```

### 参数描述

参数	是否必须	类型	描述
activity	是	Activity	接收用户操作结果的页面

### 示例代码

```
shareServiceController.confirmPermissionBeforeSharing(this)
```

## 2.6.3.2.6 开始投屏

### 接口描述

该接口用于开始投屏。

### 注意事项

需与投屏前权限确认接口组合使用，将用户操作的结果（onActivityResult中）作为入参传入。

需要调用1.3.2.20开启兼容性。

### 方法定义

```
/**  
 * 开始投屏  
 * @param requestCode 权限确认onActivityResult请求码  
 * @param resultCode 权限确认onActivityResult结果码  
 * @param intent 权限确认onActivityResult的传递信息  
 * @param callback 回调函数  
 * @return 0表示接口调用成功  
 */  
int startSharing(int requestCode, int resultCode, Intent intent, Callback<Void> callback)
```

### 参数描述

参数	是否必须	类型	描述
callback	是	Callback	回调函数，异步回调结果
requestCode	是	int	权限确认onActivityResult请求码

参数	是否必须	类型	描述
resultCode	是	int	权限确认onActivityResult结果码 -1为成功，其他为失败
intent	是	Intent	权限确认onActivityResult的传递信息

### 示例代码

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // 确认权限后开始投屏
    shareServiceController.startSharing(requestCode, resultCode, data, new Callback<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
        }
        @Override
        public void onFailure(int i, String s) {
        }
    });
}
```

### 2.6.3.2.7 停止投屏

#### 接口描述

该接口用于停止投屏。

#### 注意事项

无。

#### 方法定义

```
/**
 * 停止投屏
 * @param callback 回调函数
 * @return 0表示接口调用成功
 */
int stopSharing(Callback<Void> callback);
```

#### 参数描述

参数	是否必须	类型	描述
callback	是	Callback	回调函数，异步回调结果

### 示例代码

```
shareServiceController.stopSharing(new Callback<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
    }
    @Override
    public void onFailure(int i, String s) {
    }
});
```

### 2.6.3.2.8 获取投屏状态

#### 接口描述

该接口用于获取投屏状态。

#### 注意事项

无。

#### 方法定义

```
/**  
 * 获得投屏状态  
 * @return 投屏状态  
 */  
ShareStatus getShareStatus();
```

#### 参数描述

无

#### 示例代码

```
shareServiceController.getShareStatus();
```

### 2.6.3.2.9 获取 SDK 版本

#### 接口描述

该接口可以获取客户端SDK版本。

#### 注意事项

无。

#### 方法定义

```
/**  
 * 获得SDK版本  
 * @return 版本号  
 */  
String getSDKVersion();
```

#### 参数描述

无。

#### 返回值

版本号

#### 示例代码

```
shareServiceController.getSDKVersion();
```

### 2.6.3.2.10 获取智慧屏最新版本信息

#### 接口描述

该接口用于获取服务端SDK最新版本。

#### 注意事项

1. 没有连接不应调用此接口。

## 方法定义

```
/**  
 * 获取SDK最新版本  
 * @param callback 回调函数  
 * @return 0表示接口调用成功  
 */  
int getSdkLatestVersion(Callback<String> callback);
```

## 参数描述

参数	是否必须	类型	描述
callback	是	Callback	回调函数，异步回调结果

## 示例代码

```
shareServiceController.getSdkLatestVersion(new Callback<String>() {  
    @Override  
    public void onSuccess(String s) {  
        Message msg = new Message();  
        msg.what = SDK_VERSION_SIGNAL;  
        msg.obj = s;  
        handler.sendMessage(msg);  
    }  
  
    @Override  
    public void onFailed(int retCode, String s) {  
        sendErrorSignal(retCode);  
    }  
});
```

### 2.6.3.2.11 获取远程服务状态信息

#### 接口描述

该接口用于获取远程服务状态信息。

#### 注意事项

- 需连接成功。

## 方法定义

```
/**  
 * 获得远程服务状态信息  
 * @param callback 回调函数  
 * @return 0表示接口调用成功  
 */  
int getRemoteServiceStatus(Callback<RemoteServiceStatus> callback);
```

## 参数描述

参数	是否必须	类型	描述
callback	是	Callback	回调函数，异步回调结果

#### RemoteServiceStatus参数说明

参数	是否必须	类型	描述
isMute	/	boolean	麦克风状态: TRUE关闭 FALSE打开
isCameraMute	/	boolean	摄像头状态: TRUE关闭 FALSE打开
isSpeakerMute	/	boolean	扬声器状态: TRUE关闭 FALSE打开
isChairman	/	boolean	是否主席: TRUE主席模式 FALSE非主席模式
isBase	/	boolean	是否IdeaHub标准版 (基本类型)
volume	/	int	音量大小
callStatus	/	CallStatus	IDLE, //空闲 DAILING, //正在拨号 CALLING //正在通话中

## 示例代码

```
shareServiceController.getRemoteServiceStatus(new Callback<RemoteServiceStatus>() {  
    @Override  
    public void onSuccess(RemoteServiceStatus remoteServiceStatus) {  
    }  
    @Override  
    public void onFailure(int i, String s) {  
    }  
});
```

### 2.6.3.2.12 打开/关闭麦克风

#### 接口描述

该接口用于打开和关闭麦克风。

#### 注意事项

1. 连接后进行操作。

#### 方法定义

```
/**  
 * 打开\关闭麦克风  
 * @param isMute  
 * @param callback 回调函数  
 * @return 0表示接口调用成功  
 */  
int setRemoteMicMute(boolean isMute, Callback<Void> callback);
```

#### 参数描述

参数	是否必须	类型	描述
isMute	是	boolean	TRUE关闭 FALSE打开
callback	是	Callback	回调函数，异步回调结果

### 示例代码

```
shareServiceController.setRemoteMicMute(false, new Callback<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        LogUtil.d(TAG, "setRemoteMicUNMute success!!!!");
    }
    @Override
    public void onFailure(int retCode, String desc) {
        ErrorHandler.showErrorMsg(me, retCode);
    }
});
```

### 2.6.3.2.13 调节扬声器音量

#### 接口描述

该接口用于远程调节扬声器音量。

#### 注意事项

- 连接后执行；

#### 方法定义

```
/** 
 * 调节扬声器音量
 * @param volume 音量
 * @param callback 回调函数
 * @return 0表示接口调用成功
 */
int setRemoteSpeakerVolume(int volume, Callback<Void> callback);
```

#### 参数描述

参数	是否必须	类型	描述
volume	是	int	扬声器音量[0~21]
callback	是	Callback	回调函数，异步回调结果

### 示例代码

```
shareServiceController.setRemoteSpeakerVolume(seekBar.getProgress(), new Callback<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        LogUtil.d(TAG, "Volume set Success!!!!");
    }
    @Override
    public void onFailure(int retCode, String desc) {
        ErrorHandler.showErrorMsg(me, retCode);
    }
});
```

### 2.6.3.2.14 打开/关闭扬声器

#### 接口描述

该接口用于远程打开、关闭扬声器。

#### 注意事项

1. 连接后执行；

#### 方法定义

```
/*
 * 打开/关闭扬声器
 * @param isSpeakerMute TURE 关闭 FALSE 打开
 * @param callback 回调函数
 * @return 0表示接口调用成功
 */
int setRemoteSpeakerMute(boolean isSpeakerMute, Callback<Void> callback);
```

#### 参数描述

参数	是否必须	类型	描述
isSpeakerMute	是	boolean	TURE 关闭 FALSE 打开
callback	是	Callback	回调函数，异步回调结果

#### 示例代码

```
shareServiceController.setRemoteSpeakerMute(false, new Callback<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        LogUtil.d(TAG, "Set Volume unmute success!!!!");
    }
    @Override
    public void onFailure(int i, String s) {
        ErrorHandler.showErrMsg(me, i);
    }
});
```

### 2.6.3.2.15 打开/关闭摄像机

#### 接口描述

该接口用于打开和关闭摄像机。

#### 注意事项

1. 连接后进行操作。

#### 方法定义

```
/*
 * 打开\关闭摄像机
 * @param isCameraMute TURE 关闭 FALSE 打开
 * @param callback 回调函数
 * @return 0表示接口调用成功
 */
int setRemoteCameraMute(boolean isCameraMute, Callback<Void> callback);
```

#### 参数描述

参数	是否必须	类型	描述
isCameraMute	是	boolean	true关闭 false打开
callback	是	Callback	回调函数，异步回调结果

### 示例代码

```
shareServiceController.setRemoteCameraMute(false, new Callback<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        LogUtil.d(TAG, "setRemoteCameraMute false!!!!!");
    }
    @Override
    public void onFailure(int retCode, String desc) {
        ErrorHandler.showErrorMsg(me, retCode);
    }
});
```

## 2.6.3.2.16 离开会议

### 接口描述

该接口用于远程离开会议。

### 注意事项

1. 连接后进行操作。
2. 主席权限才可以结束会议。

### 方法定义

```
/**
 * 离开会议
 * @param isEndConference 离开时是否结束会议
 * @param callback 回调函数
 * @return 0表示接口调用成功
 */
int leaveConference(boolean isEndConference, Callback<Void> callback);
```

### 参数描述

参数	是否必须	类型	描述
isEndConference	是	boolean	离开时是否结束会议（主席可填true）
callback	是	Callback	回调函数，异步回调结果

### 示例代码

```
shareServiceController.leaveConference(false, new Callback<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        LogUtil.d(TAG, "Leave Meeting Success!!!!!");
    }
});
```

```
    @Override
    public void onFailed(int i, String s) {
        ErrorHandler.showErrorMsg(me, i);
    }
});
```

### 2.6.3.2.17 使用证书鉴权连接

#### 接口描述

该接口仅用于设置TLS连接证书。

#### 注意事项

- 需使用服务端支持的签名证书

#### 方法定义

```
/** 
 * 设置连接证书
 * @param tlsInfo 设备类型
 * @param callback 设备类型
 * @return 0表示设置成功
 */
int setTls(TlsInfo tlsInfo, Callback<Boolean> callback);
```

#### 参数描述

参数	是否必须	类型	描述
tlsInfo	是	TlsInfo	证书信息
verifyMode	是	VerifyMode	0不认证 1服务端认证
	是	String	tcp 建连证书路径
	是	String	http 建连证书路径
callback	是	Callback	回调函数，异步回调结果

#### 示例代码

```
shareServiceController.setTls(tlsInfo, new Callback<Boolean>() {
    @Override
    public void onSuccess(Boolean res) {
    }
    @Override
    public void onFailed(int i, String s) {
    }
});
```

### 2.6.3.2.18 开启/关闭 音频投放

#### 接口描述

该接口用于开启/关闭音频投放功能。

#### 注意事项

- 仅支持安卓10以上设备和target sdk 29及以上的应用。
- 音频采集需要用户授权

### 方法定义

```
/**  
 * 开启或关闭音频投放  
 * @param activity 任意页面的activity对象  
 * @param enabled 开启、关闭  
 * @return 0表示设置成功  
 */  
int enableAudio(Activity activity, boolean enabled);
```

### 参数描述

参数	是否必须	类型	描述
activity	是	Activity	当前页面引用
enabled	是	boolean	true开启 false关闭

### 示例代码

```
shareServiceController.enableAudio(RemoteActivity.this, true);
```

## 2.6.3.2.19 自定义通知

### 接口描述

该接口用于自定义前台通知

### 注意事项

- 使用自定义通知将彻底替换默认通知，需要对应用进行充分测试，否则可能影响到前台服务的拉起。

### 方法定义

```
/**  
 * 设置自定义通知  
 * @param customNotification 自定义通知对象  
 * @param notificationId 通知id  
 */  
void setCustomNotification(Notification customNotification, int notificationId);
```

### 参数描述

参数	是否必须	类型	描述
customNotification	是	Notification	自定义通知对象
notificationId	是	int	自定义通知id

### 示例代码

```
shareServiceController.setCustomNotification(createNotification(), GlobalConstant.NOTIFICATION_ID);  
private Notification createNotification() {
```

```
// Android8.0以上的系统，新建消息通道
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    // 用户可见的通道名称
    String channelName = getString(R.string.air_presence_app_name);
    // 通道的重要程度
    int importance = NotificationManager.IMPORTANCE_HIGH;
    NotificationChannel notificationChannel =
        new NotificationChannel(GlobalConstant.CHANNEL_ID, channelName, importance);
    NotificationManager notificationManager = getSystemService(NotificationManager.class);
    if (notificationManager != null) {
        notificationManager.createNotificationChannel(notificationChannel);
    }
}

NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
GlobalConstant.CHANNEL_ID);
// 通知小图标
builder.setSmallIcon(R.drawable.ic_launcher);
// 通知标题
builder.setContentTitle(getString(R.string.ideashare_return_application));
// 通知时间
builder.setWhen(System.currentTimeMillis());
// 设定启动的内容
Intent activityIntent = new Intent(this, ProjectionActivity.class);
PendingIntent pendingIntent =
    PendingIntent.getActivity(this,
        GlobalConstant.JUMP_BACK_CODE, activityIntent,
PendingIntent.FLAG_UPDATE_CURRENT);
builder.setContentIntent(pendingIntent);
// 通知按钮
Intent stopShareIntent = new Intent(this, NotificationReceiver.class);
stopShareIntent.setAction(NotificationReceiver.ACTION_STOP_PROJECTION);
PendingIntent broadcastIntent =
    PendingIntent.getBroadcast(this,
        GlobalConstant.STOP_SHARE_CODE, stopShareIntent,
PendingIntent.FLAG_UPDATE_CURRENT);
builder.addAction(GlobalConstant.NO_ICON, getString(R.string.air_presence_stop_share),
broadcastIntent);
// 创建通知并返回
return builder.build();
}
```

### 2.6.3.2.20 开启兼容性

#### 接口描述

该接口会设置是否开启兼容性，用于兼容现网CBC加密模式。注意：当前现网均是CBC加密模式，此接口必须设置成开启兼容性。

#### 注意事项

- 投屏前必须调用此接口，设置成开启状态，否者投屏会失败。

#### 方法定义

```
/**  
 * 设置自定义通知  
 * @param customNotification 自定义通知对象  
 * @param notificationId 通知id  
 */  
int setCompatibility(boolean enabled);
```

#### 参数描述

参数	是否必须	类型	描述
enabled	是	boolean	开启兼容性 0不开启 1开启

### 示例代码

```
shareServiceController.enableAudio(RemoteActivity.this, true);
```

## 2.6.3.3 业务接口（全局推送 IShareEventHandler 方法）

### 2.6.3.3.1 远程服务状态信息变化通知

#### 接口描述

该接口用于接收远程服务状态信息变化通知。

#### 注意事项

1. 用于同步大屏侧的状态变化
2. 随SDK初始化时进行方法重写注册

#### 方法定义

```
/**  
 * 远程服务状态信息变化通知  
 * @param remoteServiceStatus 远程状态信息  
 */  
void onRemoteServiceStatusChangedNotify(RemoteServiceStatus remoteServiceStatus);
```

#### 参数描述

RemoteServiceStatus 参数说明

参数	是否必须	类型	描述
isMute	/	boolean	麦克风状态: TRUE 关闭 FALSE 打开
isCameraMute	/	boolean	摄像头状态: TRUE 关闭 FALSE 打开
isSpeakerMute	/	boolean	扬声器状态: TRUE 关闭 FALSE 打开
isChirman	/	boolean	是否主席
isBase	/	boolean	是否IdeaHub标准版 (基本类型)
volume	/	int	音量大小
callStatus	/	CallStatus	IDLE, //空闲 DAILING, //正在拨号 CALLING //正在通话中

## 示例代码

```
shareServiceController =  
    ShareServiceController.getInstance().create(this, applInfo, new  
IShareServiceController.IShareEventHandler() {  
    @Override  
    public void onRemoteServiceStatusChangedNotify(RemoteServiceStatus remoteServiceStatus) {  
        Log.d(TAG, "onRemoteServiceStatusChangedNotify: " + remoteServiceStatus);  
        app.setRemoteServiceStatus(remoteServiceStatus);  
        app.notifyRemoteServiceStatusChanged(remoteServiceStatus);  
    }  
    @Override  
    public void onShareStatusChangedNotify(ShareStatus shareStatus, int i) {  
        Log.d(TAG, "onShareStatusChangedNotify: " + shareStatus);  
        app.setShareStatus(shareStatus);  
        app.notifyShareStatusChanged(shareStatus);  
    }  
});  
LogUtil.d(TAG, "Create SDK object Success !!!");  
// 存入全局  
app.setShareServiceController(shareServiceController);  
}
```

### 2.6.3.3.2 共享状态变化通知

#### 接口描述

该接口用于接收大屏共享状态变化通知。

#### 注意事项

1. 用于同步大屏侧的状态变化
2. 随SDK初始化时进行方法重写注册

#### 方法定义

```
/**  
 * 状态变化通知  
 * @param status 连接状态  
 * @param reason 原因值  
 */  
void onShareStatusChangedNotify(ShareStatus status, int reason);
```

#### 参数描述

参数	是否必须	类型	描述
status	是	ShareStatus	CONNECTED, //连接 SHARING, //共享中 STOPSHARING, //停止共享 DISCONNECT //断连
reason	是	int	原因值

## 示例代码

```
shareServiceController =  
    ShareServiceController.getInstance().create(this, applInfo, new  
IShareServiceController.IShareEventHandler() {  
    @Override  
    public void onRemoteServiceStatusChangedNotify(RemoteServiceStatus remoteServiceStatus) {
```

```
        Log.d(TAG, "onRemoteServiceStatusChangedNotify: " + remoteServiceStatus);
        app.setRemoteServiceStatus(remoteServiceStatus);
        app.notifyRemoteServiceStatusChanged(remoteServiceStatus);
    }
    @Override
    public void onShareStatusChangedNotify(ShareStatus shareStatus, int i) {
        Log.d(TAG, "onShareStatusChangedNotify: " + shareStatus);
        app.setShareStatus(shareStatus);
        app.notifyShareStatusChanged(shareStatus);
    }
});
LogUtil.d(TAG, "Create SDK object Success !!!!");
// 存入全局
app.setShareServiceController(shareServiceController);
}
```

## 2.6.4 说明

### 2.6.4.1 公网地址说明

编译连接存在如下公网地址，产品不会访问使用：

<https://android.googlesource.com/toolchain/clang>

<https://android.googlesource.com/toolchain/llvm>

### 2.6.4.2 权限说明

SDK已申请存储、截屏、悬浮窗权限，需要用户授权。

## 2.7 错误码参考

### 2.7.1 错误码

错误码对照表

值	描述
0	设备发现错误
1	连接错误
2	断开连接错误
3	启动投屏错误
4	停止投屏错误
5	遥控器参数查询结果错误
6	打开\关闭麦克风结果错误
7	调节扬声器音量结果错误
8	打开\关闭扬声器结果错误
9	打开\关闭摄像机结果错误

值	描述
10	离开会议结果错误
11	结束会议结果错误
12	获取SDK最新版本
13	正在处理连接请求
14	客户端已处于连接
15	投影码错误 数字+字母组合
16	鉴权失败
17	客户端被锁定
18	终端忙碌
19	终端正在升级
20	WEB被禁用
21	和客户端没有辅流公共能力
22	获取辅流令牌失败
23	带宽过低，无法发送辅流
24	会议无辅流能力
25	多终端连接忙碌
26	EUA鉴权失败
27	EUA客户端被锁定

# 3 IOS SDK

## 3.1 概述

投屏二次开发iOS SDK 提供了一套完整的接口集合，开发者可以通过调用ideaShare SDK（以下简称SDK）开放的API，快速集成投屏能力，包括启动投屏、遥控器、投放音频等。

## 3.2 变更记录

表 3-1 变更记录

日期	版本	变更内容
2021-03-05	1.0.1	初稿完成
2021-09-01	1.0.2	1.33SDK快速集成 增加录制进程创建方法

## 3.3 快速入门

本文面向有一定OC开发能力的开发者，以Demo编译流程为例，介绍如何使用SDK 进行二次开发

### 3.3.1 预置条件

支持SMC 2.0 , SMC 3.0, 云会议组网

### 3.3.2 开发环境准备

表 3-2 环境要求

名称	要求
Xcode版本	Xcode 12.0及以上版本

名称	要求
开发者签名	项目已设置有效的开发者签名

### 3.3.3 SDK 快速集成

#### 1. 下载SDK

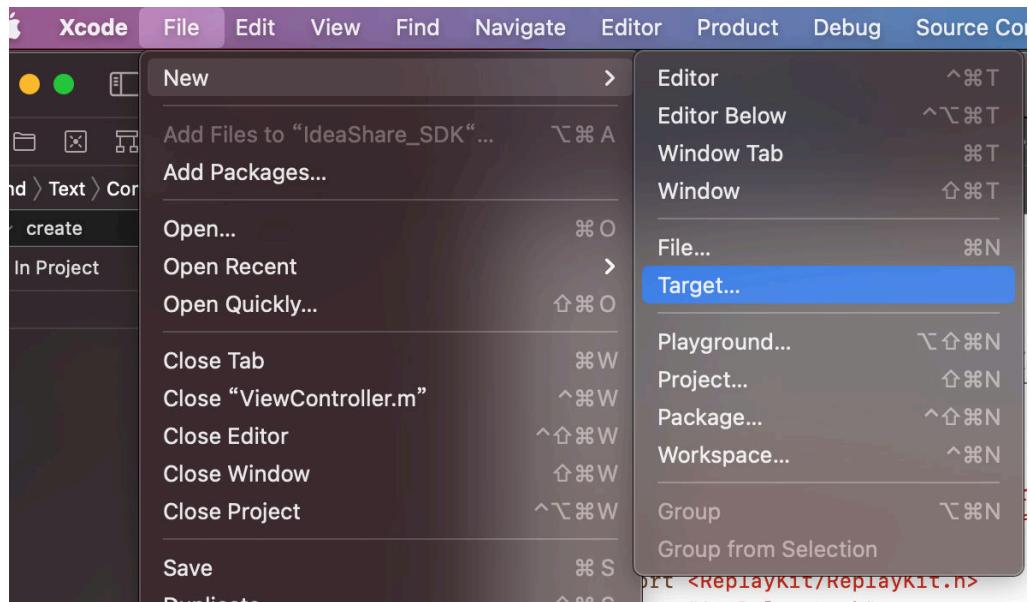
IOS的SDK是以framework库的形式供开发者集成使用，SDK下载解压后有两个framework文件，如图1-1所示，分别用于主程序和录制进程（IOS replaykit屏幕录制方案所依赖的专门用于录制屏幕的target）。

图1-1 文件目录

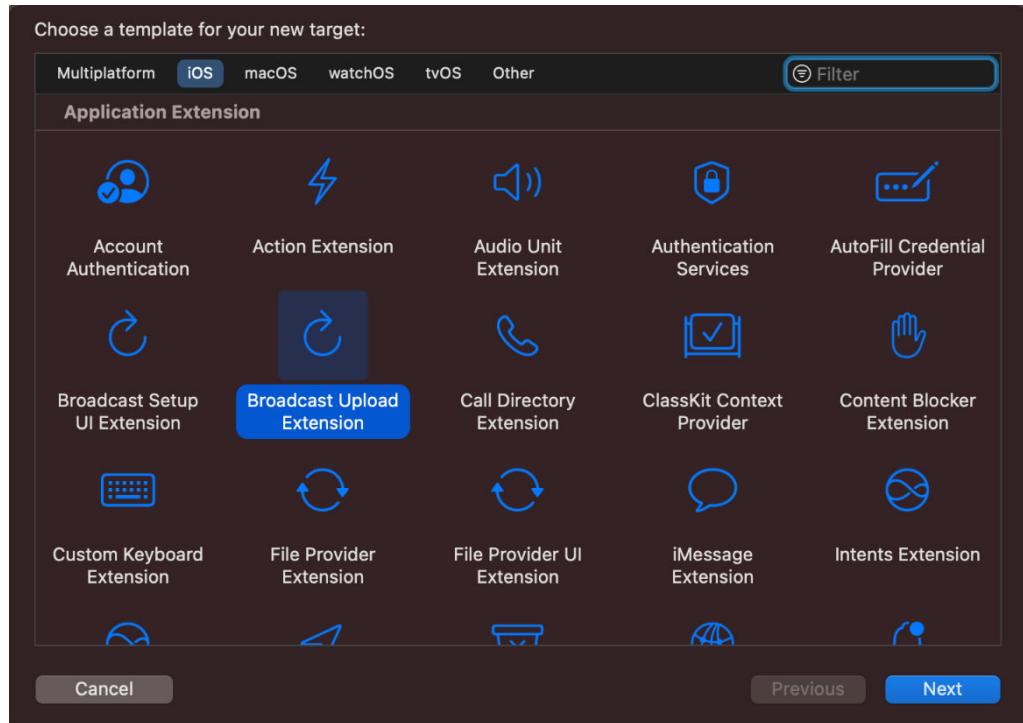


#### 2. 在项目中创建录制进程target

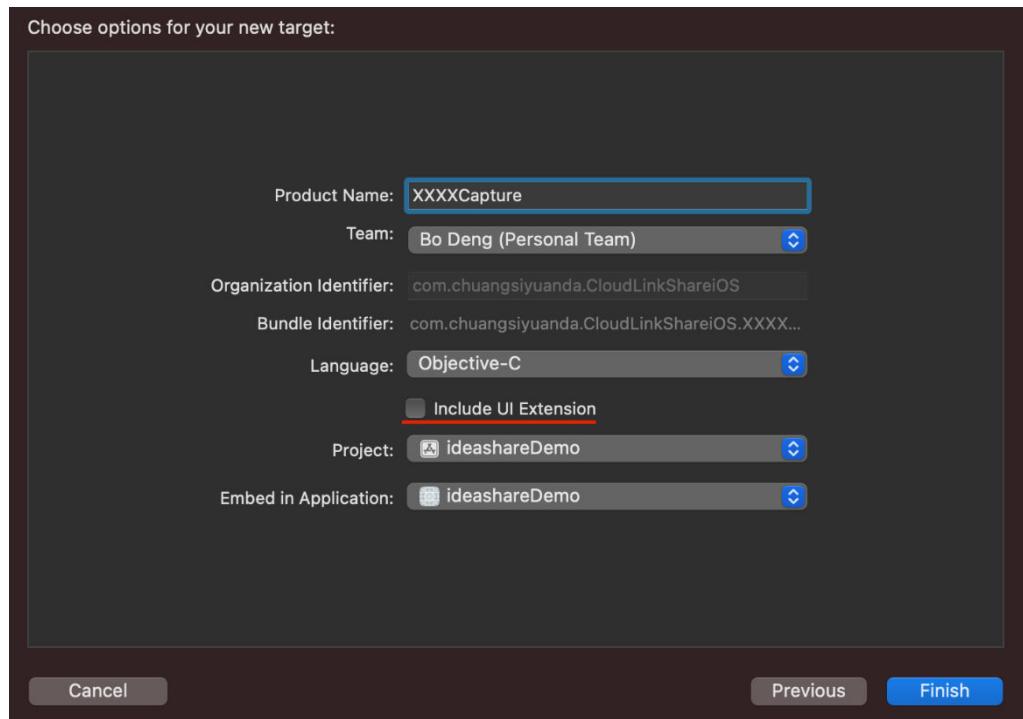
- 依次点击→file→New→Target



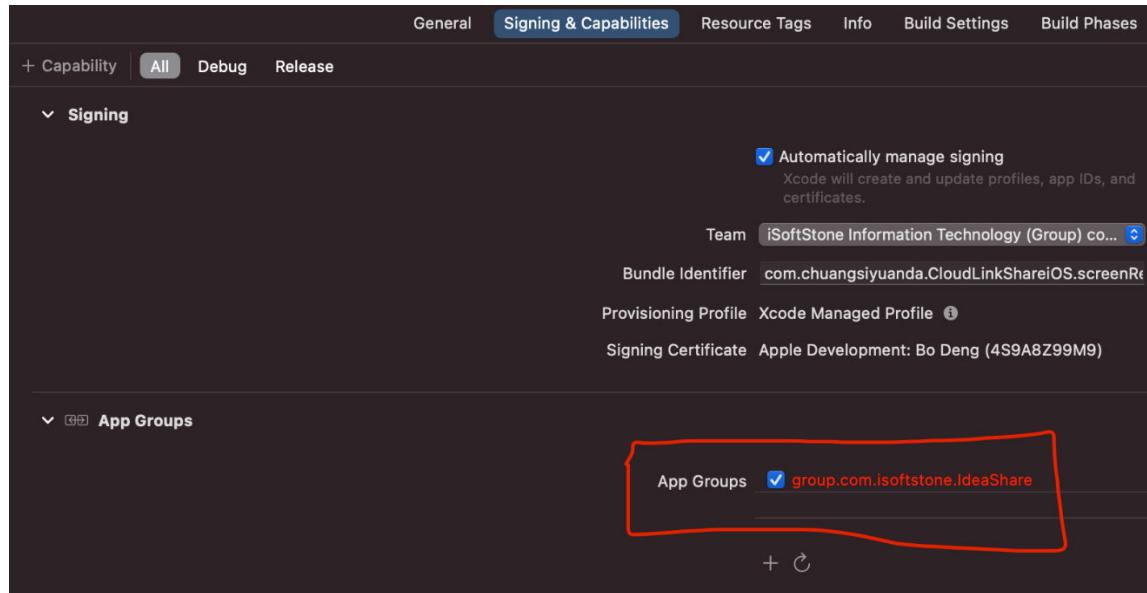
- 在弹出框中选择Broadcast Upload Extension



- 设置录制进程的名称（注意不用勾选 Include UI Extension）：



- 将录制进程的groupID设置成与主程序一致：

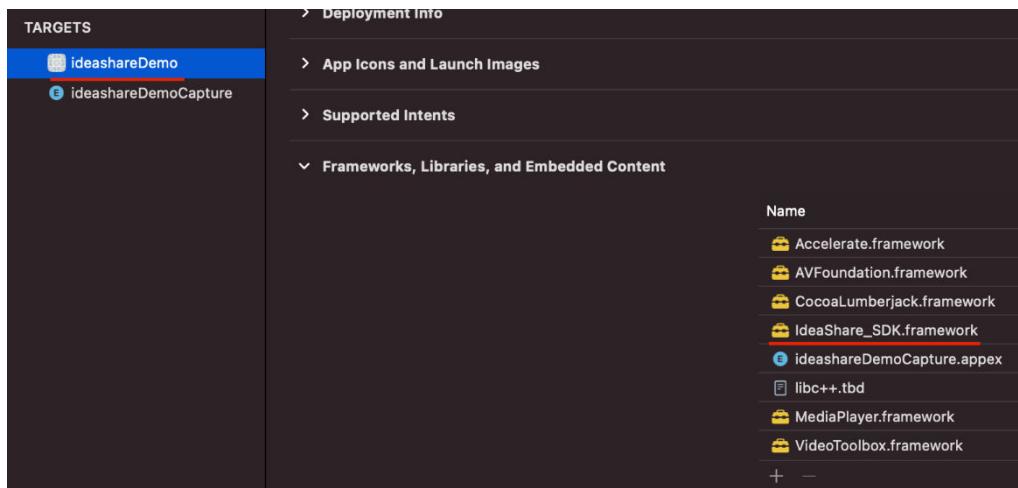


### 3. 在项目中导入SDK

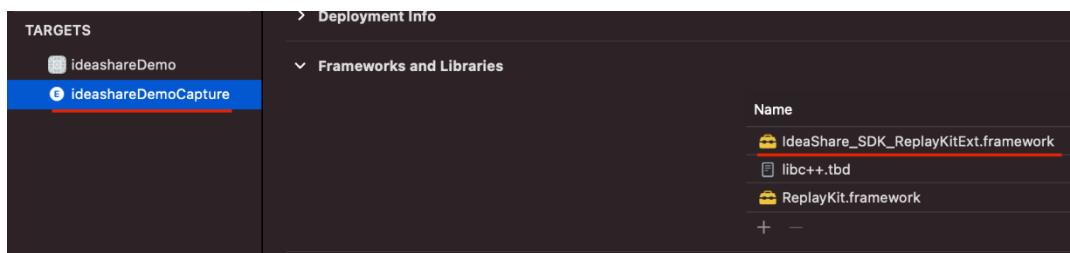
首先需要将SDK包中的framework库导入项目工程。

注意：

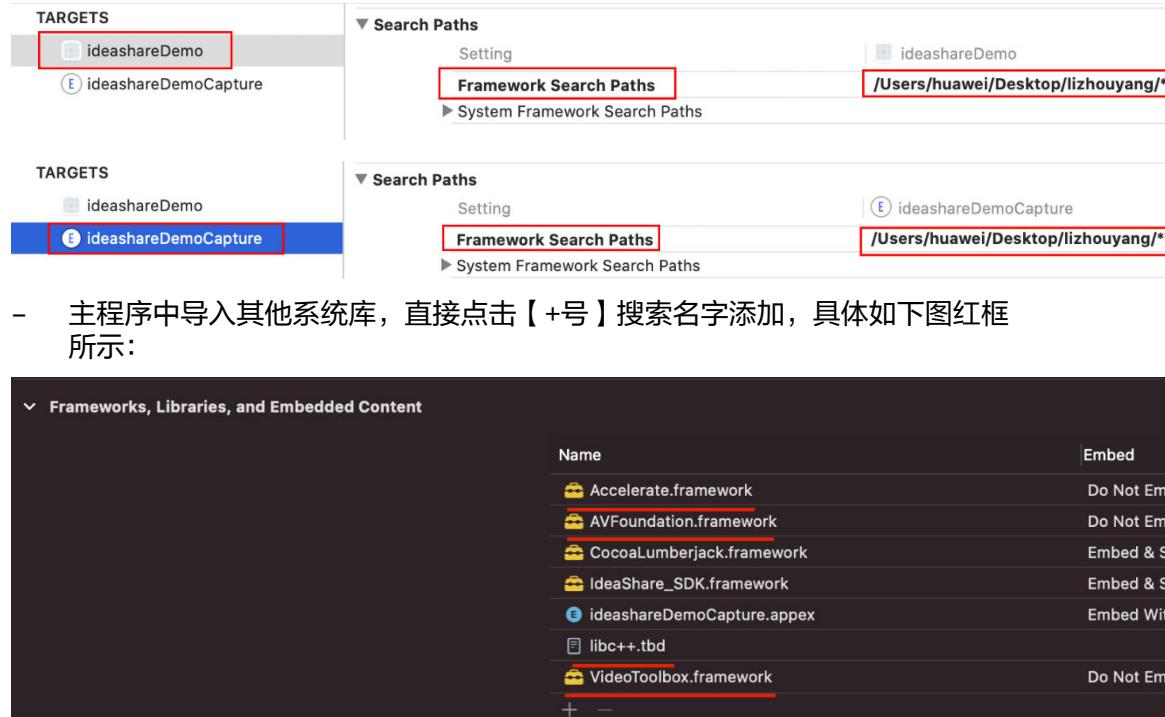
- 将<ideaShare\_SDK.framework>导入主程序  
( 注意：embed需要设置为Embed&Sign )



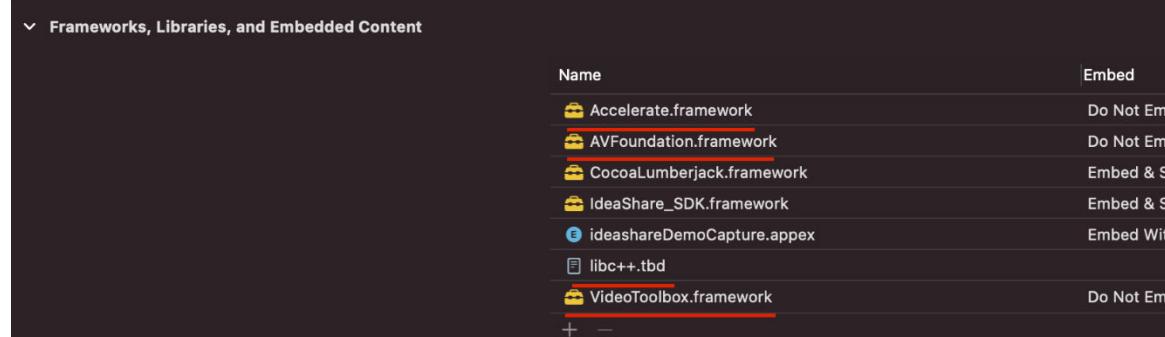
- 将<IdeaShare\_SDK\_ReplayKitExt.framework>导入录制进程对应的Target



- 确保framework在主程序和录制进程的【Build Settings】中的【Build Settings】中的Framework Search Paths能够找到。。



- 主程序中导入其他系统库，直接点击【+号】搜索名字添加，具体如下图红框所示：



#### 4. 在项目中使用API

完成上述两步之后就可以正常导入SDK中的头文件（如图1-4）并使用API接口了，接口使用流程参照图1-5。

注意：

- 投屏APP 启动前必须调用初始化接口
- 接口调用结果通过异步回调实现，如xxxResult， 服务端通知事件格式为xxxNotify

图1-4 导入头文件

```
#import <ideaShareSDK_ios/IdeaShareServiceController.h>
#import <ideaShareSDK_ios/IdeaShareDef.h>
```

图1-5 接口调用流程

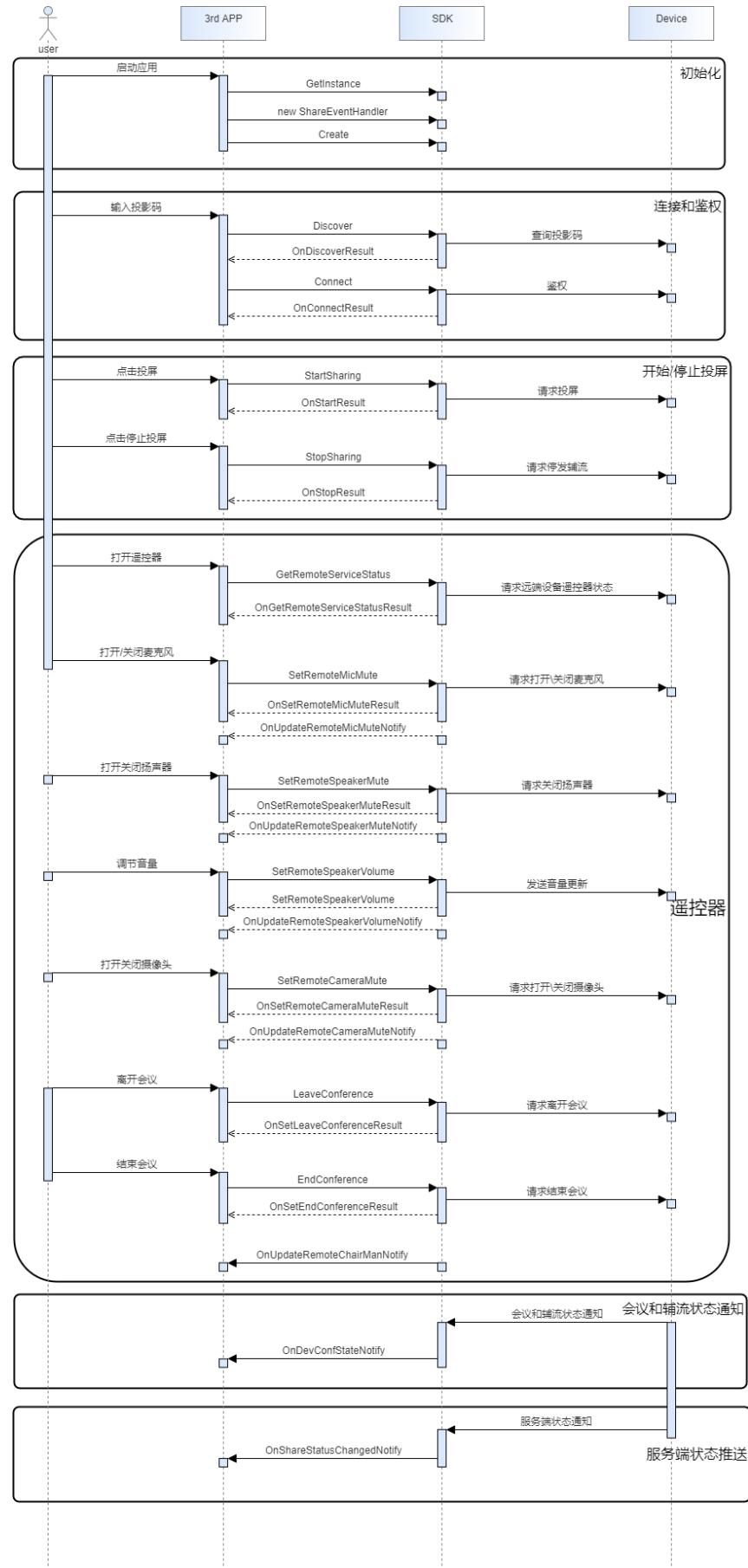
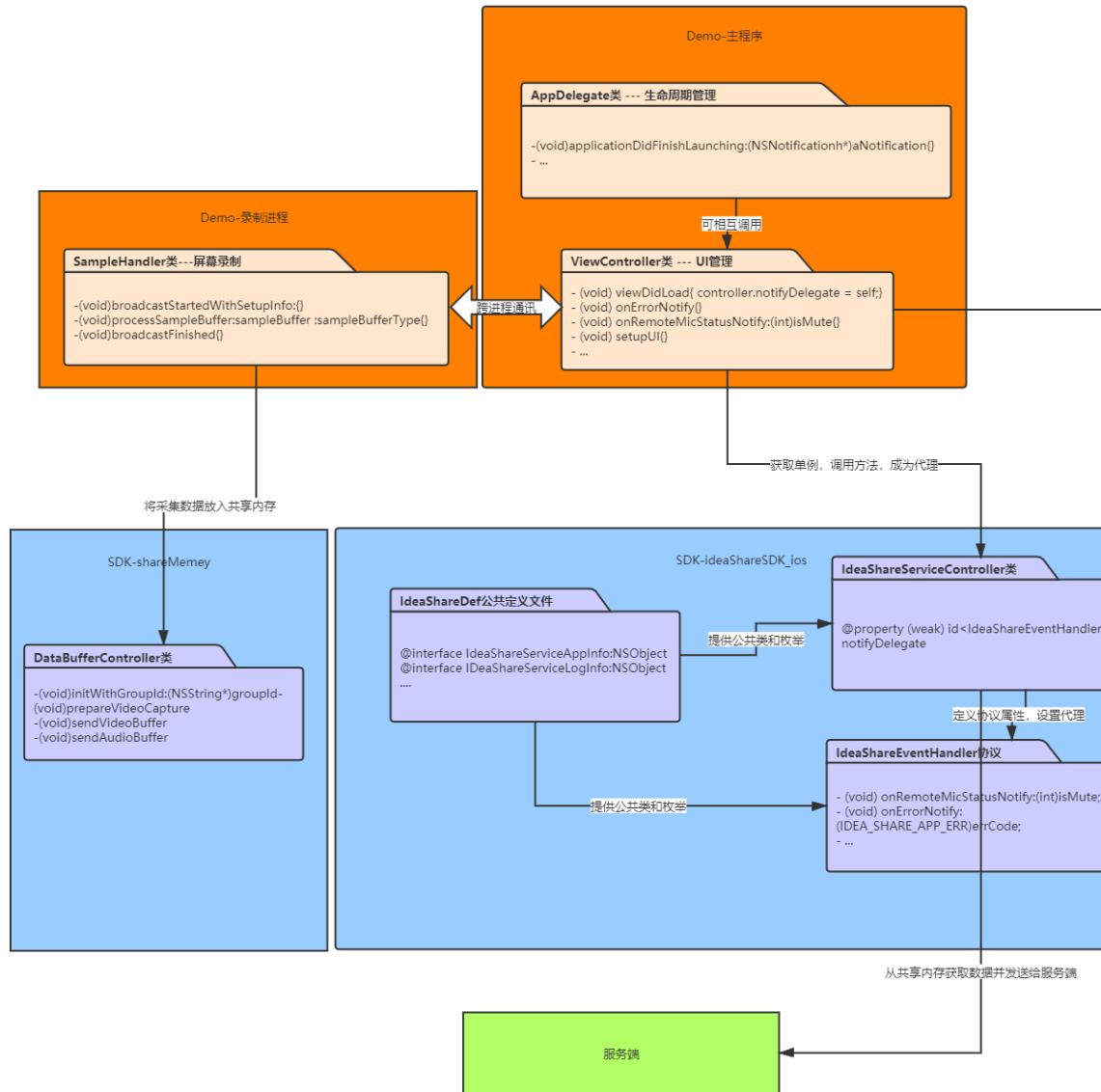


图1-6 整体架构



## 3.4 典型场景

### 3.4.1 场景 1： 初始化

#### 描述

使用投屏SDK业务组件时，需要先完成SDK的基础组件的初始化。

#### 业务流程

1. 调用IdeaShareServiceController的shareInstance类方法获取Controller的单例

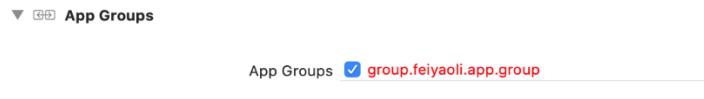
```
// 获取单例
_mShareController = [IdeaShareServiceController shareInstance];
```

## 2. 初始化dataBuffer录屏数据共享内存空间

```
[_mShareController initDataBuffer:kGroup];
```

KGroup参数为项目工程的Group ID即Signing&Capabilities设置页签中的App Groups项，如图1-6所示。

图1-6 Group ID



## 3. 初始化日志、SDK和音频策略

```
//初始化日志
_logInfo = [[[IdeaShareServiceLogInfo alloc] init];
_logInfo.enable = true;
_logInfo.logLevel = 3;
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSCachesDirectory, NSUserDomainMask, YES);
NSString *baseDir = paths.firstObject;
NSString *logsDirectory = [baseDir stringByAppendingPathComponent:@"Logs"];
_logInfo.path = logsDirectory;
[_mShareController setLog:_logInfo];

//初始化SDK
_appInfo = [[[IdeaShareServiceAppInfo alloc] init];
[_mShareController create:_appInfo];

//初始化音频策略
_policy = [[SharePolicy alloc] init];
_policy.isShareAudio = YES;
_policy.isCompatibility = YES;
[_mShareController setPolicy:_policy];
```

## 4. 注册通知

将自身设置为IdeaShareServiceController单例的通知事件代理，并自定义实现代理方法，用以收到服务端事件通知后做相应的处理

```
_mShareController.notifyDelegate = self;
```

```
- (void) onShareStatusChangeNotify : (IDEA_SHARE_APP_SERVER_STATUS) status : (IDEA_SHARE_ERROR_ID) reason;
- (void) onDevConfStatusChangeNotify : (IDEA_SHARE_APP_CONF_STATUS) confStatus : (IDEA_SHARE_APP_AUX_STATUS)
auxStatus;
- (void) onRemoteMicStatusNotify : (int) isMute;
- (void) onRemoteCameraStatusNotify : (int) isCameraMute;
- (void) onRemoteSpeakerStatusNotify : (int) isSpeakerMute;
- (void) onRemoteChairManStatusNotify : (int) isChairMan;
- (void) onRemoteSpeakerVolumeStatusNotify : (int) speakerVolume;
- (void) onErrorNotify : (IDEA_SHARE_APP_ERR) errCode;
```

## 注意事项

- Group ID需要用户传入本工程的App Groups；
- 录制进程和主程序的App Groups保持一致；
- 录制进程不需要初始化，在开始共享时由主程序唤醒，具体步骤见【场景5开始投屏】

## 3.4.2 场景 2：查询投屏码

### 描述

用户输入投屏码后，返回连接设备IP和密码

## 业务流程

1. 从UI输入组件获取用户输入的投屏码、EUA对应的IP地址、端口号
2. 调用 ShareServiceController的discover 接口，根据业务场景传入投屏码、EUA对应的IP地址、端口号
3. 处理回调中的结果，比如打印解析的地址，并进行connect连接

## 注意事项

- 初始化SDK之后才能调用查询投影码
- 非EUA场景，IP和端口号直接传空

## 示例代码

```
[_mShareController discover:_projectionNum.text :_projectionIP.text :[_projectionPort.text intValue] :^(id t) {
    DiscoverResult *discoverResult = (DiscoverResult*)t;
    DDLogInfo(@"password is : %@",discoverResult.password);
    for (NSString * str in discoverResult.address_list)
    {
        DDLogInfo(@"address list is : %@",str);
    }
    NSMutableArray *address_list = discoverResult.address_list;
    NSString *password = discoverResult.password;
    int port = 1444;
    [self->_mShareController connect:address_list :port :password :^(id t) {
        DDLogInfo(@"connect OK!!!!");
        //刷新UI界面
        //...
    }];
}];
```

### 3.4.3 场景 3：设置 TLS 证书验证

#### 描述

证书验证

## 业务流程

### 证书验证接口调用

1. 调用setTls接口
2. 通过setTls接口传递TLS信息给SDK层进行证书验证，确保所传路径有证书存在，并将验证模式设为SERVER，否则验证会失败，并且无法成功连接。

## 注意事项

不调用此接口，不影响连接和投屏等功能，仅作为安全验证使用。如果验证失败，会导致无法连接。

## 示例代码

```
//设置TLS
IdeaShareServiceTlsInfo *tlsInfo = [[IdeaShareServiceTlsInfo alloc] init];
tlsInfo.caCertpath = @"/Users/**/Desktop";
tlsInfo.clientCertpath = @"/Users/**/Desktop";
tlsInfo.verifyMode = 1;
int result = [mShareServiceController setTls: tlsInfo :^(id tlsCallback) {
}];
```

## 3.4.4 场景 4：连接/断开连接

### 描述

连接设备或断开与设备的连接（ideahub或Board）

### 业务流程-连接

1. 调用connect接口
2. 处理连接回调结果

### 业务流程-断开连接

1. 判断是否投屏中，如果投屏中需要先执行【场景6停止投屏】的流程。
2. 调用disconnect接口
3. 处理断开连接回调结果

### 注意事项

- 必须在调用初始化和查询投影码后才能调用此接口
- 建议在查询投影码的回调block中调用connect连接接口
- 断开连接时需要判断是否正在投屏中，如果正在投屏需要先调用停止投屏接口

### 示例代码-连接

```
[_mShareController discover:_projectionNum.text :_projectionIP.text :[_projectionPort.text intValue] :^(id t) {
    DiscoverResult *discoverResult = (DiscoverResult*)t;
    DDLogInfo(@"password is : %@",discoverResult.password);
    for (NSString * str in discoverResult.address_list)
    {
        DDLogInfo(@"address list is : %@",str);
    }
    NSMutableArray *address_list = discoverResult.address_list;
    NSString *password = discoverResult.password;
    int port = 1444;
    [self->_mShareController connect:address_list :port :password :^(id t) {
        DDLogInfo(@"connect OK!!!!");
        dispatch_async(dispatch_get_main_queue(), ^{
            // 显示已连接设备
            DeviceInfo *connectedDevice = (DeviceInfo *)t;
            self->_deviceNameLbl.text = [[NSString
                alloc]initWithFormat:@"已连接设备: %@",connectedDevice.deviceName];
            [self.view addSubview:self->_deviceNameLbl];
        });
    }];
}];
```

## 示例代码-断开连接

```
if(isSharing){  
    //1.通知录制进程停止采集  
    CFNotificationCenterPostNotification(CFNotificationCenterGetDarwinNotifyCenter(),  
        CFSTR("app_tell_extension_finished"), NULL,NULL, YES);  
    isSharing = NO;  
    //2.调用停止共享接口  
    [_mShareController stopSharing:^(id t){}];  
}  
[_mShareController disconnect:^(id t){  
    //UI界面处理  
    //...  
}];
```

## 3.4.5 场景 5： 开始投屏

### 描述

连接设备后，调用共享接口发起屏幕共享。

### 业务流程-开始投屏-主程序侧

1. 调用setPolicy接口设置音频投屏策略
2. 唤醒录制进程系统弹窗，进行音视频数据采集
3. 收到录制进程开始采集的通知后，调用startSharing接口启动投屏

### 业务流程-开始投屏-录制进程侧

1. 创建DataBufferController对象并初始化；
2. 发送达尔文通知给主程序；
3. 开始录制，调用sendVideoBuffer和sendAudioBuffer接口发送采集的数据；

### 注意事项

- 投屏前需要先成功连接设备；
- 投屏前一定要先设置音频投屏策略（是/否投放音频）；
- 录制进程开始采集后才能调用startSharing接口，建议主程序在收到录制进程开始采集的通知后调用；

## 示例代码-开始投屏-主程序侧

```
// 1.设置音频投屏策略  
_policy.isShareAudio = _audioSwitch.on;  
[_mShareController setPolicy:_policy];
```

```
//2.唤醒录制进程
if (@available(iOS 12.0, *)) {
    RPSYSTEMBroadcastPickerView * broadcastPickerView = [[RPSYSTEMBroadcastPickerView alloc]
        initWithFrame:CGRectZero];
    broadcastPickerView.preferredExtension = REPLAY_EXTENSION_ID;
    broadcastPickerView.showsMicrophoneButton = NO;
    for (UIView *view in broadcastPickerView.subviews) {
        if ([view isKindOfClass:[UIButton class]]) {
            [(UIButton*)view sendActionsForControlEvents:UIControlEventTouchUpInsideAllTouchEvents];
        }
    }
} else {
    // Fallback on earlier versions
}
```

录制进程的Bundle ID

```
//注册监听达尔文通知，实现跨进程接收录制进程开始采集的通知
CFNotificationCenterAddObserver(CFNotificationCenterGetDarwinNotifyCenter(), (__bridge const void *)(self),
    extensionStartedWithSetupInfo, CFSTR("extension_setup_info"), NULL,
    CFNotificationSuspensionBehaviorDeliverImmediately);
```

```
//3.调用startSharing接口启动投屏
void extensionStartedWithSetupInfo (CFNotificationCenterRef center, void *observer, CFStringRef name, const void
    *object, CFDictionaryRef userInfo) {
    int shareRes = [selfClass -> mShareController startSharing:^(id t){}];
    if(shareRes == 0){
        //UI界面处理。。
        isSharing = YES;
    }
}
```

## 示例代码-开始投屏-录制进程侧

```
//录制进程唤醒且倒计时321完成后 iOS系统会执行此方法:
- (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *, NSObject *> *)setupInfo {

    selfClass = self;
    //1.初始化dataBufferController对象，传入groupID
    _dataBufferController = [[DataBufferController new] initWithGroupId:kGroup];

    //2.发送达尔文通知给主程序，并注册监听来自主程序的消息
    [self postAndReceiveDarwinNotificationToApp];
}

- (void)postAndReceiveDarwinNotificationToApp {
    CFNotificationCenterRef notification = CFNotificationCenterGetDarwinNotifyCenter ();
    //发送开始采集的通知给主程序
    CFNotificationCenterPostNotification(CFNotificationCenterGetDarwinNotifyCenter (),
        CFSTR("extension_setup_info"), NULL, NULL, YES);
    CFNotificationCenterAddObserver(notification, (__bridge const void *)(self),
        extensionRecvShareDataStop, CFSTR("app_tell_extension_finished"), NULL,
        CFNotificationSuspensionBehaviorDeliverImmediately);
}
```

```
//屏幕录制中IOS系统每秒会调用33次此方法
- (void)processSampleBuffer:(CMSSampleBufferRef)sampleBuffer withType:(RPSampleBufferType)sampleBufferType {
    //3. 发送采集的数据
    switch (sampleBufferType) {
        case RPSampleBufferTypeVideo:
            [self->_dataBufferController sendVideoBuffer:sampleBuffer];
            break;
        case RPSampleBufferTypeAudioApp:
            [self->_dataBufferController sendAudioBuffer:sampleBuffer];
            break;
        case RPSampleBufferTypeAudioMic:
            // Handle audio sample buffer for mic audio
            break;
        default:
            break;
    }
}
```

### 3.4.6 场景 6：停止投屏

#### 描述

屏幕共享中，调用接口停止投屏，共有两种方式主动发起：

- 主程序中发起
- 录制进程的系统弹窗中发起，如下图所示：



#### 业务流程-主程序发起停止

1. 主程序发送达尔文通知，通知录制进程停止采集
2. 主程序调用stopSharing接口停止投屏
3. 录制进程收到通知后停止采集

#### 业务流程-录制进程发起停止

1. 录制进程发送达尔文通知，通知主程序停止投屏
2. 主程序收到通知后调用stopSharing接口停止投屏

#### 注意事项

- 主程序中停止投屏包括共享过程中断开连接

## 示例代码-主程序发起停止

```
- (void)shareControl:(UIButton*)btn{
    if(isSharing){
        //1.通知录制进程停止采集
        CFNotificationCenterPostNotification(CFNotificationCenterGetDarwinNotifyCenter(),
                                             CFSTR("app_tell_extension_finished"), NULL,NULL, YES);
        isSharing = NO;
        //2.调用stopSharing接口停止共享
        [selfClass->_mShareController stopSharing:^(id t){}];
    }
}

//注册监听主程序停止投屏的通知
CFNotificationCenterAddObserver(notification, (__bridge const void *)(self),
                               extensionRecvShareDataStop,CFSTR("app_tell_extension_finished"), NULL,
                               CFNotificationSuspensionBehaviorDeliverImmediately);

void extensionRecvShareDataStop(CFNotificationCenterRef center, void *observer, CFStringRef name, const void
*object, CFDictionaryRef userInfo) {
    NSDictionary * userInfotmp = [NSDictionary dictionaryWithObjectsAndKeys:@ "", NSLocalizedDescriptionKey,
                                  @"投屏中断", NSLocalizedFailureReasonErrorKey, @ "", NSLocalizedRecoverySuggestionErrorKey,nil];
    NSError *error = [[NSError alloc] initWithDomain:NSCocoaErrorDomain code:4 userInfo:userInfotmp];
    //
    [selfClass finishBroadcastWithError:error]; → 停止采集
}
```

## 示例代码-录制进程发起停止

```
- (void)broadcastFinished {
    CFNotificationCenterRef notification = CFNotificationCenterGetDarwinNotifyCenter ();
    CFNotificationCenterRemoveObserver(notification, (__bridge const void *)(self), CFSTR
NULL);
    //通知主程序停止共享
    CFNotificationCenterPostNotification(CFNotificationCenterGetDarwinNotifyCenter (),
                                         CFSTR("extension_stop_info"), NULL,NULL, YES);
}

- (void)registerNotification {
    CFNotificationCenterAddObserver(CFNotificationCenterGetDarwinNotifyCenter(), (__bridge const void *)(self)
                                 writeMyLogMethod,CFSTR("writeMyLog"), NULL, CFNotificationSuspensionBehaviorDeliverImmediately);
    //注册监听达尔文通知，实现跨进程接收录制进程开始采集的通知
    CFNotificationCenterAddObserver(CFNotificationCenterGetDarwinNotifyCenter(), (__bridge const void *)(self)
                                 extensionStartedWithSetupInfo,CFSTR("extension_setup_info"), NULL,
                                 CFNotificationSuspensionBehaviorDeliverImmediately);
    //注册监听录制进程停止采集的通知
    CFNotificationCenterAddObserver(CFNotificationCenterGetDarwinNotifyCenter(), (__bridge const void *)(self)
                                 extensionStop,CFSTR("extension_stop_info"), NULL, CFNotificationSuspensionBehaviorDeliverImmediately);

void extensionStop (CFNotificationCenterRef center, void *observer,
                  CFDictionaryRef userInfo) {
    if(isSharing == YES){
        [selfClass->_mShareController stopSharing:^(id t){}];
        isSharing = NO;
    }
}
```

### 3.4.7 场景 7：投放音频

#### 描述

共享前或共享中设置音频投放策略，选择投放或不投放音频

## 业务流程

1. 共享前调用SetPolicy 设置音频投放策略
2. 共享中调用SetPolicy 修改音频投放策略

## 注意事项

- 需要先连接才能调用此接口

## 示例代码

```
// 1.开始共享前设置音频投屏策略
_policy.isShareAudio = _audioSwitch.on;
[_mShareController setPolicy:_policy];

-(void)audioSwitchChg:(UISwitch*)audioSwitch{
    _policy.isShareAudio = audioSwitch.isOn ? YES : NO;
    if(isSharing == YES){
        //共享中修改音频投放策略
        [_mShareController setPolicy:_policy];
    }
}
```

## 3.4.8 场景 8： 遥控器

### 描述

在连接设备后，客户端可以查询设备状态，包括：麦克风状态、扬声器状态、摄像头状态、音量大小、是否在会议中，并且操作这些设备状态，如调节音量大小、离开会议（前提是已经入会）

## 业务流程

1. 调用getRemoteServiceStatus查询连接设备状态
2. 调用遥控器接口（如setRemoteMicMute）请求遥控远端设备
3. 收到服务端主动推送过来的通知事件（如OnRemoteMicStatusNotify），更新客户端遥控器状态
4. 在会议中时，调用leaveConference 或endConference 离开会议

## 注意事项

- 连接设备后遥控器功能才可用
- 需要先将自身设置为IdeaShareServiceController单例的代理，才能收到服务端的通知，具体操作参照场景1初始化

## 示例代码

```
// 查询远端设备状态
[_mShareController getRemoteServiceStatus:^(id t){
    RemoteServiceStatus * remoteServiceStatus = (RemoteServiceStatus *)t;
    // 是否主席
    self->_isChairman = remoteServiceStatus.isChairman; 回调block
    dispatch_async(dispatch_get_main_queue(), ^{
        // mic开关状态
        self->_micSwitch.on = remoteServiceStatus.isMute == 1 ? NO : YES;
        // 摄像头开关状态 如果设备是ideaboard 无摄像头
        if(remoteServiceStatus.isBase == 0){
            //不是ideaboard UI页面逻辑
            //...
        }else{
            //是ideaboard UI页面逻辑
            //...
        }
        // 音量状态
        self->_volumeSlider.value = remoteServiceStatus.volume;
        // 静音状态
        if(remoteServiceStatus.isSpeakerMute == 1){
            //UI页面逻辑
            //...
        }
    });
}];
```

```
//摄像头开关点击事件处理
- (void) cameraSwitchChg:(UISwitch*)sw{
    if(sw.on == NO){
        //摄像头开关打开
        [_mShareController setRemoteCameraMute:1 :^(id t){}];
    } else{
        //摄像头开关关闭
        [_mShareController setRemoteCameraMute:0 :^(id t){}];
    }
}
```

```
//服务端麦克风状态通知
- (void) OnRemoteMicStatusNotify : (int) isMute{
    dispatch_async(dispatch_get_main_queue(), ^{
        self->_micSwitch.on = isMute==0 ? YES : NO;
    });
}

//服务端摄像头状态通知
- (void) OnRemoteCameraStatusNotify : (int) isCameraMute{
    dispatch_async(dispatch_get_main_queue(), ^{
        self->_cameraSwitch.on = isCameraMute==0 ? YES : NO;
    });
}

//服务端静音状态通知
- (void) OnRemoteSpeakerStatusNotify : (int) isSpeakerMute{
    dispatch_async(dispatch_get_main_queue(), ^{
        //UI界面逻辑
        //...
    });
}

//服务端音量变化通知
- (void) OnRemoteSpeakerVolumeStatusNotify : (int) speakerVolume{
    dispatch_async(dispatch_get_main_queue(), ^{
        self->_volumeSlider.value = speakerVolume;
    });
}
```

### 3.4.9 场景 9：设备推送

#### 描述

远端设备定时向客户端推送投屏状态、辅流状态、会议状态，如果在会议中申请主席，还会推送主席状态

#### 业务流程

1. 处理投屏状态、辅流状态、会议状态、以及主席状态

#### 示例代码

```
//投屏状态变化通知 IDEA_SHARE_APP_SERVER_STATUS和IDEA_SHARE_ERROR_ID的定义见接口参考
- (void) onShareStatusChangeNotify : (IDEA_SHARE_APP_SERVER_STATUS) status : (IDEA_SHARE_ERROR_ID) reason{

    if(status == SERVER_STATUS_STOP_SHARE){
        //服务端停止发送辅流后的处理
    }
    if(status == SERVER_STATUS_DISCONNECT){
        //服务端断开连接后的处理
    }
}

//辅流和会议状态变化通知
- (void) onDevConfStatusChangeNotify : (IDEA_SHARE_APP_CONF_STATUS) confStatus : (IDEA_SHARE_APP_AUX_STATUS) auxStatus{
    if(confStatus == IDEA_SHARE_SDK_CONF_IDLE){
        //服务端接入会议后的处理
    }
    if(confStatus == IDEA_SHARE_SDK_CONF_BUSY){
        //服务端离开会议后的处理
    }
}
```

```
//主席状态变化通知
- (void) OnRemoteChairManStatusNotify : (int) isChairMan{
    //设备在会议中申请主席后，收到服务端推送主席状态变化后的处理
}
```

### 3.4.10 场景 10：错误上报

#### 描述

在使用过程中，出现异常时，SDK上报OnError事件

#### 业务流程

处理OnError 通知事件回调

#### 示例代码

```
- (void) onErrorNotify : (IDEA_SHARE_APP_ERR) errCode{
    switch (errCode) {
        case PROJECTION_CODE_ERROR:{
            //投屏码错误处理...
        }
        case CLIENT_IS_LOCKED_ERROR:{
            //客户端锁定错误处理...
        }
        case AUTH_ERROR:{
            //鉴权失败错误处理...
        }
        case CONNECT_ERROR:{
            //连接失败错误处理...
        }
        case CONF_NO_DATA_CAP_ERROR:{
            //不支持演示错误处理
        }
        default:{
            //默认处理...
        }
    }
}
```

### 3.4.11 场景 11：去初始化

#### 描述

退出客户端APP时，调用去初始化SDK接口

#### 业务流程

调用去初始化SDK接口

## 示例代码

```
- (void) dealloc {
    if (_mShareController != NULL) {
        [_mShareController destroy];
    }
}
```

## 3.5 常见问题

【写作要求】FAQ 提供集成过程中常见的问题；例如：定位思路和手段方法或工具，错误码，日志分析。介绍内容需要具有通用性。

### 2.5.1

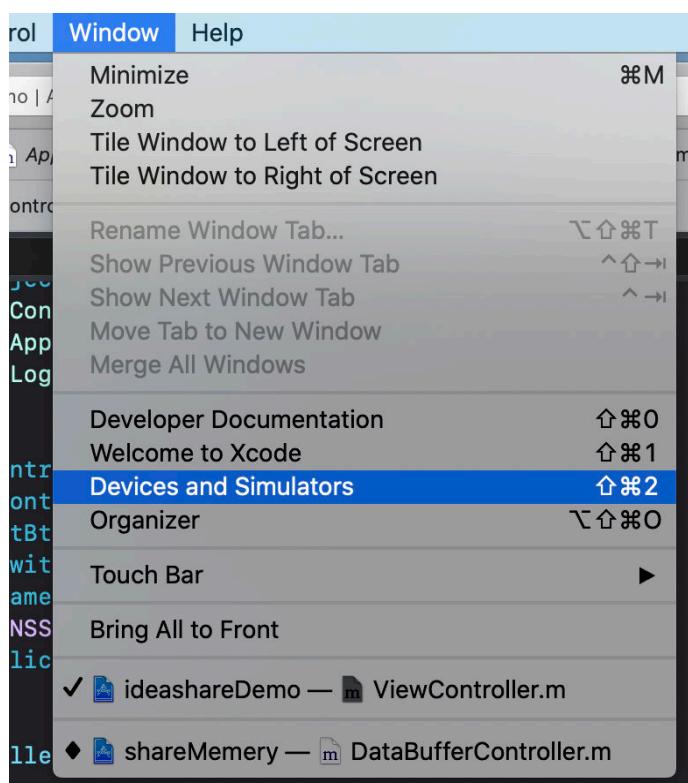
#### 2.5.1

### 3.5.1 获取日志

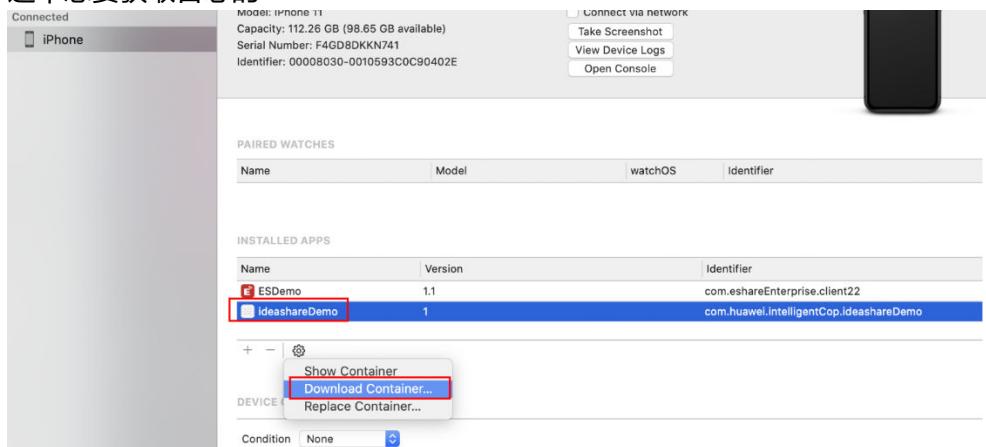
1. 将IOS设备接入安装了XCODE的MAC电脑



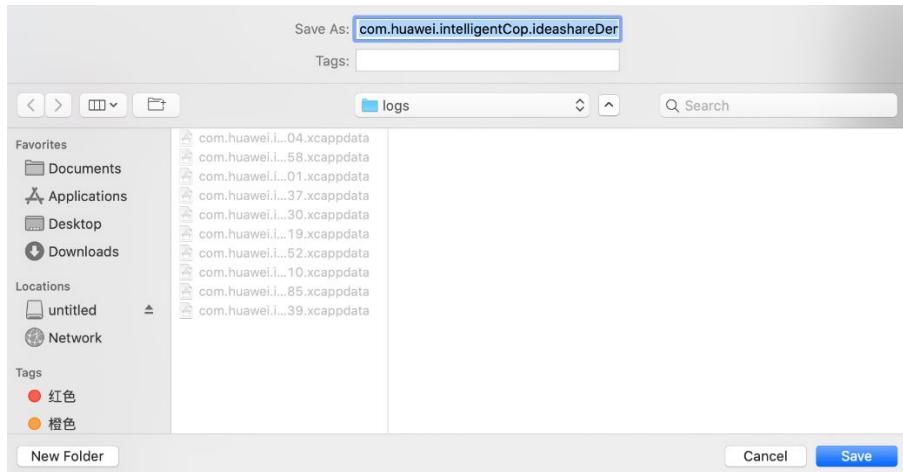
2. 选择导航栏的window->Devices and Simulators



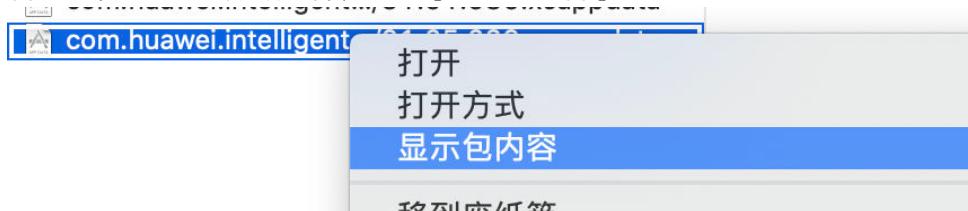
3. 选中想要获取日志的APP



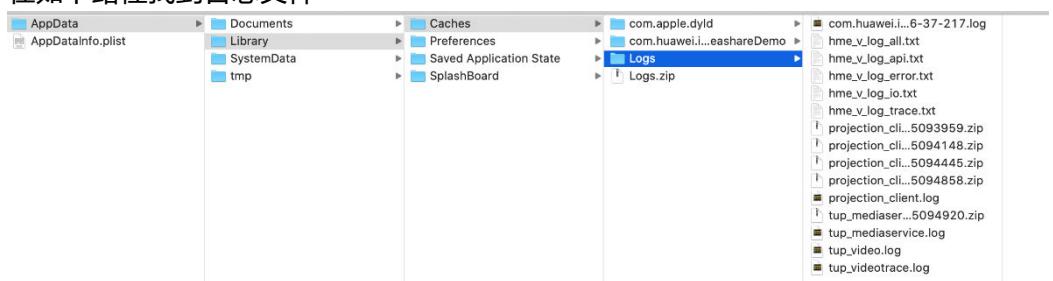
4. 选择路径保存APP的数据文件



5. 右键下载好的APP数据文件，选择【显示包内容】



6. 在如下路径找到日志文件



7. 压缩Logs文件夹，发送给开发人员定位问题

## 3.6 IOS 接口参考

## 3.6.1 概述

本文主要介绍投屏SDK Objective接口参考。主要包含接口函数定义、参数定义说明、附录，事件上报通知说明以及错误码等信息。

## 3.6.2 变更记录

表 3-3 变更记录

日期	版本	变更内容
2021-02-23	1.0.0	初稿完成。
2021-04-26	2.0.0	接口修订

## 3.6.3 接口参考

### 3.6.3.1 启动 APP

#### 3.6.3.1.1 IdeaShareServiceController : 投屏控制类，主要是提供业务接口

获取IdeaShareServiceController 单例

```
mShareServiceController = [IdeaShareServiceController shareInstance];
```

**接口描述**

获取IdeaShareServiceController 单例

**注意事项**

应用程序调用接口都依赖此单例，建议在应用程序初始化的时候就获取到此单例，并一直持有

**接口定义**

```
+ (instancetype)shareInstance;
```

#### 3.6.3.1.2 初始化日志

APP调用IdeaShareServiceController实例对象的方法初始化SDK日志

```
int logRet = [mShareServiceController setLog:logInfo];
```

**接口描述**

初始化日志前获取IdeaShareServiceController单例，然后调用IdeaShareServiceController的setLog

**注意事项**

请传入正确的IdeaShareServiceLogInfo对象，初始化日志接口在初始化SDK之前定义。

**接口定义**

```
- (int)setLog:(IdeaShareServiceLogInfo *)logInfo;
```

#### 参数描述

表 3-4 参数说明

参数	是否必须	类型	描述
logInfo	是	IdeaShareServiceLogInfo	初始化日志的必要参数，OC类

### 3.6.3.1.3 初始化 SDK

APP调用IdeaShareServiceController实例对象的create方法

```
const IdeaShareServiceAppInfo *applInfo = [[IdeaShareServiceAppInfo alloc] init];
applInfo.applInfo = @"aa";
applInfo.exePath = @"bb";
int ret = [mShareServiceController create:applInfo];
```

#### 接口描述

初始化SDK

#### 注意事项

调用此接口前，要先给IdeaShareServiceController单例的notifyDelegate代理属性赋值为(id)self，否则初始化会失败。

#### 接口定义

```
- (int)create:(const IdeaShareServiceAppInfo *)applInfo;
```

#### 参数描述

表 3-5 参数说明

参数	是否必须	类型	描述
applInfo	是	IdeaShareServiceAppInfo	目前，此参数可任意赋值，无实际意义，但不能为nil

### 3.6.3.1.4 去初始化 SDK

应用程序退出时应去初始化SDK

```
[mShareServiceController destroy];
```

#### 接口描述

去初始化SDK

#### 注意事项

mShareServiceController为IdeaShareServiceController的单例

#### 接口定义

```
- (int)destroy;
```

### 3.6.3.1.5 投屏策略配置

APP调用IdeaShareServiceController实例对象的setPolicy方法

```
self.sharepolicy.isShareAudio = YES;  
self.sharepolicy.shareQuality = 1;  
self.sharepolicy.isCompatibility = YES;  
int result = [mShareServiceController setPolicy:self.sharepolicy];
```

#### 接口描述

该接口用于在连接设备后，设置是否需要投放音频到设备。

#### 注意事项

投屏策略暂时未使用，因此该接口仅用于音频控制

#### 接口定义

```
- (int)setPolicy:(SharePolicy *)sharePolicy;
```

表 3-6 参数说明

参数	是否必须	类型	描述
sharePolicy	是	SharePolicy	OC类，用于设置是否投放音频以及投屏策略

### 3.6.3.1.6 投屏码解析

```
NSString * castCode = self.codeString;  
NSString * address = self.ipaddressString;  
int port = [self.portString intValue];  
int result = [mShareServiceController discover:castCode withAddress:address withPort:port callback:^(id  
discoverCallback){  
    }];
```

#### 接口描述

APP调用IdeaShareServiceController实例对象的discover方法开始解析，SDK解析投影码成功后会通过Block回调将IP地址集和连接码传回UI层

#### 注意事项

离线模式下，仅需输入投屏码castCode；address和port适用于在线模式投屏

#### 接口定义

```
- (int)discover:(NSString *)castCode withAddress:(NSString *)address withPort:(int)port callback:  
(callback)discovercallback;
```

表 3-7 参数说明

参数	是否必须	类型	描述
castCode	是	NSString	投屏码
address	否	NSString	在线投屏时使用，为SMC(EUA)的IP地址

参数	是否必须	类型	描述
port	否	int	在线投屏时使用，为SMC(EUA)的端口号
discovercallback	是	DiscoverResult	typedef void (^callback)(id t), 包含回调内容 - 地址集，连接码以及解析结果

### 3.6.3.2 连接设备

#### 3.6.3.2.1 设置 TLS 验证

```
IdeaShareServiceTlsInfo *tlsInfo = [[IdeaShareServiceTlsInfo alloc] init];
tlsInfo.caCertpath = @"/Users/**/Desktop";
tlsInfo.clientCertpath = @"/Users/**/Desktop";
tlsInfo.verifyMode = 1;
int result = [mShareServiceController setTls: tlsInfo:^(id tlsCallback) {
}];
```

##### 接口描述

如果APP需要TLS证书验证，则调用IdeaShareServiceController实例对象的setTls方法，带上参数本地ca证书的路径和客户端校验的路径以及验证模式，证书验证成功后，便可进行连接操作；否则，连接不会成功。如果不调用此接口，则不需要本地证书，也可以正常连接成功。

##### 接口定义

```
- (int)setTls:(IdeaShareServiceTlsInfo *)tlsInfo:(callback)tlsCallback;
```

##### 参数描述

表 3-8 connect 参数说明

参数	是否必须	类型	描述
tlsInfo	是	IdeaShareServiceTlsInfo	TLS信息 OC类
tlsCallback	是	int	TLS校验失败会返回一个int值，否则不会传递任何值

#### 3.6.3.2.2 设备连接

```
NSMutableArray * addr_list = self.discoverResult.addressList;
NSString * password = self.discoverResult.password;
int port = 1444;
int connectResult = [mShareServiceController connect: addr_list withPort:port withPassword:password
callback:^(id connectCallback) {
}];
```

##### 接口描述

APP需要调用IdeaShareServiceController实例对象的connect方法，带上参数IP地址集、连接码（投屏码解析接口的回调结果）和端口开始连接（离线投屏端口默认为1444），SDK连接成功后会通过Block回调将DeviceInfo信息传回UI层，APP需要进行相应处理

### 接口定义

```
- (int)connect:(NSMutableArray *)addressList withPort:(int)port withPassword:(NSString *)password
callback:(callback)connectcallback;
```

### 参数描述

表 3-9 connect 参数说明

参数	是否必须	类型	描述
address_list	是	NSMutableArray	投屏码解析出来的地址集
port	是	int	端口号；离线投屏端口号固定为1444
password	是	NSString	投屏码解析出来的连接码
connectcallback	是	DeviceInfo	typedef void (^callback)(id t), 包含回调内容 - 设备地址，设备名称和是否支持远程控制

### 3.6.3.2.3 断开连接

```
int result = [mShareServiceController disconnect:^(id disconnectCallback) {
}];
```

### 接口描述

APP调用IdeaShareServiceController实例对象的disconnect方法。SDK断开连接成功后会通过Block回调通知UI层，APP需要进行相应处理

### 方法定义

```
- (int)disconnect:(callback)disconnectcallback;
```

### 参数描述

表 3-10 disConnect 参数说明

参数	是否必须	类型	描述
disconnectcallback	是	int	typedef void (^callback)(id t), 包含回调内容 - 断连的原因

### 3.6.3.2.4 开始投屏

```
int result = [mShareServiceController startSharing:^(id startSharingCallBack) {
}];
```

### 接口描述

APP调用IdeaShareServiceController实例对象的startSharing方法，SDK投屏成功之后会通过Block回调通知UI层，APP需要进行相应处理

### 方法定义

```
- (int)startSharing:(callback)startSharingCallBack;
```

#### 参数描述

表 3-11 startSharing 参数说明

参数	是否必须	类型	描述
StartSharingcallback	是	int	typedef void (^callback)(id t), 包含回调内容-开始投屏结果，暂无实际意义，未传递

### 3.6.3.2.5 结束投屏

```
int result = [mShareServiceController stopSharing:^(id stopSharingCallBack) {  
}];
```

#### 接口描述

APP调用IdeaShareServiceController实例对象的stopSharing方法。SDK结束投屏之后通过Block回调通知UI层，APP需要进行相应处理

#### 方法定义

```
- (void)stopSharing:(callback)stopSharingCallBack;
```

#### 参数描述

表 3-12 Connect 参数说明

参数	是否必须	类型	描述
stopSharingcallback	是	int	typedef void (^callback)(id t), 包含回调内容-结束投屏的原因

### 3.6.3.2.6 远程遥控功能

```
int result = [mShareServiceController getRemoteServiceStatus:^(id status) {  
}];
```

#### 接口描述

APP打开遥控器界面的时候，需要调用IdeaShareServiceController实例对象的getRemoteServiceStatus方法来获取最新的IdeaHub(服务端)状态。SDK获取到最新的状态之后，会主动通知UI层，并且通过Block回调将最新状态传回UI层，APP需要进行相应处理

#### 注意事项

此接口回调参数是一个OC类 - RemoteServiceStatus，其包含六个属性，分别对应远程操控的六个状态参数。

#### 接口定义

```
- (int)getRemoteServiceStatus:(callback)remoteServiceStatuscallback;
```

### 参数描述

表 3-13 参数说明

参数	是否必须	类型	描述
remoteServiceStatuscallback	是	RemoteServiceStatus	typedef void (^callback)(id t), 包含回调内容: 1. bool isMute 2. bool isCameraMute 3. int volume 4. bool isSpeakerMute 5. bool isChairman 6. bool isBase

### 3.6.3.2.7 打开/关闭麦克风

```
int result = [mShareServiceController setRemoteMicMute:false:^(id micOpenCallback) {}];
```

#### 接口描述

APP界面执行打开/关闭麦克风后，需要调用IdeaShareServiceController实例对象的setRemoteMicMute方法，设置ideaHub的麦克风。SDK在获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理

#### 注意事项

#### 接口定义

```
- (int)setRemoteMicMute:(bool)isMute:(callback)micMutecallback;
```

#### 参数描述

表 3-14 参数说明

参数	是否必须	类型	描述
isMute	是	bool	是否设置IdeaHub麦克风静音
micMutecallback	是	int	typedef void (^callback)(id t), 包含回调内容 – IdeaHub静音结果

### 3.6.3.2.8 调节扬声器音量

```
[mShareServiceController setSpeakerVolume:self.volumeNumber :^(id volumecallback) {}];
```

#### 接口描述

APP界面执行扬声器音量调节后，需要调用IdeaShareServiceController实例对象的setRemoteSpeakerVolume方法，设置ideaHub的扬声器音量。SDK获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理

### 注意事项

Block回调返回的int值是接口调用后大屏的音量。

### 接口定义

```
- (int)setSpeakerVolume:(int)volume:(callback)volumecallback;
```

### 参数描述

表 3-15 参数说明

参数	是否必须	类型	描述
volume	是	int	想要设置的大屏音量 ( 0-21 )
volumecallback	是	int	设置后，大屏目前的音量

### 3.6.3.2.9 打开/关闭扬声器

```
int result = [mShareServiceController setRemoteSpeakerMute:true :^(id speakerCloseCallback) {}];
```

### 接口描述

APP界面执行打开/关闭扬声器后，需要调用IdeaShareServiceController实例对象的setRemoteSpeakerMute方法，设置ideaHub的扬声器。SDK在获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理。

### 注意事项

当大屏音量为0时，屏默认为静音状态，无法在保证音量为0的状态下打开Speaker。  
建议，当大屏音量为0时，对此接口的使用加以限制。

### 接口定义

```
- (int)setRemoteSpeakerMute:(bool)isSpeakerMute:(callback)speakerMutecallback;
```

### 参数描述

表 3-16 参数说明

参数	是否必须	类型	描述
isSpeakerMute	是	bool	设置IdeaHub音响是否静音
speakerMutecallback	是	int	设置IdeaHub音响静音的结果

### 3.6.3.2.10 打开/关闭摄像头

```
int result = [mShareServiceController setRemoteCameraMute:true :^(id cameraCloseCallback) {}];
```

### 接口描述

APP界面执行打开/关闭摄像机后，需要调用IdeaShareServiceController实例对象的setRemoteCameraMute方法，设置ideaHub的摄像机。SDK在获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理

#### 注意事项

#### 接口定义

```
- (int)setRemoteCameraMute:(bool)isCameraMute:(callback)cameraMutecallback;
```

#### 参数描述

表 3-17 参数说明

参数	是否必须	类型	描述
isCameraMute	是	bool	设置IdeaHub摄像机是否开启
cameraMutecallback	是	int	设置IdeaHub摄像机开关的结果

### 3.6.3.2.11 离开会议

```
int result = [mShareServiceController leaveConference:^(id leaveConferencecallback) {}];
```

#### 接口描述

APP界面执行离开会议后，需要调用IdeaShareServiceController实例对象的leaveConference方法离开会议。SDK在获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理

#### 注意事项

#### 接口定义

```
- (int)leaveConference:(callback)leaveConferencecallback;
```

#### 参数描述

表 3-18 参数说明

参数	是否必须	类型	描述
leaveConferencecallback	是	int	离开会议的结果

### 3.6.3.2.12 结束会议

```
int result = [mShareServiceController finishConference:^(id t) {}];
```

#### 接口描述

APP界面执行结束会议后，需要调用IdeaShareServiceController实例对象的finishConference方法结束会议。SDK在获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理。

#### 注意事项

#### 接口定义

```
- (int)finishConference:(callback)finishConferencecallback;
```

#### 参数描述

表 3-19 参数说明

参数	是否必须	类型	描述
finishConferencecallback	是	int	结束会议的结果

### 3.6.3.2.13 获取客户端 SDK 版本

```
self.versionLabel.stringValue = [self->mShareServiceController getSDKVersion];
```

#### 接口描述

APP调用IdeaShareServicecontroller实例对象的getSDKVersion方法。返回值为版本信息

#### 注意事项

#### 接口定义

```
- (NSString *)getSDKVersion;
```

### 3.6.3.2.14 服务端版本查询

```
int result = [mShareServiceController getSDKLatestVersion:^(id sdkLatestVersioncallback ){}];
```

#### 接口描述

APP调用IdeaShareServicecontroller实例对象的getSDKLatestVersion方法。回调值为服务端版本信息。

#### 注意事项

此接口获取的是远端服务端的版本号，与getSDKVersion有区别

#### 接口定义

```
- (int)getSDKLatestVersion:(callback)sdkLatestVersioncallback;
```

### 3.6.3.2.15 辅助进程发送视频数据

```
[self->_dataBufferController sendVideoBuffer:sampleBuffer];
```

#### 接口描述

APP调用DataBufferController实例对象的sendVideoBuffer方法。

#### 注意事项

在辅助进程中调用，用来发送采集到的视频数据

#### 接口定义

```
- (void)sendVideoBuffer:(CMSampleBufferRef)sampleBuffer;
```

### 3.6.3.2.16 辅助进程发送音频数据

```
[self-> dataBufferController sendAudioBuffer: sampleBuffer];
```

#### 接口描述

APP调用DataBufferController实例对象的sendAudioBuffer方法。

#### 注意事项

在辅助进程中调用，用来发送采集到的音频数据。

#### 接口定义

```
- (void)sendAudioBuffer:(CMSampleBufferRef)sampleBuffer;
```

### 3.6.3.2.17 辅助进程投屏初始化

```
_dataBufferController = [[DataBufferController new] initWithGroupId:kGroup];
```

#### 接口描述

APP调用DataBufferController实例对象的initWithGroupId方法。返回值为辅助进程功能对象

#### 注意事项

在辅助进程中调用，用来获取辅助进程的功能对象。

#### 接口定义

```
- (instancetype)initWithGroupId:(NSString*)groupId;
```

### 3.6.3.2.18 接口调用结果通知

### 3.6.3.2.19 discoverCallback

#### 回调方法描述

设备发现接口Block回调

#### 回调方法定义

```
typedef void (^callback)(id t);  
@property(nonatomic, copy) callback discoverCallback;
```

#### 参数描述

表 3-20 参数说明

参数	是否必须	类型	描述
discoverCallb ack	是	DiscoverResult	OC类，设备发现结果

### 示例代码

```
int result = [mShareServiceController discover:castCode :address :port :^(id discoverCallback) {  
    self.discoverResult = (DiscoverResult*)discoverCallback;  
}];
```

## 3.6.3.2.20 connectCallback

### 回调方法描述

连接接口Block回调

### 回调方法定义

```
typedef void (^callback)(id t);  
@property(nonatomic, copy) callback connectCallback;
```

### 参数描述

表 3-21 参数说明

参数	是否必须	类型	描述
ConnectCallb ack	是	DeviceInfo	OC类，设备信息

### 示例代码

```
int connectResult = [mShareServiceController connect:addr_list :port :password :^(id connectCallback) {  
    self.deviceInfo = (DeviceInfo*)connectCallback;  
}];
```

## 3.6.3.2.21 disconnectCallback

### 回调方法描述

断开连接接口Block回调

### 回调方法定义

```
typedef void (^callback)(id t);  
@property(nonatomic, copy) callback disconnectCallback;
```

### 参数描述

表 3-22 参数说明

参数	是否必 须	类型	描述
disconnectCallback	是	int	Reason-暂无实际作用

### 示例代码

```
int result = [mShareServiceController disconnect:^(id disconnectCallback) {  
    reason = (int)disconnectCallback;  
}];
```

### 3.6.3.2.22 startSharingCallback

#### 回调方法描述

开始投屏接口Block回调

#### 回调方法定义

```
@property(nonatomic, copy) callback startSharingCallback;
```

#### 参数描述

参数	是否必须	类型	描述
startSharingCallback	是	int	Result - 暂无实际作用

#### 示例代码

```
int result = [mShareServiceController startSharing:^(id startSharingCallback) {  
}];
```

### 3.6.3.2.23 stopSharingCallback

#### 回调方法描述

结束投屏接口Block回调

#### 回调方法定义

```
@property(nonatomic, copy) callback stopSharingCallback;
```

#### 参数描述

表 3-23 参数说明

参数	是否必须	类型	描述
stopSharingCallback	是	int	Reason -暂无实际作用

#### 示例代码

```
int result = [mShareServiceController stopSharing:^(id stopSharingCallback) {  
    reason = (int)stopSharingCallback;  
}];
```

### 3.6.3.2.24 remoteServiceStatusCallback

#### 回调方法描述

远程控制服务状态接口Block回调

#### 回调方法定义

```
@property(nonatomic, copy) callback remoteServiceStatusCallback;
```

#### 参数描述

**表 3-24 参数说明**

参数	是否必须	类型	描述
RemoteServiceStatusCallback	是	RemoteServiceStatus	OC类，包含大屏侧的六个遥控状态

**示例代码**

```
int result = [mShareServiceController getRemoteServiceStatus:^(id status) {  
    self.remoteserviceStatus = (RemoteServiceStatus *)status;  
}];
```

**3.6.3.2.25 micMuteCallback****回调方法描述**

设置麦克风静音接口Block回调

**回调方法定义**

```
@property(nonatomic, copy) callback micMuteCallback;
```

**参数描述****表 3-25 参数说明**

参数	是否必须	类型	描述
micMuteCallback	是	int	设置麦克风静音的结果

**示例代码**

```
int result = [mShareServiceController setRemoteMicMute:false:^(id micMuteCallback) {  
}];
```

**3.6.3.2.26 speakerVolumeCallback****回调方法描述**

大屏扬声器音量接口Block回调

**回调方法定义**

```
@property(nonatomic, copy) callback speakerVolumeCallback;
```

**参数描述**

**表 3-26 参数说明**

参数	是否必须	类型	描述
speakerVolumeCallback	是	int	设置大屏扬声器音量结果

**示例代码**

```
[mShareServiceController setSpeakerVolume:self.volumeNumber:^(id speakerVolumecallback) { }];
```

**3.6.3.2.27 speakerMuteCallback****回调方法描述**

大屏扬声器静音接口Block回调

**回调方法定义**

```
@property(nonatomic, copy) callback speakerMuteCallback;
```

**参数描述****表 3-27 参数说明**

参数	是否必须	类型	描述
speakerMuteCallback	是	int	Result -暂无实际作用

**示例代码**

```
int result = [mShareServiceController setRemoteSpeakerMute:true:^(id speakerMuteCallback) { }];
```

**3.6.3.2.28 cameraMuteCallback****回调方法描述**

大屏摄像头开关接口Block回调

**回调方法定义**

```
@property(nonatomic, copy) callback speakerMuteCallback;
```

**参数描述**

**表 3-28** 参数说明

参数	是否必须	类型	描述
cameraMuteCallback	是	int	Result -暂无实际作用

**示例代码**

```
int result = [mShareServiceController setRemoteCameraMute:true:^(id cameraMuteCallback) {  
}];
```

**3.6.3.2.29 leaveConfCallback****回调方法描述**

离开会议接口Block回调

**回调方法定义**

```
@property(nonatomic, copy) callback leaveConfCallback;
```

**参数描述****表 3-29** leaveConfCallback 参数说明

参数	是否必须	类型	描述
leaveConfCallback	是	int	离开会议的结果

**示例代码**

```
int result = [mShareServiceController leaveConference:^(id leaveConfCallback) {  
}];
```

**3.6.3.2.30 finishConfCallback****回调方法描述**

结束会议接口Block回调

**回调方法定义**

```
@property(nonatomic, copy) callback finishConfCallback;
```

**参数描述****表 3-30** 参数说明

参数	是否必须	类型	描述
finishConfCallback	是	int	结束会议的结果

### 示例代码

```
int result = [mShareServiceController finishConference:^(id t) {  
}];
```

#### 3.6.3.2.31 sdkLatestVersionCallback

##### 回调方法描述

服务端版本查询接口Block回调

##### 回调方法定义

```
@property(nonatomic, copy) callback sdkLatestVersionCallback;
```

##### 参数描述

表 3-31 参数说明

参数	是否必须	类型	描述
sdkLatestVersionCallback	是	NSString	服务端版本号

### 示例代码

```
int result = [mShareServiceController getSDKLatestVersion:^(id t) {  
}];
```

#### 3.6.3.2.32 tlsCallback

##### 回调方法描述

TLS设置接口Block回调

##### 回调方法定义

```
typedef void (^callback)(id t);  
@property(nonatomic, copy) callback tlsCallback;
```

##### 参数描述

表 3-32 参数说明

参数	是否必须	类型	描述
tlsCallback	是	int	OC类, TLS验证失败通知

### 示例代码

```
int result = [mShareServiceController setTls: tlsInfo:^(id tlsCallback) {  
    int res = (int)tlsCallback;  
}];
```

#### 3.6.3.3 设备推送通知

### 3.6.3.3.1 onRemoteServiceStatusChangedNotify

#### 回调方法描述

设备推送服务端遥控状态通知

#### 回调方法定义

```
- (void)onRemoteServiceStatusChangedNotify:(RemoteServiceStatus *) remoteServiceStatus : (int) changeCount;
```

#### 参数描述

表 3-33 参数说明

参数	是否必须	类型	描述
remoteServiceStatus	是	RemoteServiceStatus	OC类，服务端遥控状态
changeCount	是	int	遥控状态变化指数，根据isMute, isCameraMute, volume, isSpeakerMute, isChairman的顺序从左到右，代表五位的二进制，全部默认为0，当有某个状态改变了，则设为1，代表改变的状态是哪一个。例如，参数为16的时候，二进制为10000，代表isMute发生了改变了，通过remoteServiceStatus参数去获取改变后的状态

#### 示例代码

```
- (void) onRemoteServiceStatusChangedNotify : (RemoteServiceStatus *)remoteServiceStatus : (int) changeCount
{
    switch (changeCount) {
        case 16:
        {
            NSLog(@" onRemoteMicStatusNotify:isMic = [%d] ",remoteServiceStatus.isMute);
            break;
        }
    }
}
```

### 3.6.3.3.2 OnDevConfStatusNotify

#### 回调方法描述

设备推送辅流和会议状态通知

#### 回调方法定义

```
- (void)OnDevConfStatusChangeNotify:(IDEA_SHARE_APP_CONF_STATUS)confStatus:
(IDEA_SHARE_APP_AUX_STATUS)auxStatus;
```

#### 参数描述

表 3-34 参数说明

参数	是否必须	类型	描述
confStatus	是	IDEA_SHARE_APP_CONF_STATUS	IDEA_SHARE_CONF_IDLE 0 为空闲状态 IDEA_SHARE_CONF_BUSY 1 为会议状态
auxStatus	是	IDEA_SHARE_APP_AUX_STATUS	IDEA_SHARE_AUX_IDLE 0 为 空闲状态 IDEA_SHARE_AUX_BUSY 1 为 辅流发送状态

### 示例代码

```
- (void) OnDevConfStatusChangeNotify : (IDEA_SHARE_APP_CONF_STATUS) confStatus :  
(IDEA_SHARE_APP_AUX_STATUS) auxStatus  
{  
    if (confStatus == 1) {  
  
    } else {  
  
    }  
}
```

### 3.6.3.3.3 OnShareStatusChangeNotify

#### 回调方法描述

设备推送共享状态通知

#### 回调方法定义

```
- (void)OnShareStatusChangeNotify:(IDEA_SHARE_APP_SERVER_STATUS)status:  
(IDEA_SHARE_ERROR_ID)reason;
```

#### 参数描述

表 3-35 参数说明

参数	是否必须	类型	描述
status	是	IDEA_SHARE_APP_STATUS	设备状态通知枚举值
reason	是	IDEA_SHARE_ERROR_ID	错误码

### 示例代码

```
- (void) OnShareStatusChangeNotify : (IDEA_SHARE_APP_SERVER_STATUS) status :  
(IDEA_SHARE_ERROR_ID) reason  
{  
    if (status == 3)  
    {  
    }  
}
```

```
    } else if (status == 2) {  
        }  
    }
```

### 3.6.3.4 错误通知

#### 3.6.3.4.1 OnErrorNotify

##### 回调方法描述

SDK向上层推送错误通知

##### 回调方法定义

```
- (void)OnErrorNotify:(IDEA_SHARE_APP_ERR)errCode;
```

##### 参数描述

表 3-36 OnErrorNotify 参数说明

参数	是否必须	类型	描述
errCode	是	int	错误码

##### 示例代码

```
- (void) OnErrorNotify : (IDEA_SHARE_APP_ERR) errCode  
{  
    switch (errCode) {  
        case IDEA_SHARE_ERROR_SUCCESS:  
        {  
            NSLog(@"Error Code is 0, Success");  
            break;  
        }  
        case IDEA_SHARE_ERROR_CONNECT_FAIL:  
        {  
            break;  
        }  
    }  
}
```

### 3.6.4 附录

#### 3.6.4.1 枚举定义

##### 3.6.4.1.1 LOG\_LEVEL

枚举值	对应值	描述
IDEA_LEVEL_ERROR	0	错误级别
IDEA_LEVEL_WARNING	1	警告级别

枚举值	对应值	描述
IDEA_LEVEL_INFO	2	信息级别
IDEA_LEVEL_DEBUG	3	调试级别

#### 3.6.4.1.2 IDEA\_SHARE\_APP\_DEFINAION\_QUAITY

枚举值	对应值	描述
QUALITY_FLUENCY_FIRST	0	流畅优先, 默认值
QUAITY_CLEAR_FIRST	1	清晰优先
QUAITY_ADAPTIVE	2	自适应

#### 3.6.4.1.3 IDEA\_SHARE\_APP\_CONF\_STATUS

枚举值	对应值	描述
IDEA_SHARE_CONF_IDLE	0	空闲
IDEA_SHARE_CONF_BUSY	1	忙

#### 3.6.4.1.4 IDEA\_SHARE\_APP\_AUX\_STATUS

枚举值	对应值	描述
IDEA_SHARE_AUX_IDLE	0	空闲
IDEA_SHARE_AUX_BUSY	1	忙

#### 3.6.4.1.5 IDEA\_SHARE\_APP\_SERVER\_STATUS

枚举值	对应值	描述
SERVER_STATUS_CONNECT	0	连接状态

枚举值	对应值	描述
SERVER_STATUS_SHARE	1	投屏共享状态
SERVER_STATUS_STOP_SHARE	2	停止投屏共享状态
SERVER_STATUS_DISCONNECT	3	断开连接状态

### 3.6.4.1.6 IDEA\_SHARE\_APP\_ERR

枚举值	对应值	描述
DISCOVER_ERROR	0	设备发现错误
CONNECT_ERROR	1	连接错误
DISCONNECT_ERROR	2	断开连接错误
SHARE_PLAY_ERROR	3	启动投屏错误
SHARE_STOP_ERROR	4	停止投屏错误
GET_REMOTE_SERVICE_STATUS_ERROR	5	遥控器参数查询结果错误
SET_REMOTE_MIC_MUTE_ERROR	6	打开\关闭麦克风结果错误
SET_REMOTE_SPEAKER_VOLUME_ERROR	7	调节扬声器音量结果错误
SET_REMOTE_SPEAKER_MUTE_ERROR	8	打开\关闭扬声器结果错误
SET_REMOTE_CAMERA_MUTE_ERROR	9	打开\关闭摄像机结果错误
LEAVE_CONFERENCE_ERROR	10	离开会议结果错误
END_CONFERENCE_ERROR	11	结束会议结果错误
GET_SDK_LATEST_VERSION_ERROR	12	获取SDK最新版本错误
PROCESS_CONNECTING_ERROR	13	正在处理连接请求
IN_CONNECTION_ERROR	14	客户端已处于连接
PROJECTION_CODE_ERROR	15	投影码错误 数字+字母组合

枚举值	对应值	描述
AUTH_ERROR	16	鉴权失败
CLIENT_IS_LOCKED_ERROR	17	客户端被锁定
CONNECT_BUSY_ERROR	18	终端忙碌
CONNECT_UPDATING_ERROR	19	终端正在升级
WEB_DENY_ERROR	20	WEB被禁用
NO_COMMON_DATA_CAP_ERROR	21	和客户端没有辅流公共能力
GET_TOKEN_FAILD_ERROR	22	获取辅流令牌失败
LOW_BANDWIDTH_ERROR	23	带宽过低，无法发送辅流
CONF_NO_DATA_CAP_ERROR	24	会议无辅流能力
MULTICONNECT_BUSY_ERROR	25	终端正在处理辅流发送，暂且不接受辅流发送该请求
EUA_AUTH_ERROR	26	在线投屏鉴权失败
EUA_IS_LOCKED_ERROR	27	在线投屏已经锁定

### 3.6.4.1.7 IDEA\_SHARE\_APP\_TLS\_VERIFY\_MODE

枚举值	对应值	描述
TLS_VERIFY_MODE_NONE	0	不验证
TLS_VERIFY_MODE_SERVER	1	服务端验证
TLS_VERIFY_MODE_CLIENT	2	客户端验证
TLS_VERIFY_MODE_BOTH	3	双向验证
TLS_VERIFY_MODE_BUTT	4	( 无意义 )

### 3.6.4.2 OC 类定义

### 3.6.4.2.1 IdeaShareServiceAppInfo

参数名	类型	描述
appInfo	NSString	App描述信息，长度限制为128
exePath	NSString	执行路径，长度限制为128

### 3.6.4.2.2 IdeaShareServiceLogInfo

日志对象参数

参数名	类型	描述
path	NSString	日志路径，最大长度限制225
logLevel	枚举值 <a href="#">2.6.4.1.1</a>	日志级别：错误，警告，信息，调试
enable	bool	是否开启日志

### 3.6.4.2.3 SharePolicy

投放音频策略参数

参数名	类型	描述
shareQuality	<a href="#">2.6.4.1.2</a>	清晰度
isShareAudio	bool	0表示不共享音频，1表示共享音频
isCompatibility	bool	NO表示不兼容，YES表示兼容

### 3.6.4.2.4 DiscoverResult

设备发现结果

参数名	类型	描述
result	int	设备发现结果 0 成功 非0 失败
password	NSString	设备连接码
addressList	NSMutableArray	设备IP地址列表

### 3.6.4.2.5 DeviceInfo

远端设备信息

参数名	类型	描述
deviceAddress	NSString	终端地址 ( IP地址 or 域名 )
deviceName	NSString	终端名(会场名)
isSupportRemote	bool	终端侧是否支持遥控器 VOS_TRUE:支持,VOS_FALSE:不支持

### 3.6.4.2.6 RemoteServiceStatus

远端设备状态

参数名	类型	描述
isMute	bool	麦克风是否关闭 YES为关闭, NO为打开
isCameraMute	bool	摄像头是否关闭 YES为关闭, NO为打开
volume	int	扬声器音量值 范围 0 ~21
isSpeakerMute	bool	扬声器是否关闭 YES为关闭, NO为打开
isChairman	bool	在会议中是否是主席 YES为主席, NO为非主席
isBase	bool	是否是标准版 YES为标准版本不带摄像头, NO为非标准版本带摄像头

### 3.6.4.2.7 IdeaShareServiceTlsInfo

TLS信息

参数名	类型	描述
caCertpath	NSString	ca证书地址
clientCertpath	NSString	客户端证书地址
verifyMode	IDEA_SHARE_APP_TLS_VERIFY_MODE	验证模式

### 3.6.4.3 SDK 接口头文件定义

#### 3.6.4.3.1 IdeaShareEventHandler.h

协议形式的代理回调，IdeaShareServiceController的实例对象有一个notifyDelegate属性，应用程序需要为此属性赋值来注册回调，建议赋值为self

#### 3.6.4.3.2 IdeaShareServiceController.h

SDK API 和 Block回调定义文件

#### 3.6.4.4 权限说明

SDK 已申请网络权限，需要用户授权。

### 3.7 错误码参考

#### 3.7.1 错误码

参照枚举值IDEA\_SHARE\_APP\_ERR错误码定义

# 4 MAC SDK

## 4.1 概述

投屏二次开发Mac SDK 提供了一套完整的接口集合，开发者可以通过调用ideaShare SDK（以下简称SDK）开放的API，快速集投屏能力，包括启动投屏、遥控器、投放音频等。

注：macOS系统下，Framework格式的SDK包里包含有苹果开发自带的属性列表文件Info.plist，此文件为一种XML格式，内含<!DOCTYPE>标签以及标签中的DTD网址<http://www.apple.com/DTDs/PropertyList-1.0.dtd>。

## 4.2 变更记录

表 4-1 变更记录

日期	版本	变更内容
2021-03-03	1.0.0	初稿完成
2021-04-27	2.0.0	接口修订，SDK格式由dylib更改为framework

## 4.3 快速入门

本文面向有一定Objective-C和Xcode开发能力的开发者，Demo编译流程为例，介绍如何使用SDK进行二次开发

### 4.3.1 预置条件

支持SMC 2.0 , SMC 3.0, 云会议组网

## 4.3.2 开发环境准备

表 4-2 环境要求

名称	要求
Xcode版本	Xcode11
Clang	随Xcode版本
BlackHole	建议通过IdeaShare App进行安装

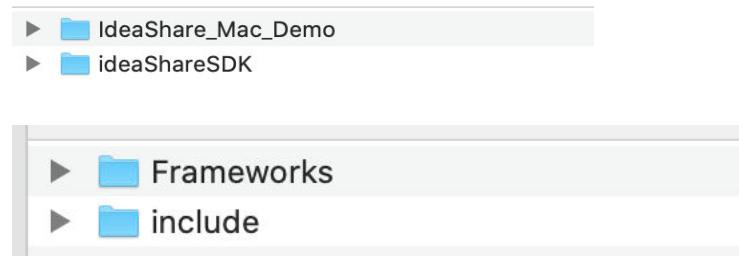
## 4.3.3 SDK 快速集成

### 步骤1 解压SDK和Demo

----结束

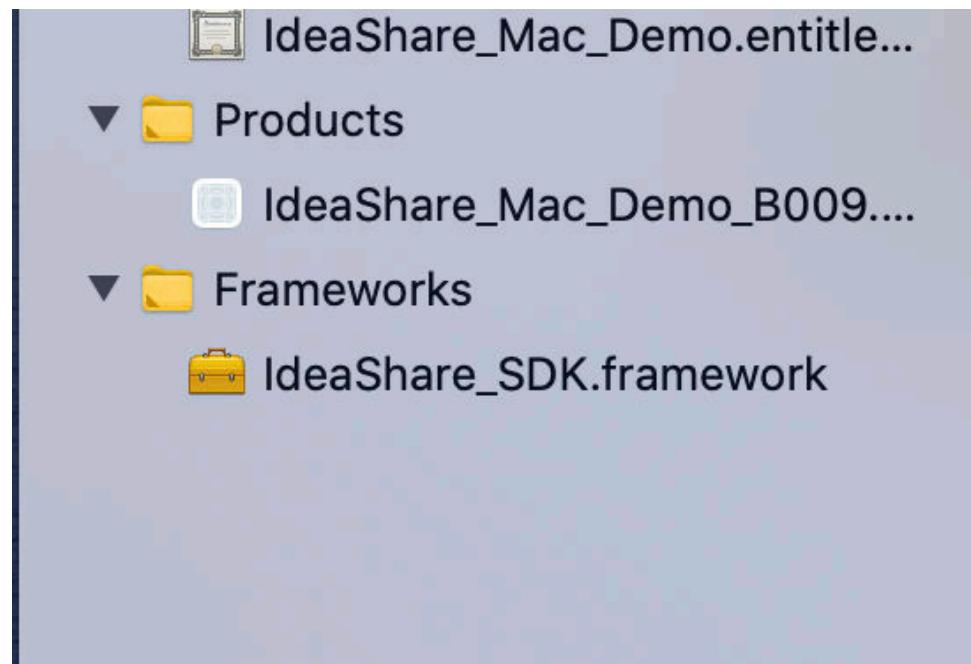
下载解压软件包，得到如下几个文件：

图 4-1 文件目录



1. 拷贝ideaShareSDK中Frameworks目录下的framework拷贝到IdeaShare\_Mac\_Demo项目中 Products目录下，如图2所示

图 4-2 导入 SDK



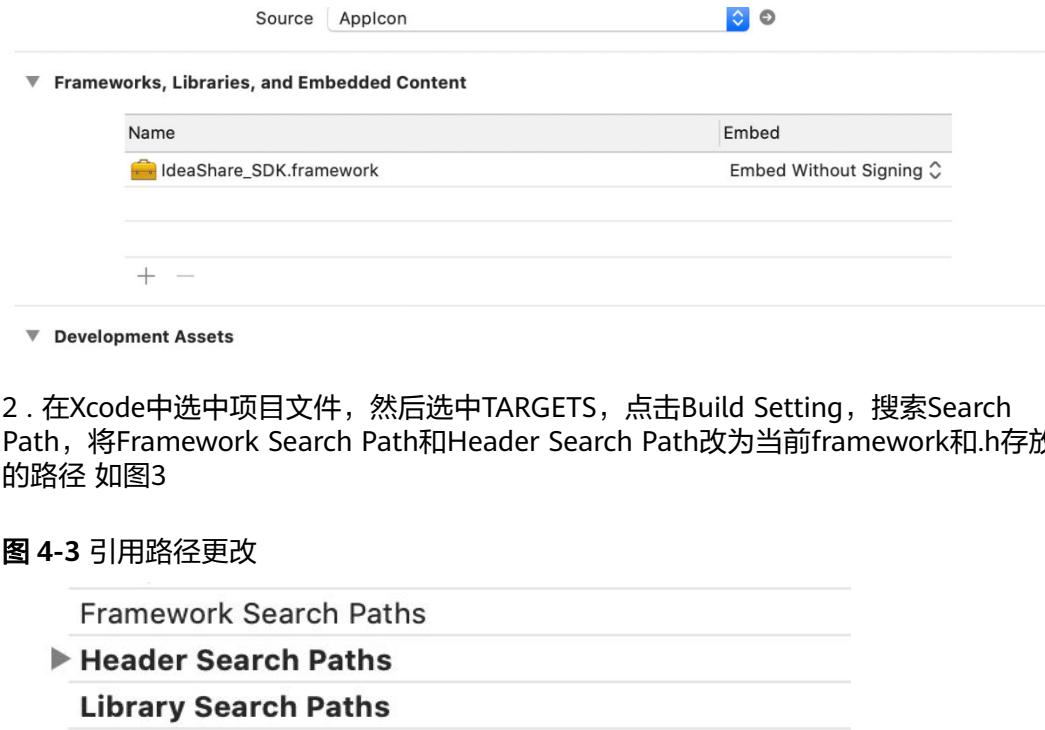


图 4-3 引用路径更改

Framework Search Paths  
► **Header Search Paths**  
Library Search Paths

3、在Xcode中选中项目文件，然后选中TARGETS，点击Build Setting，参照下面的图片进行设置（图片中的粗体字项为修改过的选项）

The screenshot shows the 'General' tab of the Xcode project settings for a target named 'IdeaShare\_Mac\_De...'. The 'TARGETS' section is selected. In the 'Architectures' section, the 'Build Active Architecture Only' option is checked. Other options include 'Setting', 'Additional SDKs', 'Architectures', 'Base SDK', 'Excluded Architectures', and 'Supported Platforms'. The 'Assets' and 'Build Locations' sections are also visible.

General    Signing & Capabilities    Resource Tags    Info

PROJECT    IdeaShare\_Mac\_De...

TARGETS    IdeaShare\_Mac\_De...

Architectures

- Setting
- Additional SDKs
- Architectures
- Base SDK
- Build Active Architecture Only**
- Excluded Architectures
- Supported Platforms

Assets

- Setting
- Asset Pack Manifest URL Prefix
- Embed Asset Packs In Product Bundle**
- Enable On Demand Resources
- On Demand Resources Initial Install Tags
- On Demand Resources Prefetch Order

Build Locations

- Setting
- Build Products Path
- Intermediate Build Files Path
- Precompiled Headers Cache Path

▼ Deployment	
Setting	IdeaShare_Mac_Demo_B006
Additional Strip Flags	
Alternate Install Group	staff
Alternate Install Owner	huawei
Alternate Install Permissions	u+w,go-w,a+rX
Alternate Permissions Files	
<b>COMBINE_HIDPI_IMAGES</b>	Yes ▲
Deployment Location	No ▲
Deployment Postprocessing	No ▲
Development Assets	
DriverKit Deployment Target	DriverKit 20.2 ▲
Install Group	staff
Install Owner	huawei
Install Permissions	u+w,go-w,a+rX
Installation Build Products Location	/tmp/IdeaShare_Mac_Demo_B006.dst
Installation Directory	/Applications
PRODUCT_DEFINITION_PLIST	
Resources Targeted Device Family	
Skip Install	No ▲
Strip Debug Symbols During Copy	No ▲
Strip Linked Product	Yes ▲
Strip Style	All Symbols ▲
Strip Swift Symbols	Yes ▲
Targeted Device Families	△
iOS Deployment Target	iOS 14.3 ▲
<b>macOS Deployment Target</b>	macOS 10.13 ▲
tvOS Deployment Target	tvOS 14.3 ▲
watchOS Deployment Target	watchOS 7.2 ▲
mac-O Type	
Order File	Executable ▾
Other Librarian Flags	
Other Linker Flags	
Path to Link Map File	<Multiple values>
Debug	build/IdeaShare_Mac_Demo_B006.build/Debug/IdeaShare_M...
Release	build/IdeaShare_Mac_Demo_B006.build/Release/IdeaShare_...
Perform Single-Object Prelink	No ▲
Prelink libraries	
Preserve Private External Symbols	No ▲
Quote Linker Arguments	Yes ▲
Re-Exported Framework Names	
Re-Exported Library Names	
Re-Exported Library Paths	
Runpath Search Paths	@executable_path/../Frameworks @loader_path
Separately Edit Symbols	No ▲
Single-Object Prelink Flags	
Symbol Ordering Flags	
Unexported Symbols File	
Warning Linker Flags	
Write Link Map File	No ▲
▼ Localization	
Setting	IdeaShare_Mac_Demo_B006
Localized String Macro Names	NSLocalizedString CFCopyLocalizedString
Localized String Swift UI Support	Yes ▲
▼ Packaging	
Setting	IdeaShare_Mac_Demo_B006
Convert Copied Files	No ▲
Create Info.plist Section in Binary	No ▲
Defines Module	No ▲
Don't Force Info.plist Generation	No ▲
	CREATE_INFOPLIST_SI

▼ Packaging

Setting	IdeaShare_Mac_Demo_B006
Convert Copied Files	No ◊
Create Info.plist Section in Binary	No ◊
Defines Module	No ◊
Don't Force Info.plist Generation	No ◊
Executable Extension	
Executable Prefix	
Expand Build Settings in Info.plist File	Yes ◊
Force Package Info Generation	Yes ◊
Framework Version	A
<b>Info.plist File</b>	<b>IdeaShare_Mac_Demo/Info.plist</b>
Info.plist Other Preprocessor Flags	
Info.plist Output Encoding	same-as-input ◊
Info.plist Preprocessor Definitions	
Info.plist Preprocessor Prefix File	
Module Map File	
Preprocess Info.plist File	No ◊
Preserve HFS Data	No ◊
Private Headers Folder Path	IdeaShare_Mac_Demo_B009.app/Contents/PrivateHeaders
Private Module Map File	
Process Header Files	No ◊
<b>Product Bundle Identifier</b>	<b>hw.packTEst</b>
Product Module Name	IdeaShare_Mac_Demo_B009
<b>Product Name</b>	<b>IdeaShare_Mac_Demo_B009</b>
Property List Output Encoding	same-as-input ◊
Public Headers Folder Path	IdeaShare_Mac_Demo_B009.app/Contents/Headers
Strings File Output Encoding	UTF-16 ◊
Wrapper Extension	app

▼ Search Paths	
Setting	IdeaShare_Mac_Demo_B006
Always Search User Paths (Deprecated)	No ▾
<b>Framework Search Paths</b>	/Users/huawei/Desktop/LFY/IdeaShare_Mac_Demo/Idea...
<b>Header Search Paths</b>	<Multiple values>
Debug	build/Debug/include /Users/huawei/Desktop/LFY/IdeaSh...
Release	build/Release/include /Users/huawei/Desktop/LFY/IdeaS...
<b>Library Search Paths</b>	
Rez Search Paths	
Sub-Directories to Exclude in Recursive Searches	*.nib *.lproj *.framework *.gch *.xcode* *.xcassets (*) .DS_...
Sub-Directories to Include in Recursive Searches	
System Framework Search Paths	
System Header Search Paths	
Use Header Maps	Yes ▾
User Header Search Paths	
▼ Signing	
Setting	IdeaShare_Mac_Demo_B006
<b>Code Signing Entitlements</b>	IdeaShare_Mac_Demo/IdeaShare_Mac_Demo.entitlements
Code Signing Identity	Apple Development ▾
Code Signing Inject Base Entitlements	Yes ▾
<b>Code Signing Style</b>	Automatic ▾
<b>Development Team</b>	Loading... ▾
Enable Hardened Runtime	No ▾
<b>Other Code Signing Flags</b>	--deep
Provisioning Profile	Automatic ▾
▼ Testing	
Setting	IdeaShare_Mac_Demo_B006
Test Host	
Treat missing baselines as test failures	No ▾

Generate Position-Dependent Code	No ▾
Inline Methods Hidden	Yes ▾
Instrument Program Flow	No ▾
Kernel Development Mode	No ▾
Link-Time Optimization	No ▾
Make Strings Read-Only	Yes ▾
No Common Blocks	Yes ▾
<b>Optimization Level</b>	<b>None [-O0] ▾</b>
Optimization Profile File	/Users/huawei/Desktop/LFY/IdeaShare_Mac_Demo/Optimizat...
Relax IEEE Compliance	No ▾
Statics are Thread-Safe	Yes ▾
Symbols Hidden by Default	Yes ▾
Unroll Loops	No ▾
Use Optimization Profile	No ▾
<b>▼ Apple Clang - Custom Compiler Flags</b>	
Setting	 IdeaShare_Mac_Demo_B006
Other C Flags	
Other C++ Flags	
Other Warning Flags	
<b>▼ Apple Clang - Language</b>	
Setting	 IdeaShare_Mac_Demo_B006
'char' Type Is Unsigned	No ▾
Allow 'asm', 'inline', 'typeof'	Yes ▾
C Language Dialect	gnu11 ▾
CodeWarrior/MS-Style Inline Assembly	Yes ▾
Compile Sources As	According to File Type ▾
Enable Linking With Shared Libraries	Yes ▾
Enable Trigraphs	No ▾
Generate Floating Point Library Calls	No ▾
<b>Increase Sharing of Precompiled Headers</b>	<b>No ▾</b>
<b>Precompile Prefix Header</b>	<b>Yes ▾</b>

▼ Apple Clang - Language	
Setting	IdeaShare_Mac_Demo_B006
'char' Type Is Unsigned	No ▾
Allow 'asm', 'inline', 'typeof'	Yes ▾
C Language Dialect	gnu11 ▾
CodeWarrior/MS-Style Inline Assembly	Yes ▾
Compile Sources As	According to File Type ▾
Enable Linking With Shared Libraries	Yes ▾
Enable Trigraphs	No ▾
Generate Floating Point Library Calls	No ▾
<b>Increase Sharing of Precompiled Headers</b>	<b>No ▾</b>
<b>Precompile Prefix Header</b>	<b>Yes ▾</b>
<b>Prefix Header</b>	<b>/Users/huawei/Desktop/LFY/IdeaShare_Mac_Demo/Idea...</b>
Recognize Builtin Functions	Yes ▾
Recognize Pascal Strings	Yes ▾
Short Enumeration Constants	No ▾
Use Standard System Header Directory Searching	Yes ▾

▼ Apple Clang - Language - C++	
Setting	IdeaShare_Mac_Demo_B006
C++ Language Dialect	GNU++14 [-std=gnu++14] ▾
C++ Standard Library	libc++ (LLVM C++ standard library with C++11 support) ▾
Destroy Static Objects	Yes ▾
Enable C++ Exceptions	Yes ▾
Enable C++ Runtime Types	Yes ▾

▼ Apple Clang - Language - Modules	
Setting	IdeaShare_Mac_Demo_B006
Allow Non-modular Includes In Framework Modules	No ▾
Disable Private Modules Warnings	No ▾
Enable Clang Module Debugging	Yes ▾
Enable Modules (C and Objective-C)	Yes ▾

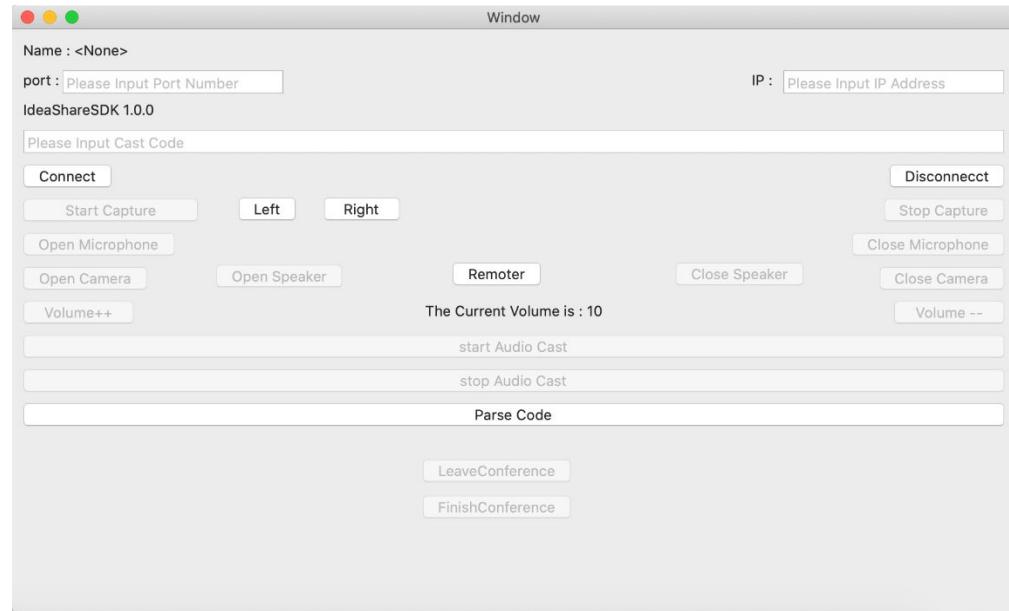
▼ Asset Catalog Compiler - Options	
Setting	IdeaShare_Mac_Demo_B006
<b>Asset Catalog App Icon Set Name</b>	<b>AppIcon</b>
Asset Catalog Launch Image Set Name	
Asset Catalog Other Flags	
Build Active Resources Only	Yes ▾
Enable Incremental Distill	No ▾
<b>Global Accent Color Name</b>	<b>AccentColor</b>
Include Asset Localizations in Info.plist	Yes ▾
Include Sticker Content	No ▾
Optimization	△
Show Notices	Yes ▾
Show Warnings	Yes ▾
Standalone Icon File Behavior	Default ▾
Sticker Pack Identifier Prefix	
Stickers Icon Role	None ▾
Widget Background Color Name	

▼ Interface Builder Storyboard Compiler - Options	
Setting	IdeaShare_Mac_Demo_B006
Auto-Activate Custom Fonts	Yes ▾
Default Module	IdeaShare_Mac_Demo_B009
Flatten Compiled Storyboard Files	Yes ▾
Other Storyboard Compiler Flags	
Show Errors	Yes ▾
Show Notices	Yes ▾
Show Warnings	Yes ▾

4、在Xcode下编译运行Demo，如图4所示

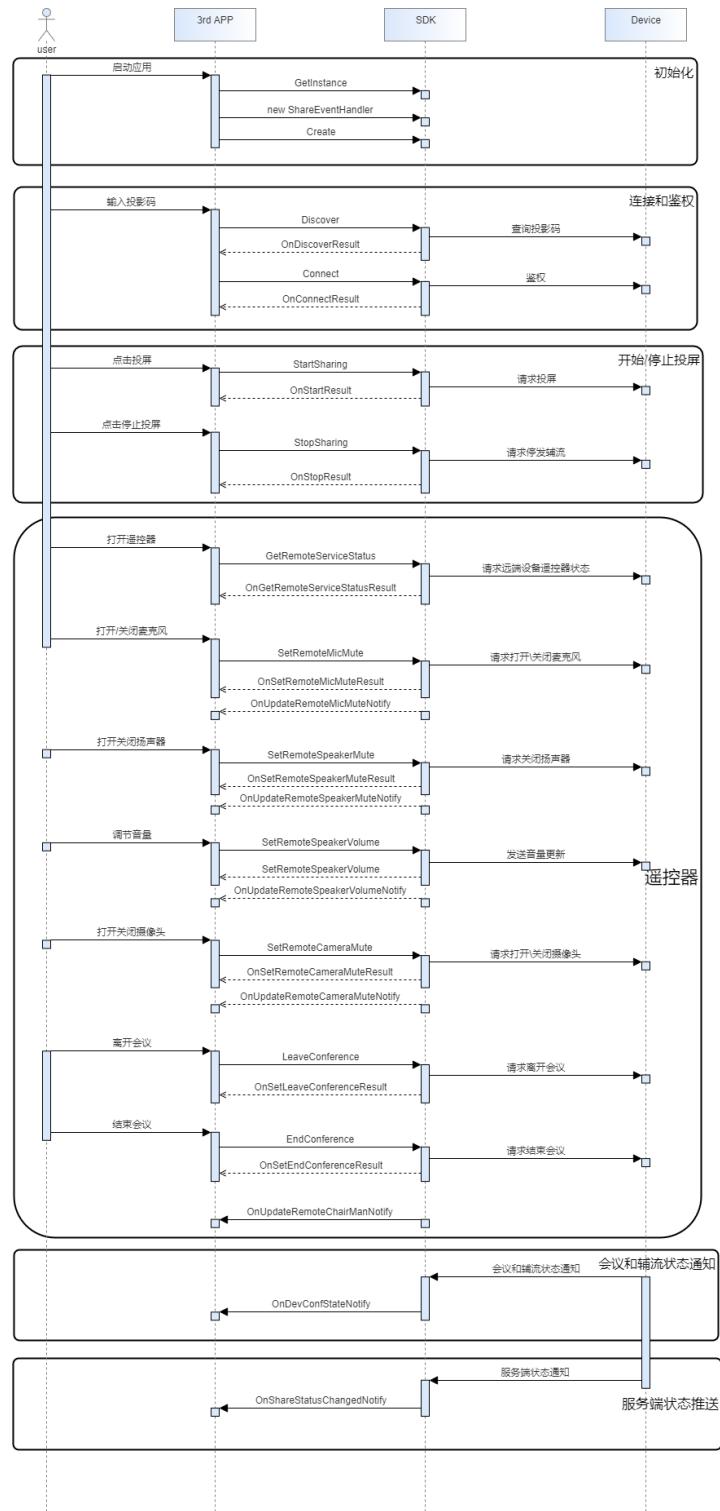
图 4-4 编译运行 demo



## 4.4 典型场景

## 4.4.1 典型场景接口调用概览

图 4-5 接口调用流程



1、投屏APP 启动前必须调用初始化接口

2、大部分接口调用结果通过异步回调实现，如xxxResult， 服务端通知事件格式  
xxxNotify

3、App使用SDK Framework需要给Framework进行重新签名， codesign -f -s，并在  
Build Setting中添加Other Code Signing Flags标签 --deep。

## 4.4.2 场景 1： 初始化

### 描述

使用投屏SDK业务组件时，需要先完成SDK的基础组件的初始化。

### 业务流程

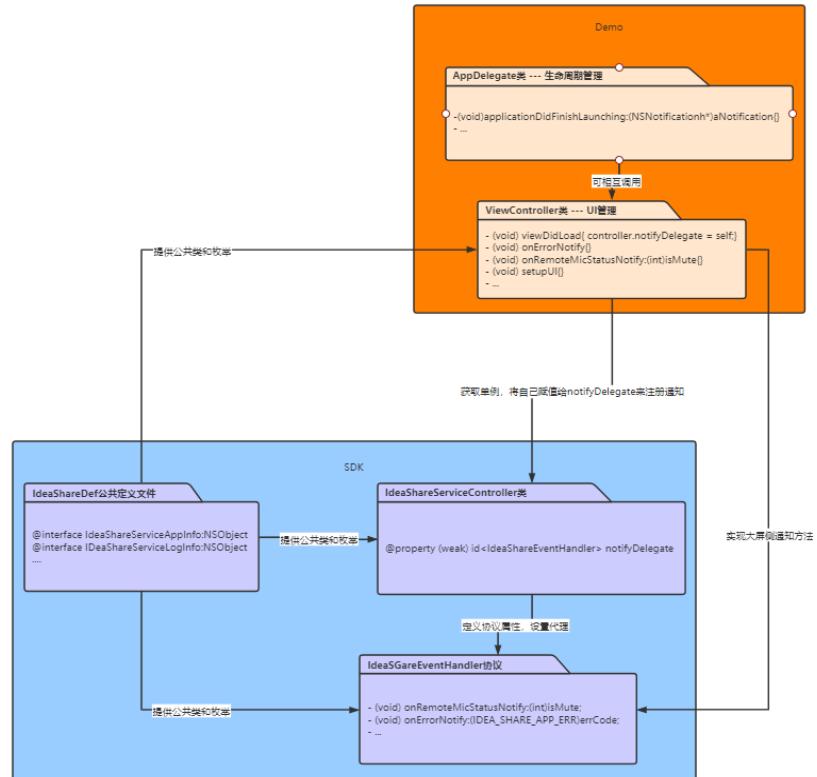
#### 步骤1 接口调用

----结束

1. 调用[IdeaShareServiceController shareInstance] 获取业务调用Controller  
mShareServiceController = [IdeaShareServiceController shareInstance];
2. 初始化日志  
int logRet = [mShareServiceController setLog:logInfo];
3. 初始化SDK  
Int ret = [mShareServiceController create:appInfo];
4. 自定义事件回调协议，在ViewController类初始化中，给业务调用Controller赋  
值，然后可在ViewController类中实现IdeaShareEventHandler的方法，对大屏侧  
发来的消息进行处理并修改UI。

```
- (void)viewDidLoad {  
    [super viewDidLoad];  
    if ([IdeaShareServiceController respondsToSelector:@selector(shareInstance)]) {  
        mShareServiceController = [IdeaShareServiceController shareInstance];  
    } else {  
        NSLog(@"ResponseToSelector_Failed");  
    }  
    mShareServiceController.notifyDelegate = (id)self;  
    IdeaShareServiceLogInfo *logInfo = [[IdeaShareServiceLogInfo alloc]init];  
    logInfo.enable = YES;  
    logInfo.level = 3;  
    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSCachesDirectory, NSUserDomainMask, YES);  
    NSString *baseDir = paths.firstObject;  
    NSString *logsDirectory = [baseDir stringByAppendingPathComponent:@"Logs"];  
    logInfo.path = logsDirectory;  
    int logRet = [mShareServiceController setLog:logInfo];  
    NSLog(@"Set Log Result is %d",logRet);  
    const IdeaShareServiceAppInfo *appInfo = [[IdeaShareServiceAppInfo alloc]init];  
    appInfo.appInfo = @"aa";  
    appInfo.exePath = @"bb";  
    NSLog(@"AppInfo is %@",appInfo);  
    NSLog(@"EXEPATH length is %lu", (unsigned long)appInfo.exePath.length);  
    int ret = [mShareServiceController create:appInfo];  
    NSLog(@"Initialization Result is %d",ret);  
    dispatch_async(dispatch_get_main_queue(), ^{  
        self.versionLabel.stringValue = [self->mShareServiceController getSDKVersion];  
    });  
    [[[NSWorkspace sharedWorkspace] notificationCenter] addObserver:self selector:@selector(sleepReaction) name:NSNotificationCenterWillSleepNotification object:nil];  
    [self setupUI];  
    You, 2 months ago • Mac Demo Upload  
  
- (void)onDevConfStatusChangeNotify : (IDEA_SHARE_APP_CONF_STATUS) confStatus : (IDEA_SHARE_APP_AUX_STATUS) auxStatus {  
    NSLog(@"Delegate Conference Status is %ld, Aux Status is %ld", (long)confStatus, (long)auxStatus);  
    if (confStatus == 1) {  
        dispatch_async(dispatch_get_main_queue(), ^{  
            self.leaveConferenceButton.enabled = YES;  
        });  
    } else {  
        dispatch_async(dispatch_get_main_queue(), ^{  
            self.leaveConferenceButton.enabled = NO;  
            self.finishConferenceButton.enabled = NO;  
        });  
    }  
}
```

5.类之间关系如下图所示：



## 注意事项

`IdeaShareEventHandler` 是用户自定义回调协议，需给`IdeaShareServiceController`的代理`notifyDelegate`赋值， 实现`IdeaShareEventHandler`所有的回调方法。

### 4.4.3 场景 2： 查询投影码

#### 描述

用户输入投影码后，返回连接设备IP和密码

#### 业务流程

##### 步骤1 接口调用

----结束

1. 获取EUA地址和端口号，EUA地址为空则传入空字符串，端口号0
2. 调用 `IdeaShareServiceController`的`discover`接口
3. 处理回调

通过Block回调的方式获取回调内容，然后处理回调中的结果。

#### 注意事项

初始化SDK之后才能调用查询投影码

## 示例代码

```
// 获取EUA地址
NSString * castCode = self.codeString;
NSString * address = self.ipaddressString;
int port = [self.portString intValue];
int result = [mShareServiceController discover:castCode :address :port :^(id discoverCallback) {
    self.discoverResult = (DiscoverResult*)discoverCallback;
    NSLog(@"UI address is %@",self.discoverResult.address_list);
}];
```

## 4.4.4 场景 4： TLS 验证

### 描述

证书验证

### 业务流程

#### 证书验证接口调用

1. 调用setTls接口
2. 通过setTls接口传递TLS信息给SDK层进行证书验证，确保所传路径有证书存在，并将验证模式设为SERVER，否则验证会失败，并且无法成功连接。

### 注意事项

不调用此接口，不影响连接和投屏等功能，仅作为安全验证使用。如果验证失败，会导致无法连接。

## 示例代码

```
//设置TLS
IdeaShareServiceTlsInfo *tlsInfo = [[IdeaShareServiceTlsInfo alloc] init];
tlsInfo.caCertpath = @"/Users/**/Desktop";
tlsInfo.clientCertpath = @"/Users/**/Desktop";
tlsInfo.verifyMode = 1;
int result = [mShareServiceController setTls: tlsInfo :^(id tlsCallback) {
}];
```

## 4.4.5 场景 5： 连接/断开连接

### 描述

连接设备（ Ideahub/Ideaboard或Board ）

### 业务流程

#### 步骤1 连接接口调用

----结束

1. 调用connect接口
2. 通过Block回调的方式获取回调内容，然后处理连接回调结果。

#### 步骤2 断开调用

1. 调用Disconnect接口
2. 通过Block回调的方式获取回调内容，然后处理连接回调结果。

## 注意事项

必须在调用初始化和查询投影码后才能调用此接口。

## 示例代码

```
//连接
NSMutableArray * addr_list = self.discoverResult.address_list;
NSString * password = self.discoverResult.password;
NSLog(@"%@", password);
int port = 1444;
int connectResult = [mShareServiceController connect : addr_list : port : password :^id connectCallback) {
    self.deviceInfo = (DeviceInfo*)connectCallback;
};

// 断开连接
int result = [mShareServiceController disconnect:^id disconnectCallback) {
    reason = (int)disconnectCallback;
};
```

## 4.4.6 场景 6： 共享/停止共享

### 描述

连接设备后，调用共享接口发起共享或调用停止共享接口停止共享。

### 业务流程

#### 步骤1 接口调用

----结束

1. 调用startSharing 或 stopSharing 启动或停止共享。
2. 通过Block回调的方式获取回调内容，然后处理开始/停止回调结果。

## 注意事项

连接设备之后才能调用这两个接口。

## 示例代码

```
// 发起共享
int result = [mShareServiceController startSharing:^id startSharingCallback) {
};

// 停止共享
int result = [mShareServiceController stopSharing:^id stopSharingCallback) {
    reason = (int)stopSharingCallback;
};
```

## 4.4.7 场景 7： 投放音频

### 描述

共享前，投放音频选择投放或不投放音频

## 业务流程

### 1. 接口调用

调用setPolicy 设置投放音频。

## 示例代码

```
// 设置是否投放音频 YES 投放 NO 不投放
self.sharepolicy.isShareAudio = NO;
[mShareServiceController setPolicy:self.sharepolicy];

self.sharepolicy.isShareAudio = YES;
[mShareServiceController setPolicy:self.sharepolicy];
```

## 注意事项

连接设备之后投屏之前调用接口

## 4.4.8 场景 8： 遥控器

### 描述

在连接设备后，客户端可以查询设备状态，包括：麦克风状态、扬声器状态、摄像头状态、音量大小，是否在会议中，并且操作这些设备状态，如调节音量大小、离开会议（前提是已经入会）

## 业务流程

### 步骤1 接口调用

----结束

1. 调用getRemoteServiceStatus查询连接设备状态
2. 使用Block回调处理回调结果
3. 调用遥控器接口如（setRemoteMicMute 或 setXXXXX）请求遥控远端设备
4. 使用Block回调处理遥控器接口的回调事件
5. 通过notifyDelegate收到服务端主动推送过来的通知事件onxxxxNotify 事件，更新客户端遥控器状态
6. 在会议中时，调用leaveConference 或finishConference 离开会议，使用Block回调处理回调事件结果

## 注意事项

连接设备后遥控器功能才可用。

## 示例代码

```
// 查询远端设备状态，使用Block处理回调结果
if (mShareServiceController != nil) {
    int result = [mShareServiceController getRemoteServiceStatus:^(id status) {
        self.remoteserviceStatus = (RemoteServiceStatus *)status;
        if (self.remoteserviceStatus.isMute != 0) {
            dispatch_async(dispatch_get_main_queue(), ^{
                self.openMicrophoneButton.enabled = YES;
                self.closeMicrophoneButton.enabled = NO;
            });
        }
    }];
}
```

```
        });
    }];
}

设备控制，此处以麦克风静音和闭音为例，扬声器、摄像头、音量控制流程类似不再赘述
int result = [mShareServiceController setRemoteMicMute:false:^(id micOpenCallback) {
}];
if (result != 0)
{
    NSLog(@"open Microphone failed");
}
else{
    NSLog(@"open Microphone success");
    self.closeMicrophoneButton.enabled = YES;
    self.openMicrophoneButton.enabled = NO;
}
// 远端设备主动推送OnUpdateRemoteMicMuteNotify事件通知UI进行更新
- (void) OnRemoteMicStatusNotify : (int) isMute
{
    NSLog(@"OnRemoteMicStatusNotify: isMic = [%d]",isMute);
    if (isMute != 0) {
        dispatch_async(dispatch_get_main_queue(), ^{
            self.openMicrophoneButton.enabled = YES;
            self.closeMicrophoneButton.enabled = NO;
        });
    } else {
        dispatch_async(dispatch_get_main_queue(), ^{
            self.closeMicrophoneButton.enabled = YES;
            self.openMicrophoneButton.enabled = NO;
        });
    }
}
// 离开会议
int result = [mShareServiceController leaveConference:^(id leaveConferencecallback) {
}];

// 结束会议
int result = [mShareServiceController finishConference:^(id finishConferencecallback) {
}];
```

## 4.4.9 场景 9：设备推送

### 描述

远端设备定时向客户端推送 投屏状态、辅流状态、会议状态、如果在会议中申请主席，还会推送主席状态

### 业务流程

1. 处理投屏状态、辅流状态、会议状态、以及主席状态

### 示例代码

```
// 投屏状态推送IDEA_SHARE_APP_SERVER_STATUS、IDEA_SHARE_ERROR_ID枚举值参见接口参考
- (void) onShareStatusChangeNotify : (IDEA_SHARE_APP_SERVER_STATUS) status :
(IDEA_SHARE_ERROR_ID) reason
{
    NSLog(@"Delegate Status is %ld, reason is %ld", (long)status, (long)reason);
    NSLog(@"Get Status Number is %d",status);
    if (status == SERVER_STATUS_DISCONNECT)
    {
        NSLog(@"After Share status equals SERVER_STATUS_DISCONNECT ");
        dispatch_async(dispatch_get_main_queue(), ^{
            self.sharepolicy.isShareAudio = NO;
            [self->mShareServiceController setPolicy:self.sharepolicy];
        });
    }
}
```

```
        }];
    } else if (status == SERVER_STATUS_STOP_SHARE) {
        NSLog(@"After Share status equals SERVER_STATUS_STOP_SHARE ");
        dispatch_async(dispatch_get_main_queue(), ^{
            self.stopCaptureButton.enabled = NO;
            self.startCaptureButton.enabled = YES;
        });
    }
}
// 辅流和会议状态推送， 会议状态
- (void)onDevConfStatusChangeNotify : (IDEA_SHARE_APP_CONF_STATUS) confStatus :
(IDEA_SHARE_APP_AUX_STATUS) auxStatus
{
    NSLog(@"Delegate Conference Status is %ld, Aux Status is %ld", (long)confStatus, (long)auxStatus);
    if (confStatus == 1) {
        dispatch_async(dispatch_get_main_queue(), ^{
            self.leaveConferenceButton.enabled = YES;
        });
    } else {
        dispatch_async(dispatch_get_main_queue(), ^{
            self.leaveConferenceButton.enabled = NO;
            self.finishConferenceButton.enabled = NO;
        });
    }
}
// 枚举值IdeaShareDefault.h中定义
typedef NS_ENUM(NSUInteger, IDEA_SHARE_APP_CONF_STATUS) {
IDEA_SHARE_CONF_IDLE = 0, // 空闲
IDEA_SHARE_CONF_BUSY // 忙
};
typedef NS_ENUM(NSUInteger, IDEA_SHARE_APP_AUX_STATUS) {
IDEA_SHARE_AUX_IDLE = 0, // 空闲
IDEA_SHARE_AUX_BUSY // 忙
};
// 设备在会议中申请主席后，设备将推送主席状态
- (void)onRemoteServiceStatusChangedNotify : (RemoteServiceStatus *)remoteServiceStatus :
(int)changeCount
{
Switch(changeCount) {
case 1:
{
    NSLog(@"OnRemoteChairManStatusNotify: isChairMan = [%d]", remoteServiceStatus.isChairman);
}
}
}
```

## 4.4.10 场景 10：错误上报

### 描述

在使用过程中，出现异常时，SDK上报onErrorNotify事件

### 业务流程

处理onErrorNotify 事件回调

### 示例代码

```
// 错误信息上报 IDEA_SHARE_APP_ERR 类型参照接口参考
- (void)OnErrorHandler : (IDEA_SHARE_APP_ERR) errCode
{
    switch (errCode) {
        case IDEA_SHARE_ERROR_SUCCESS:
        {
            NSLog(@"Error Code is 0, Success");
        }
    }
}
```

```
        break;  
    }  
}  
}
```

## 4.4.11 场景 11：去初始化

### 描述

退出客户端APP时，调用去初始化SDK接口

### 业务流程

调用去初始化SDK接口

### 示例代码

```
// 去初始化SDK  
[mShareServiceController destroy];
```

## 4.4.12 场景 12：鼠标样式更改

### 描述

投屏成功后，调用更改鼠标样式接口

### 业务流程

调用更改鼠标样式接口

### 示例代码

```
// 去初始化SDK  
[mShareServiceController setMouseShape:0];
```

## 4.5 常见问题

### 4.5.1 获得日志

1. 日志地址为/Users/Huawei/Library/Caches/Logs，可以通过Finder打开，也可以通过屏幕左上方的“前往”按钮直接进入Library：
  - 通过Finder先进入Macintosh HD，然后选择用户，进入自己用户名目录，然后选择资源库，进入Caches文件夹，打开Logs文件夹

名称	修改日期	大小	种类
hme_v_log_all.txt	昨天下午8:15	3.2 MB	纯文本
hme_v_log_api.txt	昨天下午8:15	248 KB	纯文本
hme_v_log_error.txt	昨天下午8:15	1.3 MB	纯文本
hme_v_log_io.txt	昨天下午8:15	173 KB	纯文本
hme_v_log_trace.txt	昨天下午8:15	218 KB	纯文本
hw.packTEst 2021-0...--08-20-15-698.log	昨天下午8:15	58 KB	日志文件
projection_client.log	昨天下午8:15	971 KB	日志文件
tup_audio.log	昨天下午4:21	5 KB	日志文件
tup_mediaservice.log	昨天下午8:15	508 KB	日志文件
tup_video.log	昨天下午8:15	243 KB	日志文件
tup_videotrace.log	昨天下午8:13	8 KB	日志文件

- 选中Finder，然后点击左上方的“前往”按钮，选择“资源库” / “Library”，（如果未找到“资源库”或者“Library”，按住Option）。进入资源库/Library后，打开Caches—Logs，获取日志



## 4.5.2 声卡驱动安装

前言：如果想将MAC的音频进行投放，则需要进行音频资源的采集和重定向，此时需要使用声卡驱动；BlackHole是为MAC开发出的专门用于音频采集和重定向输出的第三

方开源驱动。由于苹果官方的限制，所有上app store的app必须开启沙盒运行，此时无法在安装应用程序的时候将BlackHole安装到系统的指定目录下，因此需要手动安装，这里给出手动安装BlackHole的方法。

1. 将附件中提供的BlackHole拷贝至MAC电脑并解压，将解压后的BlackHole驱动拷贝到系统路径下的/Library/Audio/Plug-Ins/HAL下，如果在/Library/Audio/Plug-Ins路径下没有HAL文件夹，则需要手动创建一个后再进行拷贝；

2. 打开MAC的终端控制台，执行以下指令：

```
sudo launchctl kickstart -kp system/com.apple.audio.coreaudiod
```

3. 如果拷贝或者执行指令的时候提示没有权限或者不允许安装相应的软件，则需要在终端控制台中执行以下指令：

```
sudo spctl --master-disable
```

然后打开设置->安全性与隐私->通用，勾选上允许从以下位置下载的应用中的“任何来源”选项，接着重新执行步骤1、2即可

## 4.6 Mac 接口参考

### 4.6.1 概述

本文主要介绍投屏SDK Objective C接口参考。主要包含接口函数定义、参数定义说明、附录，事件上报通知说明以及错误码等信息。

### 4.6.2 变更记录

表 4-3 变更记录

日期	版本	变更内容
2021-02-23	1.0.0	初稿完成。
2021-04-26	2.0.0	接口修订，SDK由dylib更改为framework

### 4.6.3 接口参考

#### 4.6.3.1 启动 APP

##### 4.6.3.1.1 IdeaShareServiceController : 投屏控制类，主要是提供业务接口

获取IdeaShareServiceController 单例

```
mShareServiceController = [IdeaShareServiceController shareInstance];
```

**接口描述**

获取IdeaShareServiceController 单例

**注意事项**

应用程序调用接口都依赖此单例，建议在应用程序初始化的时候就获取到此单例，并一直持有

#### 接口定义

```
+ (instancetype)shareInstance;
```

### 4.6.3.1.2 初始化日志

#### APP 调用 IdeaShareServiceController 实例对象的方法初始化 SDK 日志

```
int logRet = [mShareServiceController setLog:logInfo];
```

#### 接口描述

初始化日志前获取IdeaShareServiceController单例，然后调用 IdeaShareServiceController的setLog

#### 注意事项

请传入正确的IdeaShareServiceLogInfo对象，初始化接口在初始化SDK之前定义

#### 接口定义

```
- (int)setLog:(IdeaShareServiceLogInfo *)logInfo;
```

#### 参数描述

表 4-4 参数说明

参数	是否必须	类型	描述
logInfo	是	IdeaShareServiceLogInfo	初始化日志的必要参数，OC 类

### 4.6.3.1.3 初始化 SDK

#### APP调用IdeaShareServiceController实例对象的create方法

```
const IdeaShareServiceAppInfo *applInfo = [[IdeaShareServiceAppInfo alloc] init];
applInfo.applInfo = @"aa";
applInfo.exePath = @"bb";
int ret = [mShareServiceController create:applInfo];
```

#### 接口描述

初始化SDK

#### 注意事项

调用此接口前，要先给IdeaShareServiceController单例的notifyDelegate代理属性赋值为(id)self，否则初始化会失败。

#### 接口定义

```
- (int)create:(const IdeaShareServiceAppInfo *)applInfo;
```

#### 参数描述

**表 4-5 参数说明**

参数	是否必须	类型	描述
appInfo	是	IdeaShareServiceAppInfo	目前，此参数可任意赋值，无实际意义，但不能为nil

#### 4.6.3.1.4 去初始化 SDK

应用程序退出时应去初始化SDK

```
[mShareServiceController destroy];
```

##### 接口描述

去初始化SDK

##### 注意事项

mShareServiceController为IdeaShareServiceController的单例

##### 接口定义

```
- (int)destroy;
```

#### 4.6.3.1.5 投屏策略配置

APP调用IdeaShareServiceController实例对象的setPolicy方法

```
self.sharepolicy.isShareAudio = YES;
self.sharepolicy.shareQuality = 1;
self.sharepolicy.isCompatibility = YES;
int result = [mShareServiceController setPolicy:self.sharepolicy];
```

##### 接口描述

该接口会设置是否开启兼容性，用于兼容现网CBC加密模式。注意：当前现网均是CBC加密模式，此接口必须设置成开启兼容性。

该接口用于在连接设备后，设置是否需要投放音频到设备。

##### 注意事项

注意兼容性必须设置成开启(true)。

##### 接口定义

```
- (int)setPolicy:(SharePolicy *)sharePolicy;
```

**表 4-6 参数说明**

参数	是否必须	类型	描述
sharePolicy	是	SharePolicy	OC类，用于设置是否投放音频以及投屏策略

### 4.6.3.1.6 投屏码解析

```
NSString * castCode = self.codeString;
NSString * address = self.ipaddressString;
int port = [self.portString intValue];
int result = [mShareServiceController discover:castCode :address :port :^(id discoverCallback) {
}];
```

#### 接口描述

APP调用IdeaShareServiceController实例对象的discover方法开始解析，SDK解析投影码成功后会通过Block回调将IP地址集和连接码传回UI层

#### 注意事项

离线模式下，仅需输入投屏码castCode；address和port适用于在线模式投屏

#### 接口定义

```
- (int)discover:(NSString *)castCode:(NSString *)address: (int)port: (callback)discovercallback;
```

表 4-7 参数说明

参数	是否必须	类型	描述
castCode	是	NSString	投屏码
address	否	NSString	在线投屏时使用，为SMC(EUA)的IP地址
port	否	int	在线投屏时使用，为SMC(EUA)的端口号
discovercallback	是	Discover Result	typedef void (^callback)(id t), 包含回调内容 – 地址集，连接码以及解析结果

### 4.6.3.2 连接设备

#### 4.6.3.2.1 证书验证

```
IdeaShareServiceTlsInfo *tlsInfo = [[IdeaShareServiceTlsInfo alloc] init];
tlsInfo.caCertpath = @"/Users/**/Desktop";
tlsInfo.clientCertpath = @"/Users/**/Desktop";
tlsInfo.verifyMode = 1;
int result = [mShareServiceController setTls: tlsInfo :^(id tlsCallback) {
}];
```

#### 接口描述

APP需要调用IdeaShareServiceController实例对象的setTls方法，带上参数本地ca证书的路径和客户端校验的路径以及验证模式，证书验证成功后，便可进行连接操作；否则，连接不会成功。如果不调用此接口，则不需要本地证书，也可以正常连接成功。

#### 接口定义

```
- (int)setTls:(IdeaShareServiceTlsInfo *)tlsInfo:(callback)tlsCallback;
```

#### 参数描述

**表 4-8 connect 参数说明**

参数	是否必须	类型	描述
tlsInfo	是	IdeaShareService TlsInfo	TLS信息 OC类
tlsCallback	是	int	TLS校验失败会返回一个int值，否则不会传递任何值

#### 4.6.3.2.2 设备连接

```
NSMutableArray * addr_list = self.discoverResult.address_list;  
NSString * password = self.discoverResult.password;  
int port = 1444;  
int connectResult = [mShareServiceController connect: addr_list :port :password :^(id connectCallback) {  
}];
```

##### 接口描述

APP需要调用IdeaShareServiceController实例对象的connect方法，带上参数IP地址集、连接码（投屏码解析接口的回调结果）和端口开始连接（离线投屏端口暂定为1444），SDK连接成功后会通过Block回调将DeviceInfo信息传回UI层，APP需要进行相应处理

##### 接口定义

```
- (int)connect:(NSMutableArray *)address_list:(int)port:(NSString *)password:(callback)connectcallback;
```

##### 参数描述

**表 4-9 connect 参数说明**

参数	是否必须	类型	描述
address_list	是	NSMutableArray	投屏码解析出来的地址集
port	是	int	端口号；离线投屏端口号固定为1444
password	是	NSString	投屏码解析出来的连接码
connectcallback	是	DeviceInfo	typedef void (^callback)(id t), 包含回调内容 - 设备地址，设备名称和是否支持远程控制

#### 4.6.3.2.3 断开连接

```
int result = [mShareServiceController disconnect:^^(id disconnectCallback) {  
}];
```

##### 接口描述

APP调用IdeaShareServiceController实例对象的disconnect方法。SDK断开连接成功后会通过Block回调通知UI层，APP需要进行相应处理

##### 方法定义

```
- (int)disconnect:(callback)disconnectcallback;
```

#### 参数描述

表 4-10 disConnect 参数说明

参数	是否必须	类型	描述
disconnectcallback	是	int	typedef void (^callback)(id t), 包含回调内容 - 断连的原因

### 4.6.3.2.4 开始投屏

```
int result = [mShareServiceController startSharing:^(id startSharingCallback) {  
}];
```

#### 接口描述

APP调用IdeaShareServiceController实例对象的startSharing方法，SDK投屏成功之后会通过Block回调通知UI层，APP需要进行相应处理。

需要注意先调用[4.6.3.1.5 投屏策略配置](#)将兼容性设置为true。

#### 方法定义

```
- (int)startSharing:(callback)startSharingcallback;
```

#### 参数描述

表 4-11 startSharing 参数说明

参数	是否必须	类型	描述
StartSharingcallback	是	int	typedef void (^callback)(id t), 包含回调内容 - 开始投屏结果

### 4.6.3.2.5 结束投屏

```
int result = [mShareServiceController stopSharing:^(id stopSharingCallback) {  
}];
```

#### 接口描述

APP调用IdeaShareServiceController实例对象的stopSharing方法。SDK结束投屏之后通过Block回调通知UI层，APP需要进行相应处理

#### 方法定义

```
- (int)stopSharing:(callback)stopSharingcallback;
```

#### 参数描述

**表 4-12 Connect 参数说明**

参数	是否必须	类型	描述
stopSharingcallback	是	int	typedef void (^callback)(id t), 包含回调内容 - 结束投屏的原因

#### 4.6.3.2.6 远程遥控功能

```
int result = [mShareServiceController getRemoteServiceStatus:^(id status) {  
}];
```

##### 接口描述

APP打开遥控器界面的时候，需要调用IdeaShareServiceController实例对象的getRemoteServiceStatus方法来获取最新的Ideahub/board状态。SDK获取到最新的状态之后，会主动通知UI层，并且通过Block回调将最新状态传送回UI层，APP需要进行相应处理

##### 注意事项

此接口回调参数是一个OC类 – RemoteServiceStatus，其包含六个属性，分别对应远程操控的六个状态参数。

##### 接口定义

```
- (int)getRemoteServiceStatus:(callback)remoteServiceStatuscallback;
```

##### 参数描述

**表 4-13 参数说明**

参数	是否必须	类型	描述
remoteServiceStatuscallback	是	RemoteServiceStatus	typedef void (^callback)(id t), 包含回调内容: 1. bool isMute 2. bool isCameraMute 3. int volume 4. bool isSpeakerMute 5. bool isChairman 6. bool isBase

#### 4.6.3.2.7 打开/关闭麦克风

```
int result = [mShareServiceController setRemoteMicMute:false:^(id micOpenCallback) {}];
```

##### 接口描述

APP界面执行打开/关闭麦克风后，需要调用IdeaShareServiceController实例对象的setRemoteMicMute方法，设置Ideahub/board的麦克风。SDK在获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理

**注意事项****接口定义**

```
- (int)setRemoteMicMute:(bool)isMute:(callback)micMutecallback;
```

**参数描述****表 4-14 参数说明**

参数	是否必须	类型	描述
isMute	是	bool	是否设置Ideahub/board麦克风静音
micMutecallback	是	int	typedef void (^callback)(id t), 包含回调内容 – Ideahub/board静音结果

**4.6.3.2.8 调节扬声器音量**

```
[mShareServiceController setSpeakerVolume:self.volumeNumber:^(id volumecallback) {}];
```

**接口描述**

APP界面执行扬声器音量调节后，需要调用IdeaShareServiceController实例对象的setRemoteSpeakerVolume方法，设置Ideahub/board的扬声器音量。SDK获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理

**注意事项**

Block回调返回的int值是接口调用后大屏的音量。

**接口定义**

```
- (int)setSpeakerVolume:(int)volume:(callback)volumecallback;
```

**参数描述****表 4-15 参数说明**

参数	是否必须	类型	描述
volume	是	int	想要设置的大屏音量(0-21)
volumecallback	是	int	设置后，大屏目前的音量

**4.6.3.2.9 打开/关闭扬声器**

```
int result = [mShareServiceController setRemoteSpeakerMute:true:^(id speakerCloseCallback) {}];
```

**接口描述**

APP界面执行打开/关闭扬声器后，需要调用IdeaShareServiceController实例对象的setRemoteSpeakerMute方法，设置Ideahub/board的扬声器。SDK在获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理。

### 注意事项

当大屏音量为0时，屏默认为静音状态，无法在保证音量为0的状态下打开Speaker。建议，当大屏音量为0时，对此接口的使用加以限制。

### 接口定义

```
- (int)setRemoteSpeakerMute:(bool)isSpeakerMute:(callback)speakerMutecallback;
```

### 参数描述

表 4-16 参数说明

参数	是否必须	类型	描述
isSpeakerMute	是	bool	设置Ideahub/board音响是否静音
speakerMutecallback	是	int	设置Ideahub/board音响静音的结果

## 4.6.3.2.10 打开/关闭摄像头

```
int result = [mShareServiceController setRemoteCameraMute:true:^(id cameraCloseCallback) {}];
```

### 接口描述

APP界面执行打开/关闭摄像机后，需要调用IdeaShareServiceController实例对象的setRemoteCameraMute方法，设置Ideahub/board的摄像机。SDK在获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理

### 注意事项

### 接口定义

```
- (int)setRemoteCameraMute:(bool)isCameraMute:(callback)cameraMutecallback;
```

### 参数描述

表 4-17 参数说明

参数	是否必须	类型	描述
isCameraMute	是	bool	设置Ideahub/board摄像机是否开启
cameraMutecallback	是	int	设置Ideahub/board摄像机开关的结果

### 4.6.3.2.11 离开会议

```
int result = [mShareServiceController leaveConference:^(id leaveConferencecallback) {}];
```

#### 接口描述

APP界面执行离开会议后，需要调用IdeaShareServiceController实例对象的leaveConference方法，离开会议。SDK在获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理。

#### 注意事项

#### 接口定义

```
- (int)leaveConference:(callback)leaveConferencecallback;
```

#### 参数描述

表 4-18 参数说明

参数	是否必须	类型	描述
leaveConferencecallback	是	int	离开会议的结果

### 4.6.3.2.12 结束会议

```
int result = [mShareServiceController finishConference:^(id t) {}];
```

#### 接口描述

APP界面执行结束会议后，需要调用IdeaShareServiceController实例对象的finishConference方法，结束会议。SDK在获取到最新状态之后，通过Block回调主动通知UI层设置结果，APP需要进行相应处理。

#### 注意事项

#### 接口定义

```
- (int)finishConference:(callback)finishConferencecallback;
```

#### 参数描述

表 4-19 参数说明

参数	是否必须	类型	描述
finishConferencecallback	是	int	结束会议的结果

### 4.6.3.2.13 版本查询

```
self.versionLabel.stringValue = [self->mShareServiceController getSDKVersion];
```

#### 接口描述

APP调用IdeaShareServicecontroller实例对象的getSDKVersion方法。返回值为版本信息

#### 注意事项

#### 接口定义

```
- (NSString *)getSDKVersion;
```

### 4.6.3.2.14 打开权限窗口

```
[mShareServiceController openAuthorizationSetting:IDEA_SHARE_MIC_AUTH];
```

#### 接口描述

APP调用IdeaShareServicecontroller实例对象的openAuthorizationSetting方法。

#### 注意事项

调用该方法，会打开MacOS系统对应的权限窗口。

注：macOS系统下，Framework格式的SDK包里包含有苹果开发自带的属性列表文件Info.plist，此文件为一种XML格式，内含<!DOCTYPE>标签以及标签中的DTD网址<http://www.apple.com/DTDs/PropertyList-1.0.dtd>。

#### 接口定义

```
- (void) openAuthorizationSetting : (IDEA_SHARE_APP_AUTH) authorization;
```

### 4.6.3.2.15 服务端版本查询

```
int result = [mShareServiceController getSDKLatestVersion:^(id sdkLatestVersioncallback ){}];
```

#### 接口描述

APP调用IdeaShareServicecontroller实例对象的getSDKLatestVersion方法。回调值为服务端版本信息。

#### 注意事项

此接口获取的是远端服务端的版本号，与getSDKVersion有区别

#### 接口定义

```
- (int) getSDKLatestVersion : (callback) sdkLatestVersioncallback;
```

### 4.6.3.2.16 服务端鼠标样式设置

```
int result = [mShareServiceController setMouseShape:0];
```

#### 接口描述

APP调用IdeaShareServicecontroller实例对象的setMouseShape, 无block回调。

#### 注意事项

此接口能设置大屏侧（投屏后）鼠标的样式，可选择激光笔样式，箭头样式，和不显示。

#### 接口定义

```
- (int) setMouseShape:(IDEA_SHARE_MOUSE_SHAPE)mouseShape;
```

### 4.6.3.3 接口调用结果通知

#### 4.6.3.3.1 tlsCallback

##### 回调方法描述

TLS设置接口Block回调

##### 回调方法定义

```
typedef void (^callback)(id t);
@property(nonatomic, copy) callback tlsCallback;
```

##### 参数描述

表 4-20 参数说明

参数	是否必须	类型	描述
tlsCallback	是	int	OC类, TLS验证失败通知

##### 示例代码

```
int result = [mShareServiceController setTls: tlsInfo:^(id tlsCallback) {
    int res = (int)tlsCallback;
}];
```

#### 4.6.3.3.2 discoverCallback

##### 回调方法描述

设备发现接口Block回调

##### 回调方法定义

```
typedef void (^callback)(id t);
@property(nonatomic, copy) callback discoverCallback;
```

##### 参数描述

表 4-21 参数说明

参数	是否必须	类型	描述
discoverCallback	是	DiscoverResult	OC类, 设备发现结果

##### 示例代码

```
int result = [mShareServiceController discover:castCode :address :port :^(id discoverCallback) {
    self.discoverResult = (DiscoverResult*)discoverCallback;
}];
```

#### 4.6.3.3.3 connectCallback

##### 回调方法描述

连接接口Block回调

#### 回调方法定义

```
typedef void (^callback)(id t);
@property(nonatomic, copy) callback connectCallback;
```

#### 参数描述

表 4-22 参数说明

参数	是否必须	类型	描述
ConnectCallback	是	DeviceInfo	OC类，设备信息

#### 示例代码

```
int connectResult = [mShareServiceController connect:addr_list :port :password :^(id connectCallback) {
    self.deviceInfo = (DeviceInfo*)connectCallback;
}];
```

### 4.6.3.3.4 disconnectCallback

#### 回调方法描述

断开连接接口Block回调

#### 回调方法定义

```
typedef void (^callback)(id t);
@property(nonatomic, copy) callback disconnectCallback;
```

#### 参数描述

表 4-23 参数说明

参数	是否必须	类型	描述
disconnectCallback	是	int	Reason-暂无实际作用

#### 示例代码

```
int result = [mShareServiceController disconnect:^(id disconnectCallback) {
    reason = (int)disconnectCallback;
}];
```

### 4.6.3.3.5 startSharingCallback

#### 回调方法描述

开始投屏接口Block回调

#### 回调方法定义

```
@property(nonatomic, copy) callback startSharingCallback;
```

### 参数描述

参数	是否必须	类型	描述
startSharingCallback	是	int	Result - 暂无实际作用

### 示例代码

```
int result = [mShareServiceController startSharing:^(id startSharingCallback) {  
}];
```

## 4.6.3.3.6 stopSharingCallback

### 回调方法描述

结束投屏接口Block回调

### 回调方法定义

```
@property(nonatomic, copy) callback stopSharingCallback;
```

### 参数描述

表 4-24 参数说明

参数	是否必须	类型	描述
stopSharingCallback	是	int	Reason -暂无实际作用

### 示例代码

```
int result = [mShareServiceController stopSharing:^(id stopSharingCallback) {  
    reason = (int)stopSharingCallback;  
}];
```

## 4.6.3.3.7 remoteServiceStatusCallback

### 回调方法描述

远程控制服务状态接口Block回调

### 回调方法定义

```
@property(nonatomic, copy) callback remoteServiceStatusCallback;
```

### 参数描述

表 4-25 参数说明

参数	是否必须	类型	描述
RemoteServiceStatusCallback	是	RemoteServiceStatus	OC类，包含大屏侧的六个遥控状态

### 示例代码

```
int result = [mShareServiceController getRemoteServiceStatus:^(id status) {  
    self.remoteserviceStatus = (RemoteServiceStatus *)status;  
}];
```

## 4.6.3.3.8 micMuteCallback

### 回调方法描述

设置麦克风静音接口Block回调

### 回调方法定义

```
@property(nonatomic, copy) callback micMuteCallback;
```

### 参数描述

表 4-26 参数说明

参数	是否必须	类型	描述
micMuteCallback	是	int	设置麦克风静音的结果

### 示例代码

```
int result = [mShareServiceController setRemoteMicMute:false:^(id micMuteCallback) {  
}];
```

## 4.6.3.3.9 speakerVolumeCallback

### 回调方法描述

大屏扬声器音量接口Block回调

### 回调方法定义

```
@property(nonatomic, copy) callback speakerVolumeCallback;
```

### 参数描述

表 4-27 参数说明

参数	是否必须	类型	描述
speakerVolumeCallback	是	int	设置大屏扬声器音量结果

### 示例代码

```
[mShareServiceController setSpeakerVolume:self.volumeNumber:^(id speakerVolumecallback) {  
}];
```

#### 4.6.3.3.10 speakerMuteCallback

##### 回调方法描述

大屏扬声器静音接口Block回调

##### 回调方法定义

```
@property(nonatomic, copy) callback speakerMuteCallback;
```

##### 参数描述

表 4-28 参数说明

参数	是否必须	类型	描述
speakerMuteCallba ck	是	int	Result -暂无实际作用

##### 示例代码

```
int result = [mShareServiceController setRemoteSpeakerMute:true:^(id speakerMuteCallback) {  
}];
```

#### 4.6.3.3.11 cameraMuteCallback

##### 回调方法描述

大屏摄像头开关接口Block回调

##### 回调方法定义

```
@property(nonatomic, copy) callback speakerMuteCallback;
```

##### 参数描述

表 4-29 参数说明

参数	是否必须	类型	描述
speakerMute Callback	是	int	Result -暂无实际作用

##### 示例代码

```
int result = [mShareServiceController setRemoteCameraMute:true:^(id cameraMuteCallback) {  
}];
```

#### 4.6.3.3.12 leaveConfCallback

##### 回调方法描述

离开会议接口Block回调

### 回调方法定义

```
@property(nonatomic, copy) callback leaveConfCallback;
```

### 参数描述

表 4-30 leaveConfCallback 参数说明

参数	是否必须	类型	描述
leaveConfCal lback	是	int	离开会议的结果

### 示例代码

```
int result = [mShareServiceController leaveConference:^(id leaveConfCallback) {  
}];
```

## 4.6.3.3.13 finishConfCallback

### 回调方法描述

结束会议接口Block回调

### 回调方法定义

```
@property(nonatomic, copy) callback finishConfCallback;
```

### 参数描述

表 4-31 参数说明

参数	是否必须	类型	描述
finishConfCallback	是	int	结束会议的结果

### 示例代码

```
int result = [mShareServiceController finishConference:^(id t) {  
}];
```

## 4.6.3.3.14 sdkLatestVersionCallback

### 回调方法描述

服务端版本查询接口Block回调

### 回调方法定义

```
@property(nonatomic, copy) callback sdkLatestVersionCallback;
```

### 参数描述

表 4-32 参数说明

参数	是否必须	类型	描述
sdkLatestVersionCallback	是	NSString	服务端版本号

### 示例代码

```
int result = [mShareServiceController getSDKLatestVersion:^(id t) {  
}];
```

## 4.6.3.4 设备推送通知

### 4.6.3.4.1 onRemoteServiceStatusChangedNotify

#### 回调方法描述

设备推送服务端遥控状态通知

#### 回调方法定义

```
- (void)onRemoteServiceStatusChangedNotify:(RemoteServiceStatus *)remoteServiceStatus : (int)changeCount;
```

#### 参数描述

表 4-33 参数说明

参数	是否必须	类型	描述
remoteServiceStatus	是	RemoteServiceStatus	OC类，服务端遥控状态
changeCount	是	int	遥控状态变化指数，根据isMute, isCameraMute, volume, isSpeakerMute, isChairman的顺序从左到右，代表五位的二进制，全部默认为0，当有某个状态改变了，则设为1，代表改变的状态是哪一个。例如，参数为16的时候，二进制为10000，代表isMute发生了改变了，通过remoteServiceStatus参数去获取改变后的状态

### 示例代码

```
- (void) onRemoteServiceStatusChangedNotify : (RemoteServiceStatus *)remoteServiceStatus : (int)changeCount  
{  
    switch (changeCount) {
```

```
case 16:  
{  
    NSLog(@" onRemoteMicStatusNotify:isMic = [%d] ",remoteServiceStatus.isMute);  
    break;  
}  
}
```

#### 4.6.3.4.2 onDevConfStatusNotify

##### 回调方法描述

设备推送辅流和会议状态通知

##### 回调方法定义

```
- (void)onDevConfStatusChangeNotify:(IDEA_SHARE_APP_CONF_STATUS)confStatus:  
(IDEA_SHARE_APP_AUX_STATUS)auxStatus;
```

##### 参数描述

表 4-34 参数说明

参数	是否必须	类型	描述
confStatus	是	IDEA_SHARE_APP_CO NF_STATUS	IDEA_SHARE_CONF_IDLE 0 为空闲状态 IDEA_SHARE_CONF_BUSY 1 为会议状态
auxStatus	是	IDEA_SHARE_APP_AU X_STATUS	IDEA_SHARE_AUX_IDLE 0 为空闲状态 IDEA_SHARE_AUX_BUSY 1 为辅流发送状态

##### 示例代码

```
- (void) onDevConfStatusChangeNotify : (IDEA_SHARE_APP_CONF_STATUS) confStatus :  
(IDEA_SHARE_APP_AUX_STATUS) auxStatus  
{  
    if (confStatus == 1) {  
  
    } else {  
  
    }  
}
```

#### 4.6.3.4.3 onShareStatusChangeNotify

##### 回调方法描述

设备推送共享状态通知

##### 回调方法定义

```
- (void)onShareStatusChangeNotify:(IDEA_SHARE_APP_SERVER_STATUS)status:  
(IDEA_SHARE_ERROR_ID)reason;
```

##### 参数描述

表 4-35 参数说明

参数	是否必须	类型	描述
status	是	IDEA_SHARE_APP_STATUS	设备状态通知枚举值
reason	是	IDEA_SHARE_ERROR_ID	错误码

### 示例代码

```
- (void) onShareStatusChangeNotify : (IDEA_SHARE_APP_SERVER_STATUS) status :  
(IDEA_SHARE_ERROR_ID) reason  
{  
    if (status == 3)  
    {  
    } else if (status == 2) {  
    }  
}
```

#### 4.6.3.4.4 onServerMouseFeatureNotify

##### 回调方法描述

设备推送共享状态通知

##### 回调方法定义

```
- (void) onServerMouseFeatureNotify : (int) mouseResult;
```

##### 参数描述

表 4-36 参数说明

参数	是否必须	类型	描述
mouseResult	是	int	设备是否支持更改鼠标样式, 1为可支持, 非1为不支持

### 示例代码

```
- (void) onServerMouseFeatureNotify (int) mouseResult  
{  
if(mouseResult == 1) {  
    NSLog(@"The Server is available to set the style, [%d]", mouseResult);  
} else {  
    NSLog(@"The Server is not available to set the style, [%d]", mouseResult);  
}  
}
```

#### 4.6.3.5 错误通知

#### 4.6.3.5.1 onErrorNotify

##### 回调方法描述

SDK向上层推送错误通知

##### 回调方法定义

```
- (void)onErrorNotify:(IDEA_SHARE_APP_ERR)errCode;
```

##### 参数描述

表 4-37 onErrorNotify 参数说明

参数	是否必须	类型	描述
errCode	是	int	错误码

##### 示例代码

```
- (void) onErrorNotify : (IDEA_SHARE_APP_ERR) errCode
{
    switch (errCode) {
        case IDEA_SHARE_ERROR_SUCCESS:
        {
            NSLog(@"Error Code is 0, Success");
            break;
        }
        case IDEA_SHARE_ERROR_CONNECT_FAIL:
        {
            break;
        }
    }
}
```

### 4.6.4 附录

#### 4.6.4.1 枚举定义

##### 4.6.4.1.1 LOG\_LEVEL

枚举值	对应值	描述
IDEA_LEVEL_ERROR	0	错误级别
IDEA_LEVEL_WARNING	1	警告级别
IDEA_LEVEL_INFO	2	信息级别
IDEA_LEVEL_DEBUG	3	调试级别

#### 4.6.4.1.2 IDEA\_SHARE\_APP\_DEFINAION\_QUAITY

枚举值	对应值	描述
QUALITY_FLUENCY_FIRST	0	流畅优先, 默认值
QUAITY_CLEAR_FIRST	1	清晰优先
QUAITY_ADAPTIVE	2	自适应

#### 4.6.4.1.3 IDEA\_SHARE\_APP\_AUTH

枚举值	对应值	描述
IDEA_SHARE_MIC_AUTH	0	打开麦克风权限窗口
IDEA_SHARE_RECORD_SCREEN_AUTH	1	打开录屏权限窗口
IDEA_SHARE_ACCESSIBILITY_AUTH	2	打开辅助权限窗口

#### 4.6.4.1.4 IDEA\_SHARE\_APP\_CONF\_STATUS

枚举值	对应值	描述
IDEA_SHARE_CONF_IDLE	0	空闲
IDEA_SHARE_CONF_BUSY	1	忙

#### 4.6.4.1.5 IDEA\_SHARE\_APP\_AUX\_STATUS

枚举值	对应值	描述
IDEA_SHARE_AUX_IDLE	0	空闲
IDEA_SHARE_AUX_BUSY	1	忙

#### 4.6.4.1.6 IDEA\_SHARE\_APP\_SERVER\_STATUS

枚举值	对应值	描述
SERVER_STATUS_CONNECT	0	连接状态
SERVER_STATUS_SHARE	1	投屏共享状态
SERVER_STATUS_STOP_SHARE	2	停止投屏共享状态
SERVER_STATUS_DISCONNECT	3	断开连接状态

#### 4.6.4.1.7 IDEA\_SHARE\_APP\_ERR

枚举值	对应值	描述
DISCOVER_ERROR	0	设备发现错误
CONNECT_ERROR	1	连接错误
DISCONNECT_ERROR	2	断开连接错误
SHARE_PLAY_ERROR	3	启动投屏错误
SHARE_STOP_ERROR	4	停止投屏错误
GET_REMOTE_SERVICE_STATUS_ERROR	5	遥控器参数查询结果错误
SET_REMOTE_MIC_MUTE_ERROR	6	打开\关闭麦克风结果错误
SET_REMOTE_SPEAKER_VOLUME_ERROR	7	调节扬声器音量结果错误
SET_REMOTE_SPEAKER_MUTE_ERROR	8	打开\关闭扬声器结果错误
SET_REMOTE_CAMERA_MUTE_ERROR	9	打开\关闭摄像机结果错误
LEAVE_CONFERENCE_ERROR	10	离开会议结果错误
END_CONFERENCE_ERROR	11	结束会议结果错误
GET_SDK_LATEST_VERSION_ERROR	12	获取SDK最新版本错误

枚举值	对应值	描述
PROCESS_CONNECTING_ERROR	13	正在处理连接请求
IN_CONNECTION_ERROR	14	客户端已处于连接
PROJECTION_CODE_ERROR	15	投影码错误 数字+字母组合
AUTH_ERROR	16	鉴权失败
CLIENT_IS_LOCKED_ERROR	17	客户端被锁定
CONNECT_BUSY_ERROR	18	终端忙碌
CONNECT_UPDATING_ERROR	19	终端正在升级
WEB_DENY_ERROR	20	WEB被禁用
NO_COMMON_DATA_CAP_ERROR	21	和客户端没有辅流公共能力
GET_TOKEN_FAILD_ERROR	22	获取辅流令牌失败
LOW_BANDWIDTH_ERROR	23	带宽过低，无法发送辅流
CONF_NO_DATA_CAP_ERROR	24	会议无辅流能力
MULTICONNECT_BUSY_ERROR	25	终端正在处理辅流发送，暂且不接受辅流发送该请求
EUA_AUTH_ERROR	26	在线投屏鉴权失败
EUA_IS_LOCKED_ERROR	27	在线投屏已经锁定
MOUSE_SHAPE_ERROR	28	鼠标设置失败
ERROR_BUTT	29	预留错误码

#### 4.6.4.1.8 IDEA\_SHARE\_APP\_TLS\_VERIFY\_MODE

枚举值	对应值	描述
TLS_VERIFY_MODE_NONE	0	不验证
TLS_VERIFY_MODE_SERVER	1	服务端验证
TLS_VERIFY_MODE_CLIENT	2	客户端验证
TLS_VERIFY_MODE_BOTH	3	双向验证

枚举值	对应值	描述
TLS_VERIFY_MODE_BUTT	4	( 无意义 )

#### 4.6.4.1.9 IDEA\_SHARE\_MOUSE\_SHAPE

枚举值	对应值	描述
MOUSE_SHAPE_HIDE	0	鼠标隐藏
MOUSE_SHAPE_BLACK_ARROW	253	鼠标箭头样式
MOUSE_LASER_POINTER	255	鼠标激光样式

#### 4.6.4.2 OC 类定义

##### 4.6.4.2.1 IdeaShareServiceAppInfo

参数名	类型	描述
applInfo	NSString	App描述信息，长度限制为128
exePath	NSString	执行路径，长度限制为128

##### 4.6.4.2.2 IdeaShareServiceLogInfo

日志对象参数

参数名	类型	描述
path	NSString	日志路径，最大长度限制256
logLevel	枚举值 <a href="#">4.6.4.1.1 LOG_LEVEL</a>	日志级别：错误， 警告， 信息， 调试
enable	bool	是否开启日志

##### 4.6.4.2.3 SharePolicy

投放音频策略参数

参数名	类型	描述
shareQuality	4.6.4.1.2 IDEA_SHARE_APP_DEFINION_QUAITY	清晰度
isShareAudio	bool	NO表示不共享音频, YES表示共享音频
isCompatibility	bool	NO表示不兼容, YES表示兼容

#### 4.6.4.2.4 DiscoverResult

设备发现结果

参数名	类型	描述
result	int	设备发现结果 0 成功 非0 失败
password	NSString	设备连接码
addressList	NSMutableArray	设备IP地址列表

#### 4.6.4.2.5 DeviceInfo

远端设备信息

参数名	类型	描述
deviceAddress	NSString	终端地址 ( IP地址 or 域名 )
deviceName	NSString	终端名(会场名)
isSupportRemote	bool	终端侧是否支持遥控器 VOS_TRUE:支持,VOS_FALSE:不支持

#### 4.6.4.2.6 RemoteServiceStatus

远端设备状态

参数名	类型	描述
isMute	bool	麦克风是否关闭 YES为关闭, NO为打开
isCameraMute	bool	摄像头是否关闭 YES为关闭, NO为打开
volume	int	扬声器音量值 范围 0 ~21

参数名	类型	描述
isSpeakerMute	bool	扬声器是否关闭 YES为关闭，NO为打开
isChairman	bool	在会议中是否是主席 YES为主席，NO为非主席
isBase	bool	是否是标准版 YES为标准版本不带摄像头，NO为非标准版本带摄像头

#### 4.6.4.2.7 IdeaShareServiceTlsInfo

TLS信息		
参数名	类型	描述
caCertpath	NSString	ca证书地址
clientCertpath	NSString	客户端证书地址
verifyMode	IDEA_SHARE_APP_TLS_VERIFY_MODE	验证模式

#### 4.6.4.3 SDK 接口头文件定义

##### 4.6.4.3.1 IdeaShareEventHandler.h

协议形式的代理回调，IdeaShareServiceController的实例对象有一个notifyDelegate属性，应用程序需要为此属性赋值来注册回调，建议赋值为self

##### 4.6.4.3.2 IdeaShareServiceController.h

SDK API 和 Block回调定义文件

#### 4.6.4.4 公网地址说明

macOS系统中，Framework格式的SDK包在编译的过程中会自动产生含有苹果开发自带的属性列表文件Info.plist，此文件为一种XML格式，内含<!DOCTYPE>标签以及标签中的DTD网址 [1.5.4](#)。SDK本身不会访问该网址。

## 4.7 错误码参考

### 4.7.1 错误码

参照枚举值IDEA\_SHARE\_APP\_ERR错误码定义

# 5 Windows SDK

## 5.1 概述

投屏二次开发Windows SDK 提供了一套完整的接口集合，开发者可以通过调用 IdeaShare SDK ( 以下简称SDK ) 开放的API，快速集成投屏能力，包括启动投屏、遥控器、投放音频等。

## 5.2 变更记录

表1-1 变更记录

日期	版本	变更内容
2021-02-07	1.0.0	初稿完成

## 5.3 快速入门

本文面向有一定C++和Qt开发能力的开发者，以Demo编译流程为例，介绍如何使用SDK 进行二次开发。

### 5.3.1 预置条件

支持SMC 2.0 、SMC 3.0, 云会议组网。

### 5.3.2 开发环境准备

表1-2 环境要求

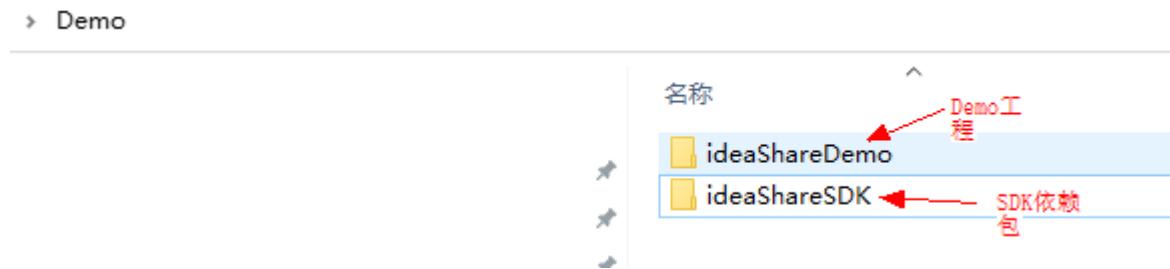
名称	要求
Qt版本	qt-opensource-windows-x86-5.14.2
Visual Studio	Visual Studio 2017专业版

名称	要求
CPU架构支持	x86-32/64位
Cmake	3.14.1 或以上

### 5.3.3 SDK 快速集成

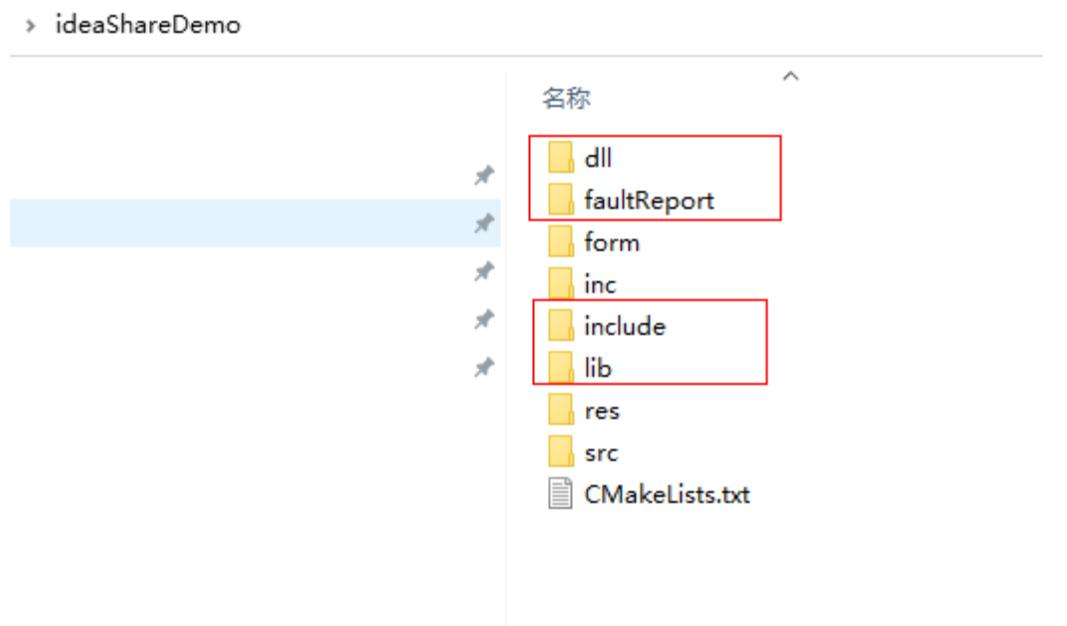
下载解压软件包，得到如下几个文件：

图1-1 文件目录



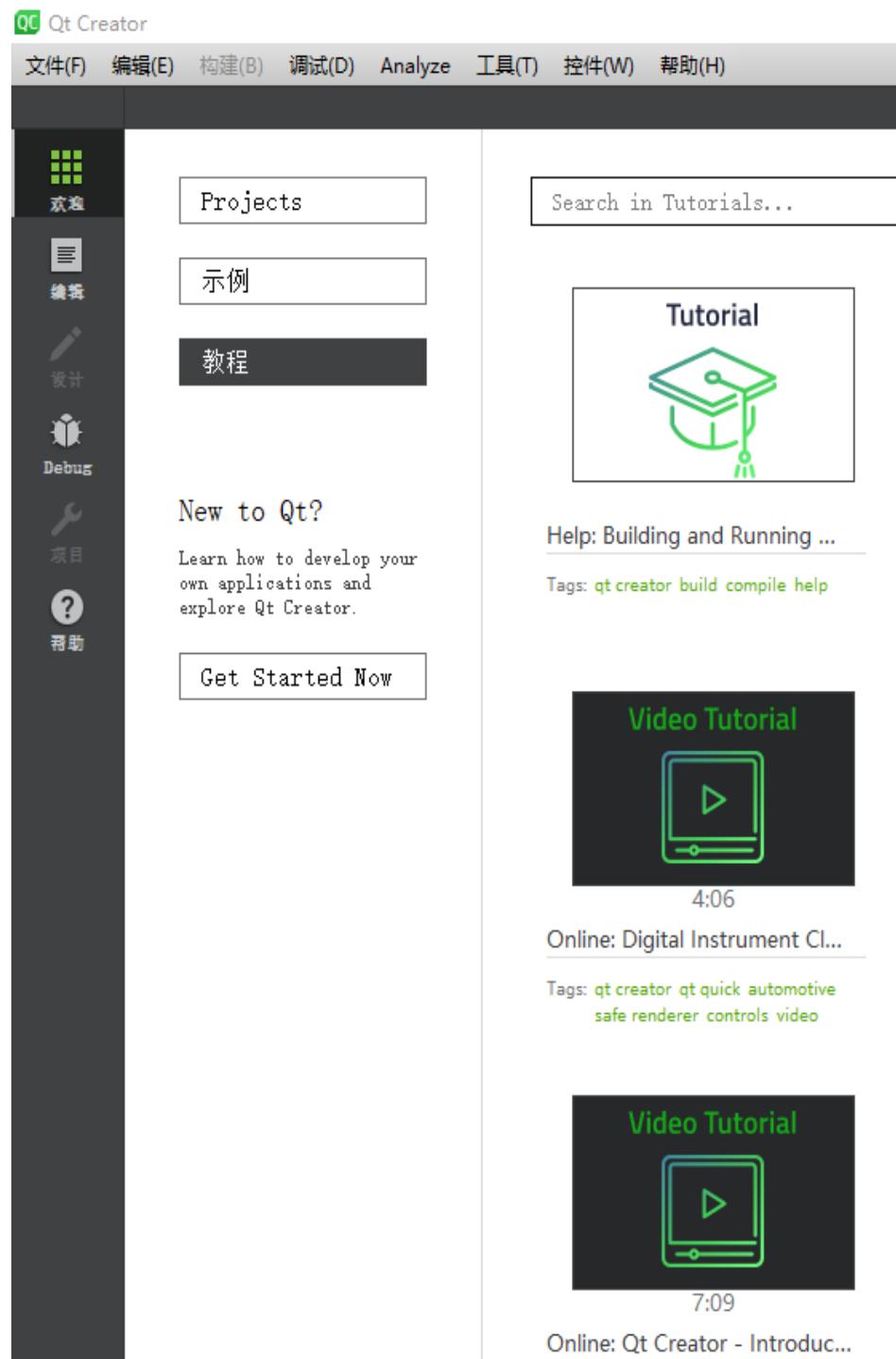
1. 拷贝ideaShare SDK 目录下的所有目录到ideaShareDemo 目录下，如图1-2所示

图1-2 导入SDK



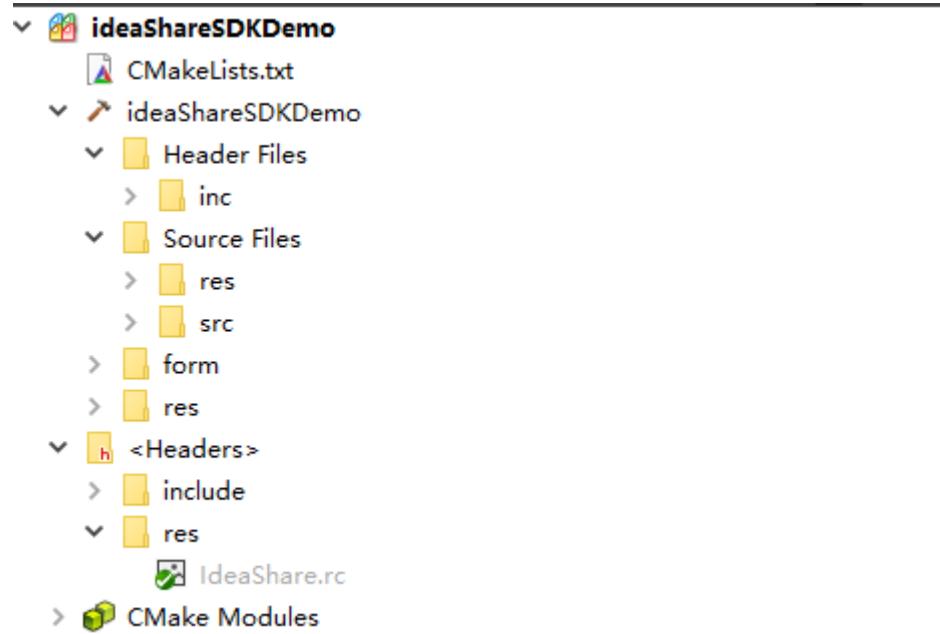
2 . 打开QtCreator, 如图1-3

图1-3 打开Qt Creator



3. 打开Demo示例工程，打开**ideaShareDemo** 目录下的CMakeList.txt, 如图1-4

图1-4 打开CMakeLists.txt



4、在QtCreator下编译运行Demo，如图1-5所示

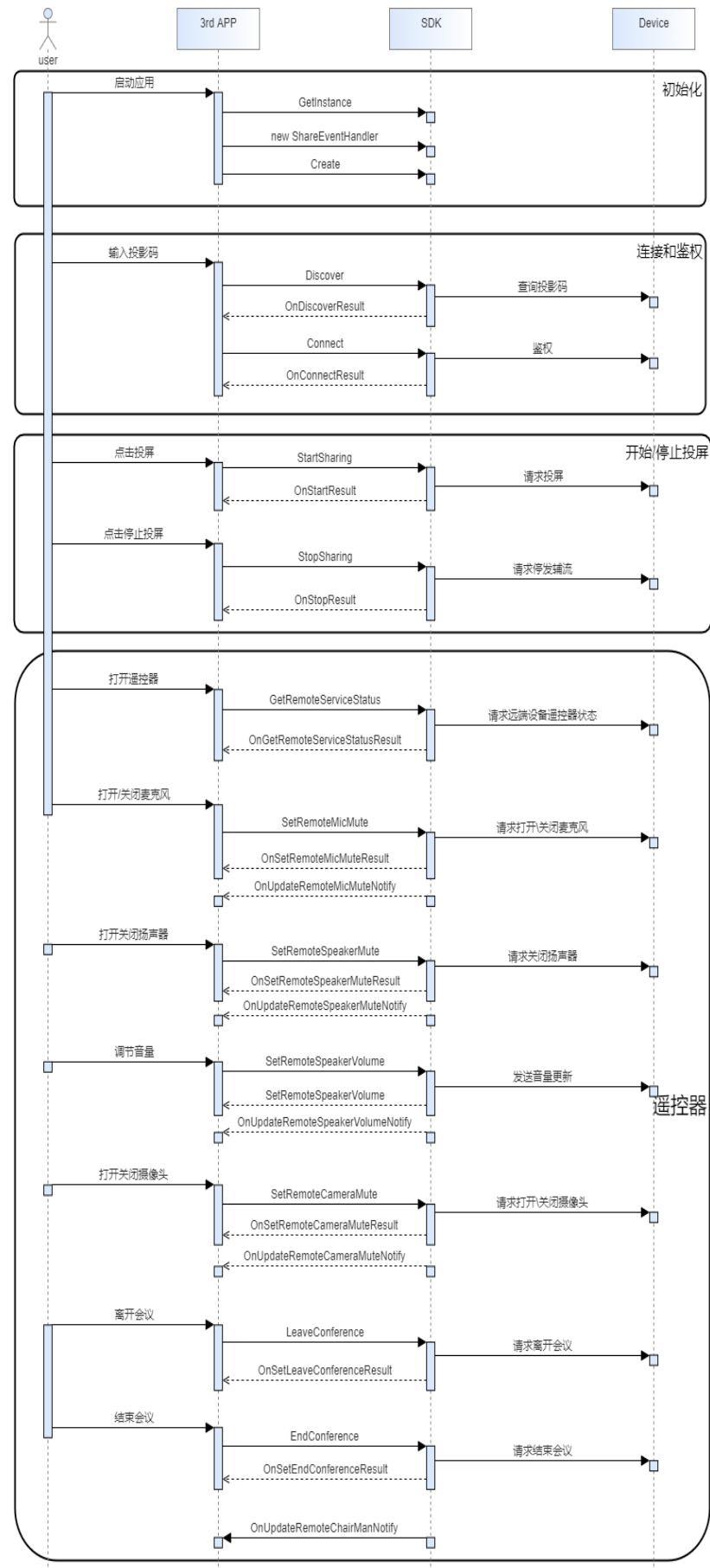
图1-5 编译运行demo



## 5.4 典型场景

### 5.4.1 典型场景接口调用概览

图1-6 接口调用流程



- 1、投屏APP 启动前必须调用初始化接口。
- 2、大部分接口调用结果通过异步回调实现，如xxxResult， 服务端通知事件格式xxxNotify。

## 5.4.2 场景 1： 初始化

### 描述

使用投屏SDK业务组件时，需要先完成SDK的基础组件的初始化。

### 业务流程

1. 调用ShareServiceController::GetInstance() 获取业务调用Controller  

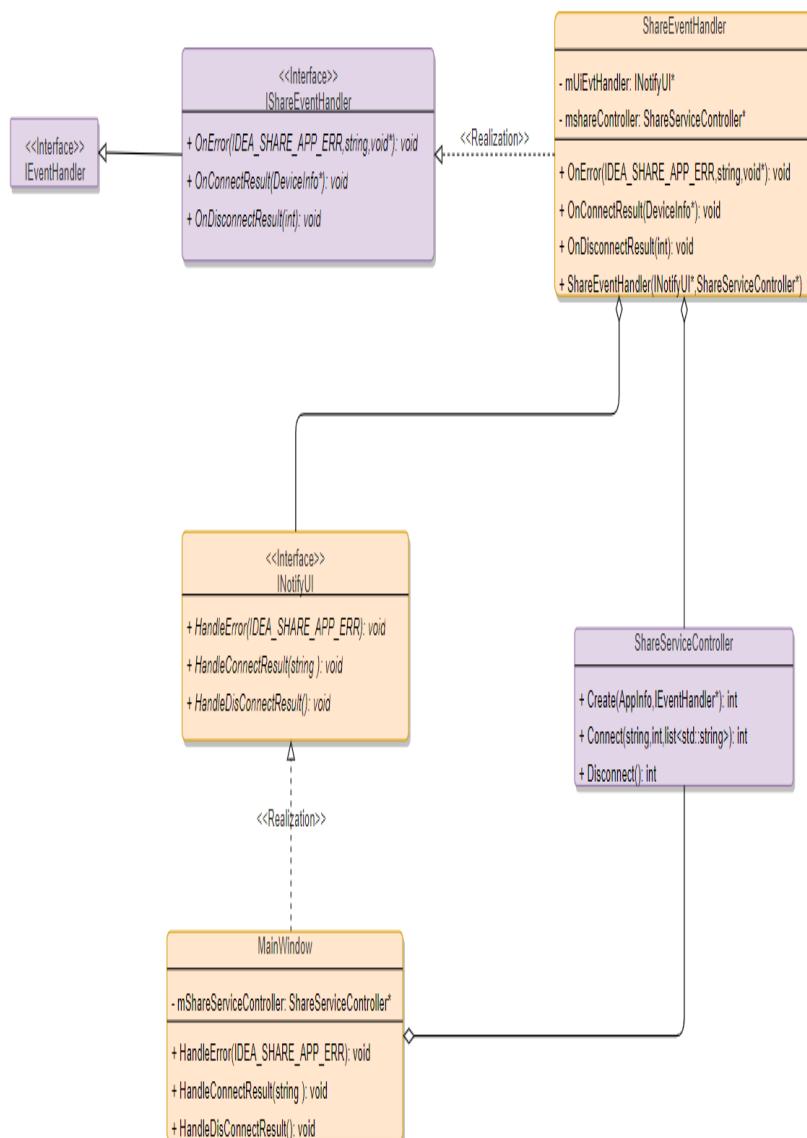
```
Huawei::Idea::ShareServiceController* mShareServiceController =  
Huawei::Idea::ShareServiceController::GetInstance()
```
2. 初始化日志  

```
const LogInfo logInfo = {LOG_PATH,IDEA_LEVEL_DEBUG,true};  
mShareServiceController->SetLog(logInfo);
```
3. 自定义事件回调类，继承自Huawei::Idea::IEventHandler, 其中构造函数INotifyUI入参是抽象类，可由UI 类继承实现，用作向UI通知消息  

```
class IDEA_SHARE_API IShareEventHandler : public  
Huawei::Idea::IEventHandler{ public: IShareEventHandler (INotifyUI*  
evtHandler, ShareServiceController* controller); ~IShareEventHandler (); void  
OnError(ShareAppErr errorCode, std::string errorMsg, void* extraInfo); void  
OnDiscoverResult(DiscoverResult* discoverResult); void  
OnConnectResult(DeviceInfo* deviceInfo); void OnDisconnectResult(int  
reason); .... }; class INotifyUI { public: virtual void  
HandleConnectResult(std::string deviceName) = 0; virtual void  
HandleDisConnectResult() = 0; virtual void HandleStartSharingResult() = 0;  
virtual void HandleStopSharingResult() = 0; .... };
```

类之间关系如下图所示：

图1-7 类之间关系



#### 4. 注册回调类，初始化SDK

```
const AppInfo appInfo = {};
mshareEvtHandler = new IShareEventHandler(this,mShareServiceController);
int ret = -1;
if (NULL != mshareEvtHandler){
    ret = mShareServiceController->Create(appInfo,mshareEvtHandler);
}
```

#### 注意事项

**IShareEventHandler**是用户自定义回调类，需继承**IEventHandler**，实现**IEventHandler**所有的回调方法。

### 5.4.3 场景 2：查询投屏码

#### 描述

用户输入投屏码后，返回连接设备IP和连接码

## 业务流程

1. 获取EUA地址和端口号，EUA地址为空则传入空字符串，端口号0
2. 调用 ShareServiceController的Discover 接口
3. 处理回调中的结果

## 注意事项

初始化SDK之后才能调用查询投屏码

## 示例代码

```
// 获取EUA地址
string address = SettingDialog::getEuaIP().toString();
uint32_t euaPort = SettingDialog::getEuaPort();
// 调用
if (mShareServiceController != NULL) {
    // text.toString()是string类型的投影码
    mShareServiceController->Discover(text.toString(),address,euaPort);
}
// 处理回调结果 IShareEventHandler类中
void IShareEventHandler::OnDiscoverResult(DiscoverResult* discoverResult)
{
    qDebug() << "receive onDiscover";
    if (mshareController == NULL) {
        qDebug() << "share controller is null";
        return;
    }
    // 调用连接接口
    mshareController->Connect(discoverResult->addressList,0,discoverResult->password);
}
```

## 5.4.4 场景 3： 连接/断开连接

### 描述

连接设备（IdeaHub）

## 业务流程

### 连接接口调用：

1. 调用Connect接口；
2. 处理连接回调结果。

### 断开接口调用：

1. 调用Disconnect接口；
2. 处理断开连接回调结果。

## 注意事项

必须在调用初始化和查询投屏码后才能调用此接口。

## 示例代码

```
//连接
if (mshareController == NULL) {
```

```
qDebug() << "share controller is null";
return;
}
mshareController->Connect(discoverResult->addressList,0,discoverResult->password);
// 连接结果回调
void IShareEventHandler::OnConnectResult(DeviceInfo* deviceInfo)
{
    qDebug() << "receive OnConnectResult";
    if (deviceInfo == NULL) {
        qDebug() << "deviceInfo is null, so return";
        return;
    }
    if (mUiEvtHandler != NULL) {
        // 刷新UI
        mUiEvtHandler->HandleConnectResult(deviceInfo->deviceName);
    }
}
// 断开连接
if (mShareServiceController != NULL) {
    mShareServiceController->Disconnect();
}
// 断开连接结果回调
void IShareEventHandler::OnDisconnectResult(int reason)
{
    qDebug() << "receive OnDisconnectResult";
    if (mUiEvtHandler != NULL) {
        mUiEvtHandler->HandleDisConnectResult();
    }
}
```

## 5.4.5 场景 4： 共享/停止投屏

### 描述

连接设备后，调用投屏接口发起投屏或调用停止投屏接口停止投屏。

### 业务流程

1. 调用StartSharing 或 StopSharing 启动或停止投屏。
2. 处理回调函数OnSharePlayResult和OnShareStopResult

### 注意事项

连接设备之后才能调用这两个接口。

### 示例代码

```
// 发起投屏
if (mShareServiceController != NULL) {
    mShareServiceController->StartSharing();
}
// 处理投屏结果回调
void IShareEventHandler::OnSharePlayResult()
{
    qDebug() << "receive OnSharePlayResult";
    if (mUiEvtHandler != NULL) {
        mUiEvtHandler->HandleStartSharingResult();
    }
}
// 停止投屏
if (mShareServiceController != NULL) {
    mShareServiceController->StopSharing();
}
// 处理停止投屏结果回调
```

```
void IShareEventHandler::OnShareStopResult(int reason)
{
    qDebug() << "receive OnShareStopResult";
    if (mUiEvtHandler != NULL) {
        mUiEvtHandler->HandleStopSharingResult();
    }
}
```

## 5.4.6 场景 5： 投放音频

### 描述

投屏前，投放音频选择投放或不投放音频

### 业务流程

调用SetPolicy 设置投放音频。

### 注意事项

连接设备之后投屏之前调用接口

### 示例代码

```
// 设置是否投放音频 true 投放 false 不投放
bool on = true;
if (mShareServiceController != NULL) {
    const SharePolicy sharePolicy = { QUAITY_CLEAR_FIRST, on};
    mShareServiceController->SetPolicy(sharePolicy);
}
```

## 5.4.7 场景 6： 设置投屏对象

### 描述

双屏情况下选择要投的是主屏还是扩展屏，不支持3屏

### 注意事项

连接设备之后投屏之前调用接口， 默认主屏

### 业务流程

调用SetDisplayScreen 设置投屏对象

```
// SetDisplayScreen参数入参是枚举类型 SCREEN_PRIMARY 0 主屏 SCREEN_SECONDARY 扩展屏，参见
IDEA_SHARE_APP_SCREEN_TYPE
ShareServiceController* controller = ShareServiceController::GetInstance();
if (controller != NULL) {
    controller->SetDisplayScreen(this->chooseShareScreen);
}
```

## 5.4.8 场景 7：遥控器

### 描述

在连接设备后，客户端可以查询设备状态，包括：麦克风状态、扬声器状态、摄像头状态、音量大小、是否在会议中，并且操作这些设备状态，如调节音量大小、离开会议（前提是已经入会）

### 业务流程

1. 调用GetRemoteServiceStatus查询连接设备状态
2. 处理回调结果事件OnGetRemoteServiceStatusResult
3. 调用遥控器接口（如SetRemoteMicMute）请求遥控远端设备
4. 处理回调事件OnSetRemoteMicMuteResult事件
5. 收到服务端主动推送过来的通知事件OnxxxxNotify事件，更新客户端遥控器状态
6. 在会议中时，调用LeaveConference或EndConference离开会议，处理回调事件结果OnSetLeaveConferenceResult或OnSetEndConferenceResult

### 注意事项

连接设备后遥控器功能才可用。

### 示例代码

```
// 查询远端设备状态
if (mShareServiceController != NULL) {
    mShareServiceController->GetRemoteServiceStatus();
}

// 处理OnGetRemoteServiceStatusResult回调
void IShareEventHandler::OnGetRemoteServiceStatusResult(RemoteServiceStatus *remoteServiceStatus)
{
    qDebug() << "receive OnGetRemoteServiceStatusResult";
    if (mUiEvtHandler != NULL) {
        mUiEvtHandler->HandleGetRemoteServiceStatusResult(remoteServiceStatus);
    }
}

// 设备控制，此处以麦克风静音和闭音为例，扬声器、摄像头、音量控制流程类似不再赘述
if (mShareServiceController != NULL) {
    // true 关闭， false 打开
    mShareServiceController->SetRemoteMicMute(false);
}

// 当前对结果事件暂未上报的UI，更新UI在OnUpdateRemoteMicMuteNotify中处理
void IShareEventHandler::OnSetRemoteMicMuteResult(int result)
{
    if (result != 0) {
        qDebug() << "receive set remote mic mute failed";
    }
}

// 远端设备主动推送OnUpdateRemoteMicMuteNotify事件通知UI进行更新
void IShareEventHandler::OnUpdateRemoteMicMuteNotify(bool isMute)
{
    if (mUiEvtHandler != NULL) {
        mUiEvtHandler->HandleUpdateMicMuteNotify(isMute);
    }
}

// 离开会议
if (mShareServiceController != NULL) {
    mShareServiceController->LeaveConference();
}

// 离开会议结果回调
```

```
void IShareEventHandler::OnSetLeaveConferenceResult(int result)
{
    qDebug() << "receive OnSetLeaveConferenceResult" << result;
    if (mUiEvtHandler != NULL) {
        mUiEvtHandler->HandleLeaveConference();
    }
}
// 结束会议
if (mShareServiceController != NULL) {
    mShareServiceController->EndConference();
}
// 结束会议结果回调
void IShareEventHandler::OnSetEndConferenceResult(int result)
{
    qDebug() << "receive OnSetEndConferenceResult result = " << result;
    if (mUiEvtHandler != NULL) {
        mUiEvtHandler->HandleEndConference();
    }
}
```

## 5.4.9 场景 8：设备推送

### 描述

远端设备定时向客户端推送投屏状态、辅流状态、会议状态，如果在会议中申请主席，还会推送主席状态

### 业务流程

1. 处理投屏状态、辅流状态、会议状态、主席状态

### 示例代码

```
// 投屏状态推送ShareAppServerStatus、ShareAppErr枚举值参见接口参考
void IShareEventHandler::OnShareStatusChangedNotify(ShareAppServerStatus status, ShareAppErr reason)
{
    if (mUiEvtHandler == NULL) {
        return;
    }
    // 服务端停发辅流
    if (status == SERVER_STATUS_STOP_SHARE) {
        mUiEvtHandler->HandleServerStopShare(reason);
    }
    // 服务端主动断开连接
    if (status == SERVER_STATUS_DISCONNECT) {
        mUiEvtHandler->HandleServerDisconnect(reason);
    }
}
// 辅流和会议状态推送，会议状态
void IShareEventHandler::OnDevConfStateNotify(ConfState confState, AuxState auxState)
{
    if (mUiEvtHandler != NULL) {
        mUiEvtHandler->HandleDevConfStatusNotify(confState, auxState);
    }
}
// 枚举值IdeaShareDefault.h中定义
typedef enum tagConfState {
    CONF_IDLE = 0,           // 空闲
    CONF_BUSY,               // 忙
} ConfState;

typedef enum tagAuxState {
    AUX_IDLE = 0,           // 空闲
    AUX_BUSY,                // 忙
} AuxState;
```

```
// 设备在会议中申请主席后，设备将推送主席状态
void IShareEventHandler::OnUpdateRemoteChairManNotify(bool isChairMan)
{
    if (mUiEvtHandler != NULL) {
        mUiEvtHandler->HandleUpdateChairManNotify(isChairMan);
    }
}
```

## 5.4.10 场景 9：错误上报

### 描述

在使用过程中，出现异常时，SDK上报OnError事件

### 业务流程

处理OnError 事件回调

### 示例代码

```
// 错误信息上报 ShareAppErr 类型参照接口参考
void IShareEventHandler::OnError(ShareAppErr errorCode, std::string errorMsg, void* extraInfo)
{
    if (mUiEvtHandler != NULL) {
        mUiEvtHandler->HandleError(errorCode);
    }
}
```

## 5.4.11 场景 10：去初始化

### 描述

退出客户端APP时，调用去初始化SDK接口

### 业务流程

调用去初始化SDK接口

### 示例代码

```
// 去初始化SDK
if (mShareServiceController ) {
    mShareServiceController->Destory();
}
```

## 5.5 常见问题

### 5.5.1 获取日志

- 进入exe文件所在目录下，如下图所示，日志log目录与 exe文件同级。

图2-1 log目录

此电脑 > Windows (C:) > Program Files (x86) > ideaShareSDKDemo			
名称	修改日期	类型	大小
fr_exception	2021/1/13 17:03	文件夹	
iconengines	2021/1/13 17:03	文件夹	
imageformats	2021/1/13 17:03	文件夹	
log	2021/1/13 17:03	文件夹	
platforms	2021/1/13 17:03	文件夹	
styles	2021/1/13 17:03	文件夹	
translations	2021/1/13 17:03	文件夹	
AudioDLL.dll	2021/1/13 16:02	应用程序扩展	115 KB
d3dcompiler_47.dll	2017/3/30 0:38	应用程序扩展	3,596 KB
data_projection_client_windows.dll	2021/1/13 16:07	应用程序扩展	1,918 KB
FaultReport.exe	2021/1/13 16:07	应用程序	1,039 KB
fr_config.ini	2021/1/13 16:07	配置设置	4 KB
fr_lang.ini	2021/1/13 16:07	配置设置	12 KB
fr_plugin.dll	2021/1/13 16:07	应用程序扩展	363 KB
HME_Video.dll	2021/1/13 16:04	应用程序扩展	3,437 KB
HME_Video_H264D.dll	2021/1/13 16:01	应用程序扩展	320 KB
HME_Video_H264E.dll	2021/1/13 16:01	应用程序扩展	547 KB
HME_Video_SrtP_ALG.dll	2021/1/13 16:02	应用程序扩展	96 KB
idea_share_sdk_wrapper.dll	2021/1/13 16:08	应用程序扩展	53 KB
ideaShareSDKDemo.exe	2021/1/13 16:11	应用程序	967 KB
ideaShareSDKDemo.exp	2021/1/13 16:11	EXP 文件	11 KB
ideaShareSDKDemo.lib	2021/1/13 16:11	Object File Library	19 KB
libEGL.dll	2020/3/27 21:20	应用程序扩展	22 KB
libGLESv2.dll	2020/3/27 21:20	应用程序扩展	2,791 KB
opengl32sw.dll	2016/6/14 21:08	应用程序扩展	15,621 KB
Qt5Core.dll	2021/1/13 16:11	应用程序扩展	5,175 KB
Qt5Gui.dll	2021/1/13 16:11	应用程序扩展	5,839 KB
Qt5Network.dll	2021/1/13 16:10	应用程序扩展	1,090 KB

## 2. 查看日志

此电脑 > Windows (C:) > Program Files (x86) > ideaShareSDKDemo > log			
名称	修改日期	类型	大小
hme_v_log_all.txt	2021/1/13 17:03	文本文档	10 KB
hme_v_log_api.txt	2021/1/13 17:03	文本文档	4 KB
hme_v_log_error.txt	2021/1/13 17:03	文本文档	1 KB
hme_v_log_io.txt	2021/1/13 17:03	文本文档	0 KB
hme_v_log_trace.txt	2021/1/13 17:03	文本文档	0 KB
IdeaShare.log	2021/1/13 17:03	文本文档	1 KB
ideashare_wrapper.log	2021/1/13 17:03	文本文档	2 KB
projection_client.log	2021/1/13 17:03	文本文档	16 KB
tup_mediaservice.log	2021/1/13 17:03	文本文档	4 KB
tup_video.log	2021/1/13 17:03	文本文档	4 KB
tup_videotrace.log	2021/1/13 17:03	文本文档	1 KB

## 5.6 Windows 接口参考 ( C++ )

### 5.6.1 概述

本文主要介绍投屏服务 C++ 接口参考。主要包含接口函数定义、参数定义说明、附录，事件上报通知说明以及错误码等信息。

### 5.6.2 变更记录

表1-1 变更记录

日期	版本	变更内容
2021-02-07	1.0.0	初稿完成。
2021-5-15	2.0.0	变更接口，完善细节内容

### 5.6.3 接口参考

#### 5.6.3.1 基础配置

##### 5.6.3.1.1 获取 ShareServiceController 单例

获取ShareServiceController 单例。

```
Huawei::Idea::ShareServiceController* mShareServiceController =  
Huawei::Idea::ShareServiceController::GetInstance();
```

##### 接口描述

获取ShareServiceController单例。

##### 注意事项

应用程序调用接口前都获取ShareServiceController 单例，建议获取一次后在应用程序中持有这个单例。

##### 接口定义

```
static ShareServiceController* GetInstance();
```

#### 5.6.3.1.2 初始化日志

初始化日志。

```
const LogInfo logInfo = {LOG_PATH,IDEA_LEVEL_DEBUG,true};  
mShareServiceController->SetLog(logInfo);
```

##### 接口描述

初始化日志前获取ShareServiceController 对象，然后调用ShareServiceController的 SetLog。

### 注意事项

请传入正确的LogInfo对象，初始化接口在初始化SDK之前定义。

### 接口定义

```
/* 日志设置 */
int SetLog(const LogInfo &logInfo);
```

### 参数描述

参数	是否必须	类型	描述
logInfo	是	<a href="#">4.6.5.2 LogInfo&amp;</a>	日志对象。

### 5.6.3.1.3 自定义通知回调类

#### 示例代码

继承Huawei::Idea::IEventHandler，参见IEventHandler.h

IShareServiceController.h

```
#include "IShareServiceController.h"
#include "IdeaShareDef.h"

class IDEA_SHARE_API IShareEventHandler : public Huawei::Idea::IEventHandler{
public:
    ~IShareEventHandler();
    void OnError(ShareAppErr errorCode, std::string errorMsg, void* extrallInfo);
    void OnDiscoverResult(DiscoverResult* discoverResult);
    void OnConnectResult(DeviceInfo* deviceInfo);
    void OnDisconnectResult(int reason);
    void OnSharePlayResult();
    void OnShareStopResult(int reason);
    ...
}
```

ShareEventHandler.cpp 实现

```
#include "IShareServiceController.h"
#include "IdeaShareDef.h"

IShareEventHandler::~IShareEventHandler() {}

void IShareEventHandler::OnError(ShareAppErr errorCode, std::string errorMsg, void* extrallInfo
{
    // 错误事件上报
}

void IShareEventHandler::OnDiscoverResult(DiscoverResult* discoverResult)
{
    // 投屏码查询
}

void IShareEventHandler::OnConnectResult(DeviceInfo* deviceInfo)
{
    // 连接事件上报
}

void IShareEventHandler::OnDisconnectResult(int reason)
{
    // 断开连接事件上报
}

void IShareEventHandler::OnSharePlayResult()
{
```

```
// 开始共享事件上报
}
void IShareEventHandler::OnShareStopResult(int reason)
{
    //结束共享事件上报
}
....
```

详情参见[4.6.4 通知参考](#)。

#### 5.6.3.1.4 初始化 SDK

注册通知回调类，并初始化SDK。

```
const AppInfo applInfo = {};
mshareEvtHandler = new IShareEventHandler();
int ret = -1;
if (NULL != mshareEvtHandler){
    ret = mShareServiceController->Create(applInfo,mshareEvtHandler);
}
```

##### 接口描述

注册接口回调，并初始化SDK。

##### 注意事项

mshareEvtHandler 是自定义IShareEventHandler的对象。

##### 接口定义

```
/* 初始化 */
int Create(const AppInfo &applInfo, IEventHandler* eventHandler);
```

IEventHandler 定义参见[4.6.5.3.1 IEventHandler.h](#)。

#### 5.6.3.1.5 去初始化 SDK

应用程序退出时应去初始化SDK。

```
// 去初始化SDK
if (mShareServiceController ) {
    mShareServiceController->Destory();
}
```

##### 接口描述

去初始化SDK。

##### 接口定义

```
/* 去初始化 */
int Destory();
```

#### 5.6.3.2 连接设备

##### 5.6.3.2.1 设备发现

###### 接口描述

该接口用于根据投屏码解析得到设备IP和设备连接码。

###### 方法定义

```
/* 投屏码解析ip请求 */
int Discover(std::string castCode, std::string address, int port);
```

### 参数描述

参数	是否必须	类型	描述
castCode	是	std::string	投影码
address	是	std::string	SMC(EUA) IP地址，在线解析时填入EUA IP地址，本地解析传""
port	是	int	SMC(EUA) 端口号。在线解析时填入EUA 端口号，本地解析时传0

### 异步回调事件结果

参见[4.6.4.1.1 OnDiscoverResult](#)。

### 示例代码

```
// 获取EUA地址
string address = SettingDialog::getEuaIP().toStdString();
uint32_t euaPort = SettingDialog::getEuaPort();
// 调用
if (mShareServiceController != NULL) {
    // text.toStdString()是string类型的投影码
    mShareServiceController->Discover(text.toStdString(),address,euaPort);
}
```

## 5.6.3.2.2 TLS 证书校验设置

### 接口描述

该接口用于设置TLS证书校验的相关设置。

### 方法定义

```
/* TLS设置 */
int SetTLS(const ShareTlsInfo &tlsInfo);
```

### 参数描述

参数	是否必须	类型	描述
tlsInfo	否	<a href="#">4.6.5.2.7 ShareTlsInfo &amp;</a>	设备发现结果返回的设备IP列表

### 异步回调事件结果

参见[4.6.4.1.2 OnConnectFailed](#)，如果设置了TLS证书路径信息，则会在connect的过程中去校验，如果校验通过，则不会有回调，如果校验不通过，则回调OnConnectFailed。

### 示例代码

```
//连接
if (mshareController == NULL) {
    return;
}
mshareController->SetTLS(&tlsInfo);
```

## 5.6.3.2.3 连接

### 接口描述

该接口用于连接设备。

### 方法定义

```
/* 连接设备 */
int Connect(list<std::string>addressList, int port, std::string password);
```

### 参数描述

参数	是否必须	类型	描述
addressList	是	list<std::string>	设备发现结果返回的设备IP列表
port	是	int	设备连接端口号， 默认0
password	是	std::string	设备连接码

### 异步回调事件结果

参见[4.6.4.1.2 OnConnectResult](#)。

### 示例代码

```
//连接
if (mshareController == NULL) {
    return;
}
mshareController->Connect(discoverResult->addressList,0,discoverResult->password);
```

## 5.6.3.2.4 断开连接

### 接口描述

该接口用于断开连接设备。

### 方法定义

```
/* 断开连接设备 */
int Disconnect();
```

### 参数描述

NA

### 异步回调事件结果

参见[4.6.4.1.3 OnDisconnectResult](#)。

### 示例代码

```
// 断开连接
if (mShareServiceController != NULL) {
    mShareServiceController->Disconnect();
}
```

### 5.6.3.3 投屏

#### 5.6.3.3.1 开始投屏

##### 接口描述

该接口用于共享投屏。

##### 注意事项

连接后，才能调用开始投屏接口。注意需要调用[5.6.3.3.3 设置投屏策略](#)将兼容性设置为true。

##### 接口定义

```
/* 开始投屏 */
int StartSharing();
```

##### 参数描述

NA

##### 返回值

0 接口调用成功 非0 接口调用失败

##### 异步回调事件结果

参见[4.6.4.1.4 OnSharePlayResult](#)。

##### 示例代码

```
// 发起投屏
if (mShareServiceController != NULL) {
    mShareServiceController->StartSharing();
}
```

#### 5.6.3.3.2 停止投屏

##### 接口描述

该接口用于停止投屏。

##### 注意事项

调用开始投屏后，才能调用停止投屏接口。

##### 接口定义

```
/* 停止投屏 */
int StopSharing();
```

##### 参数描述

NA

##### 返回值

0 接口调用成功 非0 接口调用失败

### 异步回调事件结果

参见[4.6.4.1.5 OnShareStopResult](#)。

#### 示例代码

```
// 停止投屏
if (mShareServiceController != NULL) {
    mShareServiceController->StopSharing();
}
```

### 5.6.3.3 设置投屏策略

#### 接口描述

该接口会设置是否开启兼容性，用于兼容现网CBC加密模式。注意：当前现网均是CBC加密模式，此接口必须设置成开启兼容性。

同时会用于在开始投屏前设置投屏效果的清晰度以及是否需要投放音频到设备。

#### 注意事项

调用开启投屏接口之前或者投屏过程中调用。

#### 接口定义

```
/* 设置投屏策略 */
int SetPolicy(const SharePolicy& sharePolicy);
```

#### 参数描述

参数	是否必须	类型	描述
sharePolicy	是	<a href="#">4.6.5.2.3 SharePolicy &amp;</a>	设置投屏测试，包括是否开启音频，视频流畅度以及是否开启兼容性

#### 返回值

0 接口调用成功 非0 接口调用失败

#### 示例代码

```
// 设置投屏效果 FLUENCY_FIRST: 流畅优先 QUAITY_CLEAR_FIRST: 清晰优先 QUAITY_ADAPTIVE: 自适应
// 设置是否投放音频 true 投放 false 不投放
bool on = true;
if (mShareServiceController != NULL) {
    const SharePolicy sharePolicy = { QUAITY_CLEAR_FIRST, on, true };
    mShareServiceController->SetPolicy(sharePolicy);
}
```

### 5.6.3.4 设置投屏对象

#### 接口描述

该接口用于在开始投屏前设置主屏或扩展屏。

#### 注意事项

调用开启投屏接口之前调用。

#### 接口定义

```
/* 设置投屏对象 */
int SetDisplayScreen(ShareAppScreenType screenChoice);
```

#### 参数描述

参数	是否必须	类型	描述
screenChoice	是	<a href="#">4.6.5.1.3 ShareAppScreenType</a>	设置时投主屏还是扩展屏 SCREEN_PRIMARY 0 主屏 默认值 SCREEN_SECONDARY 1 扩展屏

#### 返回值

0 接口调用成功 非0 接口调用失败

#### 示例代码

```
ShareServiceController* controller = ShareServiceController::GetInstance();
if (controller != NULL) {
    controller->SetDisplayScreen(this->chooseShareScreen);
}
```

### 5.6.3.3.5 设置投屏设备类型

#### 接口描述

该接口用于设置投屏设备类型。

#### 注意事项

调用开启投屏接口之前调用。

#### 接口定义

```
/* 设置投屏设备类型 */
int SetDeviceType(ShareDeviceType shareDeviceType);
```

#### 参数描述

参数	是否必须	类型	描述
shareDeviceType	是	<a href="#">4.6.5.1.9 ShareDeviceType</a>	投屏设备类型

#### 返回值

0 接口调用成功 非0 接口调用失败

#### 示例代码

```
ShareServiceController* controller = ShareServiceController::GetInstance();
if (controller != NULL) {
    controller->SetDeviceType(this->shareDeviceType);
}
```

### 5.6.3.3.6 获取投屏状态

#### 接口描述

该接口用于获取当前投屏的状态。

#### 注意事项

/

#### 接口定义

```
/* 获取投屏状态 */  
int GetShareStatus();
```

#### 参数描述

NA

#### 返回值

代表了当前的投屏状态，详见[4.6.5.1.10](#)

#### 示例代码

```
ShareServiceController* controller = ShareServiceController::GetInstance();  
if (controller != NULL) {  
    controller->GetShareStatus();  
}
```

### 5.6.3.3.7 设置鼠标样式

#### 接口描述

该接口用于设置投屏后在大屏侧显示的鼠标样式。

#### 注意事项

/

#### 接口定义

```
/* 设置鼠标样式 */  
int SetMouseShape(ShareMouseShape mouseShape);
```

#### 参数描述

参数	是否必须	类型	描述
mouseShape	是	<a href="#">4.6.5.1.11</a>	设置的鼠标样式

#### 返回值

0 接口调用成功 非0 接口调用失败

#### 示例代码

```
ShareServiceController* controller = ShareServiceController::GetInstance();  
if (controller != NULL) {  
    ShareMouseShape mouseShape = SHARE_MOUSE_SHAPE_WHITE_ARROW; // 白色箭头鼠标  
    controller->SetMouseShape(mouseShape);  
}
```

### 5.6.3.3.8 获取窗口句柄信息

### 5.6.3.4 遥控器

#### 5.6.3.4.1 查询远端设备状态

##### 接口描述

该接口用于查询远端设备麦克风、扬声器、扬声器音量、摄像头、会议状态。

##### 注意事项

连接设备后使用。

##### 接口定义

```
/* 获取远端设备状态信息 */  
int GetRemoteServiceStatus();
```

##### 参数描述

NA

##### 返回值

0 接口调用成功 非0 接口调用失败

##### 异步回调事件结果

参见[4.6.4.1.6 OnGetRemoteServiceStatusResult](#)。

##### 示例代码

```
// 查询远端设备状态  
if (mShareServiceController != NULL) {  
    mShareServiceController->GetRemoteServiceStatus();  
}
```

#### 5.6.3.4.2 打开/关闭麦克风

##### 接口描述

该接口用于控制服务端麦克风打开或关闭。

##### 注意事项

连接设备后使用。

##### 接口定义

```
/* 打开\关闭麦克风 */  
int SetRemoteMicMute(bool isMute);
```

##### 参数描述

参数	是否必须	类型	描述
isMute	是	bool	true 麦克风静音 false 麦克风打开

### 返回值

0 接口调用成功 非0 接口调用失败

### 异步回调事件结果

参见[4.6.4.1.7 OnSetRemoteMicMuteResult](#)和[4.6.4.2.1 OnRemoteServiceStatusChangedNotify](#)。

### 示例代码

```
// 打开麦克风
if (mShareServiceController != NULL) {
    // true 关闭 false 打开
    mShareServiceController->SetRemoteMicMute(false);
}
```

## 5.6.3.4.3 打开/关闭扬声器

### 接口描述

该接口用于控制服务端扬声器打开或关闭。

### 注意事项

连接设备后使用。

### 接口定义

```
/* 打开\关闭扬声器 */
int SetRemoteSpeakerMute(bool isSpeakerMute);
```

### 参数描述

参数	是否必须	类型	描述
isSpeakerMute	是	bool	true 扬声器静音 false 扬声器打开

### 返回值

0 接口调用成功 非0 接口调用失败

### 异步回调事件结果

参见[4.6.4.1.9 OnSetRemoteSpeakerMuteResult](#)和[4.6.4.2.1 OnRemoteServiceStatusChangedNotify](#)。

### 示例代码

```
// 扬声器静音
if (mShareServiceController != NULL) {
    mShareServiceController->SetRemoteSpeakerMute(true);
}
```

## 5.6.3.4.4 打开/关闭摄像头

### 接口描述

该接口用于控制服务端摄像头打开关闭。

### 注意事项

连接设备后使用。

### 接口定义

```
/* 打开\关闭摄像机 */
int SetRemoteCameraMute(bool isCameraMute);
```

#### 参数描述[4.6.4.2.1](#)

参数	是否必须	类型	描述
isCameraMute	是	bool	true 摄像头关闭 false 摄像头打开

### 返回值

0 接口调用成功 非0 接口调用失败

### 异步回调事件结果

参见[4.6.4.1.10 OnSetRemoteCameraMuteResult](#)和[4.6.4.2.1 OnRemoteServiceStatusChangedNotify](#)。

### 示例代码

```
// 摄像头关闭
if (mShareServiceController != NULL) {
    mShareServiceController->SetRemoteCameraMute(true);
}
```

## 5.6.3.4.5 调节扬声器音量

### 接口描述

该接口用于控制扬声器音量大小。

### 注意事项

连接设备后使用。

### 接口定义

```
/* 调节扬声器音量 */
int SetRemoteSpeakerVolume(int volume);
```

#### 参数描述

参数	是否必须	类型	描述
volume	是	int	音量值大小，范围 0~21

### 返回值

0 接口调用成功 非0 接口调用失败

### 异步回调事件结果

参见[4.6.4.1.8 OnSetRemoteSpeakerVolumeResult](#)和[4.6.4.2.1 OnRemoteServiceStatusChangedNotify](#)。

#### 示例代码

```
// 设置音量值为16
int volume = 16;
if (mShareServiceController != NULL) {
    mShareServiceController->SetRemoteSpeakerVolume(volume);
}
```

### 5.6.3.4.6 离开会议

#### 接口描述

该接口用于控制设备离开会议。

#### 注意事项

已连接设备，并且设备在会议中。

#### 接口定义

```
/* 离开会议 */
int LeaveConference();
```

#### 参数描述

NA

#### 返回值

0 接口调用成功 非0 接口调用失败

#### 异步回调事件结果

参见[4.6.4.1.11 OnSetLeaveConferenceResult](#)。

#### 示例代码

```
// 离开会议
if (mShareServiceController != NULL) {
    mShareServiceController->LeaveConference();
}
```

### 5.6.3.4.7 结束会议

#### 接口描述

该接口用于控制设备结束会议。

#### 注意事项

已连接设备，并且设备在会议中，设备账号是主席身份。

#### 接口定义

```
/* 结束会议 */
int EndConference();
```

#### 参数描述

NA

#### 返回值

0 接口调用成功 非0 接口调用失败

### 异步回调事件结果

参见[4.6.4.1.12 OnSetEndConferenceResult。](#)

### 示例代码

```
// 结束会议
if (mShareServiceController != NULL) {
    mShareServiceController->EndConference();
}
```

## 5.6.3.5 版本信息查询

### 5.6.3.5.1 获取 SDK 版本信息

#### 接口描述

该接口用于获取本地客户端SDK版本信息。

#### 注意事项

连接设备后使用。

#### 接口定义

```
/* 获取SDK版本 */
std::string GetSDKVersion();
```

#### 参数描述

参数	是否必须	类型	描述
NA	否	NA	NA

#### 返回值

std::string类型，内容为当前SDK版本信息

#### 异步回调事件结果

同步调用接口，无异步回调事件接口。

#### 示例代码

```
// 获取当前SDK版本信息
std::string sdkVersionInfo;
if (mShareServiceController != NULL) {
    sdkVersionInfo = mShareServiceController->GetSDKVersion();
}
```

## 5.6.3.5.2 获得服务端版本信息

#### 接口描述

该接口用于获得服务端版本信息。

#### 注意事项

连接设备后使用。

## 接口定义

```
/* 获取最新的服务端版本信息 */
int GetSDKLatestVersion();
```

## 参数描述

参数	是否必须	类型	描述
NA	否	NA	NA

## 返回值

0 接口调用成功 非0 接口调用失败

## 异步回调事件结果

参见OnGetSDKLatestVersionResult。

## 示例代码

```
// 设置音量值为16
str::string sdkVersionInfo;
if (mShareServiceController != NULL) {
    mShareServiceController->GetSDKLatestVersion();
}
```

## 5.6.4 通知参考

### 5.6.4.1 接口调用结果通知

#### 5.6.4.1.1 OnDiscoverResult

##### 回调方法描述

设备发现接口回调结果事件。

##### 回调方法定义

```
/* 设备发现结果事件 */
void OnDiscoverResult(DiscoverResult* discoverResult) { }
```

##### 参数描述

参数	是否必须	类型	描述
discoverResult	是	4.6.5.2.4 DiscoverResult*	设备发现结果结构体

##### 示例代码

```
void ShareEventHandler::OnDiscoverResult(DiscoverResult* discoverResult)
{
    // 设备发现结果事件处理
}
```

### 5.6.4.1.2 OnConnectResult

#### 回调方法描述

连接接口回调结果事件。

#### 回调方法定义

```
/* 连接结果事件 */
void OnConnectResult(DeviceInfo* deviceInfo) {}
```

#### 参数描述

参数	是否必须	类型	描述
deviceInfo	是	<a href="#">4.6.5.2.5 DeviceInfo*</a>	远端设备信息

#### 示例代码

```
void ShareEventHandler::OnConnectResult(DeviceInfo* deviceInfo)
{
    // 上报UI显示设备名
}
```

### 5.6.4.1.3 OnDisconnectResult

#### 回调方法描述

断开连接接口回调结果事件。

#### 回调方法定义

```
/* 断开连接结果事件 */
void OnDisconnectResult(int reason) {}
```

#### 参数描述

参数	是否必须	类型	描述
reason	是	int	暂无实际作用

#### 示例代码

```
void ShareEventHandler::OnDisconnectResult(int reason)
{
    // 通知UI刷新
}
```

### 5.6.4.1.4 OnSharePlayResult

#### 回调方法描述

开始投屏接口回调结果事件。

#### 回调方法定义

```
/* 开始投屏结果事件通知 */
void OnSharePlayResult() {}
```

### 参数描述

NA

### 示例代码

```
void ShareEventHandler::OnSharePlayResult()
{
    // 开始投屏结果事件，通知UI刷新
}
```

## 5.6.4.1.5 OnShareStopResult

### 回调方法描述

停止投屏接口回调结果事件。

### 回调方法定义

```
/* 停止投屏结果事件通知 */
void OnShareStopResult(int reason)
```

### 参数描述

参数	是否必须	类型	描述
reason	是	int	停止投屏原因值

### 示例代码

```
void ShareEventHandler::OnShareStopResult()
{
    // 停止投屏结果事件，通知UI刷新
}
```

## 5.6.4.1.6 OnGetRemoteServiceStatusResult

### 回调方法描述

查询远端设备状态回调结果事件。

### 回调方法定义

```
/* 查询远端设备状态结果事件通知 */
void OnGetRemoteServiceStatusResult(RemoteServiceStatus* remoteServiceStatus){}
```

### 参数描述

参数	是否必须	类型	描述
remoteServiceStatus	是	<a href="#">4.6.5.2.6 RemoteServiceStatus</a> *	远端设备麦克风、扬声器、摄像头等的状态

### 示例代码

```
void ShareEventHandler::OnGetRemoteServiceStatusResult(RemoteServiceStatus *remoteServiceStatus)
{
    // 根据设备状态刷新UI
}
```

### 5.6.4.1.7 OnSetRemoteMicMuteResult

#### 回调方法描述

设置麦克风静音回调结果事件。

#### 回调方法定义

```
/* 设置麦克风静音结果事件通知 */
void OnSetRemoteMicMuteResult(int result) {}
```

#### 参数描述

参数	是否必须	类型	描述
result	是	int	1 成功 非1 失败

#### 示例代码

```
void ShareEventHandler::OnSetRemoteMicMuteResult(int result)
{
    // 刷新UI
}
```

### 5.6.4.1.8 OnSetRemoteSpeakerVolumeResult

#### 回调方法描述

设置扬声器音量结果事件。

#### 回调方法定义

```
/* 设置扬声器音量结果事件通知 */
void OnSetRemoteSpeakerVolumeResult(int result) {}
```

#### 参数描述

参数	是否必须	类型	描述
result	是	int	1 成功 非1 失败

#### 示例代码

```
void ShareEventHandler::OnSetRemoteSpeakerVolumeResult(int result)
{
    // 事件处理
}
```

### 5.6.4.1.9 OnSetRemoteSpeakerMuteResult

#### 回调方法描述

设置扬声器打开关闭结果事件。

#### 回调方法定义

```
/* 设置扬声器打开关闭结果事件通知 */
void OnSetRemoteSpeakerMuteResult(int result) {}
```

### 参数描述

参数	是否必须	类型	描述
result	是	int	1 成功 非1 失败

### 示例代码

```
void ShareEventHandler::OnSetRemoteSpeakerMuteResult(int result)
{
    // 事件处理
}
```

## 5.6.4.1.10 OnSetRemoteCameraMuteResult

### 回调方法描述

设置摄像头打开关闭结果事件。

### 回调方法定义

```
/* 设置摄像头打开关闭结果事件通知 */
void OnSetRemoteCameraMuteResult(int result) {}
```

### 参数描述

参数	是否必须	类型	描述
result	是	int	1 成功 非1 失败

### 示例代码

```
void ShareEventHandler::OnSetRemoteCameraMuteResult(int result)
{
    // 事件处理
}
```

## 5.6.4.1.11 OnSetLeaveConferenceResult

### 回调方法描述

离开会议结果事件。

### 回调方法定义

```
/* 离开会议结果事件通知 */
void OnSetLeaveConferenceResult(int result) {}
```

### 参数描述

参数	是否必须	类型	描述
result	是	int	1 成功 非1 失败

### 示例代码

```
void ShareEventHandler::OnSetLeaveConferenceResult(int result)
{
    // 事件处理
}
```

#### 5.6.4.1.12 OnSetEndConferenceResult

##### 回调方法描述

结束会议结果事件。

##### 回调方法定义

```
/* 结束会议结果事件通知 */
void OnSetEndConferenceResult(int result) {}
```

##### 参数描述

参数	是否必须	类型	描述
result	是	int	1 成功 非1 失败

##### 示例代码

```
void ShareEventHandler::OnSetEndConferenceResult(int result)
{
    // 事件处理
}
```

#### 5.6.4.1.13 OnGetSDKLatestVersionResult

##### 回调方法描述

获取服务端版本信息回调事件。

##### 回调方法定义

```
/* 获取服务端版本信息回调事件 */
void OnGetSDKLatestVersionResult(std::string version) {}
```

##### 参数描述

参数	是否必须	类型	描述
version	否	std::string	服务端版本信息内容

##### 示例代码

```
void ShareEventHandler::OnGetSDKLatestVersionResult(std::string version)
{
    // UI界面对获取到的信息进行处理,比如显示版本号以及比对是否需要更新证书等
}
```

#### 5.6.4.1.14 OnSetMouseShapeResult

##### 回调方法描述

设置鼠标样式结果回调。

### 回调方法定义

```
/* 设置鼠标样式结果事件通知 */
void OnSetMouseShapeResult(int result) {}
```

### 参数描述

参数	是否必须	类型	描述
result	是	int	1 成功 非1 失败

### 示例代码

```
void ShareEventHandler::OnSetMouseShapeResult(int result)
{
    // 事件处理
}
```

## 5.6.4.2 设备推送通知

### 5.6.4.2.1 OnRemoteServiceStatusChangedNotify

#### 回调方法描述

IdeaHub侧麦克风、摄像头、音量等状态信息发生了变化后会主动触发告知上层的回调事件。

#### 回调方法定义

```
/* IdeaHub设备自身状态变化回调 */
void OnRemoteServiceStatusChangedNotify(RemoteServiceStatus *remoteServiceStatus, int index) {}
```

#### 参数描述

参数	是否必须	类型	描述
remoteServiceStatus	是	<a href="#">4.6.5.2.6 RemoteServiceStatus</a>	bool isMute; bool isCameraMute; int volume; bool isSpeakerMute; bool isChairman; bool isBase;

参数	是否必须	类型	描述
index	是	int	指示了上述结构体中发生了哪一个事件: index = 1,则指示当前isMute的值, 其他值无需关心(随机值) index = 2,则指示当前isCameraMute的值, 其他值无需关心(随机值) index = 4,则指示当前volume的值, 其他值无需关心(随机值) index = 8,则指示当前isSpeakerMute的值, 其他值无需关心(随机值) index = 16,则指示当前isChairman的值, 其他值无需关心(随机值) 其中isBase当前不会上报, 作为预留, 其值无需关心(随机值)

### 示例代码

```
void ShareEventHandler::OnRemoteServiceStatusChangedNotify(RemoteServiceStatus *remoteServiceStatus,  
int index)  
{  
    if (index == 8) {  
        /* 麦克风静音事件, 刷新麦克风图标 */  
    }  
}
```

### 5.6.4.2.2 OnDevConfStateNotify

#### 回调方法描述

设备推送辅流和会议状态通知。

#### 回调方法定义

```
/* 设备辅流和会议状态通知 */  
void OnDevConfStateNotify(ConfState confState, AuxState auxState) {}
```

#### 参数描述

参数	是否必须	类型	描述
confState	是	<a href="#">4.6.5.1.4 ConfState</a>	会议状态 CONF_IDLE 0 空闲 CONF_BUSY 1 忙
auxState	是	<a href="#">4.6.5.1.5 AuxState</a>	辅流状态 AUX_IDLE 0 空闲 AUX_BUSY 1 忙

### 示例代码

```
void ShareEventHandler::OnDevConfStateNotify(ConfState confState, AuxState auxState)
{
    //更新会议或辅流状态
}
```

## 5.6.4.2.3 OnShareStatusChangedNotify

### 回调方法描述

设备推送共享状态通知。

### 回调方法定义

```
/* 设备推送共享状态通知 */
void OnShareStatusChangedNotify(ShareAppServerStatus status, ShareAppErr reason) {}
```

### 参数描述

参数	是否必须	类型	描述
status	是	<a href="#">4.6.5.1.6 ShareAppServerStatus</a>	设备状态通知枚举值
reason	是	<a href="#">4.6.5.1.7 ShareAppErr</a>	错误码

### 示例代码

```
void ShareEventHandler::OnShareStatusChangedNotify(ShareAppServerStatus status, ShareAppErr reason)
{
    // 刷新UI
}
```

## 5.6.4.3 错误通知

### 5.6.4.3.1 OnError

### 回调方法描述

SDK项上层推送错误通知。

### 回调方法定义

```
/* 错误通知推送 */
void OnError(ShareAppErr errorCode, std::string errorMsg, void* extraInfo) {}
```

### 参数描述

参数	是否必须	类型	描述
errorCode	是	<a href="#">4.6.5.1.7 ShareAppErr</a>	错误枚举值
errorMsg	是	std::string	错误描述

参数	是否必须	类型	描述
extralInfo	是	void*	其他附加信息

### 示例代码

```
void ShareEventHandler::OnError(ShareAppErr errorCode, std::string errorMsg, void* extralInfo)
{
    // 错误上报给UI
}
```

## 5.6.4.3.2 OnDiscoverFailed 参考样例

### 回调方法描述

Discover命令的失败回调，除了在OnError里会上报该事件外，还会在OnDiscoverFailed里上报该事件，同样的，其他的接口失败回调也是在上报OnError的同时还会再上报一个对应事件的失败回调，接口入参和OnDiscoverFailed相同，名称换成对应的接口名称的失败名称，详情请参考IShareServiceController.h。

### 回调方法定义

```
/* 错误通知推送 */
virtual void OnDiscoverFailed(ShareAppErr errorCode, std::string errorMsg) {}
```

### 参数描述

参数	是否必须	类型	描述
errorCode	是	4.6.5.1.7 ShareAppErr	错误枚举值
errorMsg	是	std::string	错误描述

### 示例代码

```
void ShareEventHandler::OnDiscoverFailed(ShareAppErr errorCode, std::string errorMsg)
{
    // 错误上报给UI
}
```

## 5.6.5 附录

### 5.6.5.1 枚举定义

#### 5.6.5.1.1 LogLevel

枚举值	对应值	描述
IDEA_LEVEL_ERROR	0	错误级别

枚举值	对应值	描述
IDEA_LEVEL_WARNING	1	警告级别
IDEA_LEVEL_INFO	2	信息级别
IDEA_LEVEL_DEBUG	3	调试级别

#### 5.6.5.1.2 ShareAppDefinationQuality

枚举值	对应值	描述
FLUENCY_FIRST	0	流畅优先， 默认值
QUAITY_CLEAR_FIRST	1	清晰优先
QUAITY_ADAPTIVE	2	自适应

#### 5.6.5.1.3 ShareAppScreenType

枚举值	对应值	描述
SCREEN_PRIMARY	0	主屏 默认值
SCREEN_SECONDARY	1	扩展屏

#### 5.6.5.1.4 ConfState

枚举值	对应值	描述
CONF_IDLE	0	空闲
CONF_BUSY	1	忙

### 5.6.5.1.5 AuxState

枚举值	对应值	描述
AUX_IDLE	0	空闲
AUX_BUSY	1	忙

### 5.6.5.1.6 ShareAppServerStatus

枚举值	对应值	描述
SERVER_STATUS_CONNECT	0	连接状态
SERVER_STATUS_SHARE	1	投屏共享状态
SERVER_STATUS_STOP_SHARE	2	停止投屏共享状态
SERVER_STATUS_DISCONNECT	3	断开连接状态
SERVER_STATUS_BUTT	4	默认

### 5.6.5.1.7 ShareAppErr

枚举值	对应值	描述
DISCOVER_ERROR	0	设备发现错误
CONNECT_ERROR	1	连接错误
DISCONNECT_ERROR	2	断开连接错误
SHARE_PLAY_ERROR	3	启动投屏错误
SHARE_STOP_ERROR	4	停止投屏错误
GET_REMOTE_SERVICE_STATUS_ERROR	5	遥控器参数查询结果错误
SET_REMOTE_MIC_MUTE_ERROR	6	打开\关闭麦克风结果错误
SET_REMOTE_SPEAKER_VOLUME_ERROR	7	调节扬声器音量结果错误

枚举值	对应值	描述
SET_REMOTE_SPEAKER_MUTE_ERROR	8	打开\关闭扬声器结果错误
SET_REMOTE_CAMERA_MUTE_ERROR	9	打开\关闭摄像机结果错误
LEAVE_CONFERENCE_ERROR	10	离开会议结果错误
END_CONFERENCE_ERROR	11	结束会议结果错误
GET_SDK_LATEST_VERSION_ERROR	12	获取SDK最新版本错误
PROCESS_CONNECTING_ERROR	13	正在处理连接请求
IN_CONNECTION_ERROR	14	客户端已处于连接
PROJECTION_CODE_ERROR	15	投影码错误 数字+字母组合
AUTH_ERROR	16	鉴权失败
CLIENT_IS_LOCKED_ERROR	17	客户端被锁定
CONNECT_BUSY_ERROR	18	终端忙碌
CONNECT_UPDATING_ERROR	19	终端正在升级
WEB_DENY_ERROR	20	WEB被禁用
NO_COMMON_DATA_CAP_ERROR	21	客户端缺少辅流公共能力
GET_TOKEN_FAILD_ERROR	22	获取辅流令牌失败
LOW_BANDWIDTH_ERROR	23	带宽过低，无法发送辅流
CONF_NO_DATA_CAP_ERROR	24	会议无辅流能力
MULTICONNECT_BUSY_ERROR	25	终端正在处理辅流发送，暂且不接受辅流发送该请求
EUA_AUTH_ERROR	26	EUA鉴权失败
EUA_IS_LOCKED_ERROR	27	EUA客户端被锁定

### 5.6.5.1.8 ShareClientTlsVerifyMode

枚举值	对应值	描述
TLS_VERIFY_MODE_NONE	0	不认证
TLS_VERIFY_MODE_SERVER	1	认证服务端
TLS_VERIFY_MODE_CLIENT	2	认证客户端
TLS_VERIFY_MODE_BOTH	3	客户端，服务端双向认证
TLS_VERIFY_MODE_BUTT	4	BUTT

### 5.6.5.1.9 ShareDeviceType

枚举值	对应值	描述
NON_PROJECTOR	0	非投屏器
PROJECTOR	1	投屏器
DEVICE_BUTT	2	BUTT

### 5.6.5.1.10 IdeaShareStatus

枚举值	对应值	描述
IDEA_SHARE_CONNECTED	0	连接
IDEA_SHARE_SHARING	1	共享
IDEA_SHARE_STOPSHARING	2	停止共享
IDEA_SHARE_DISCONNECT	3	断联
IDEA_SHARE_BUTT	4	BUTT

### 5.6.5.1.11 ShareMouseShape

枚举值	对应值	描述
SHARE_MOUSE_SHAPE_HIDE	0	鼠标隐藏
SHARE_MOUSE_SHAPE_WHITE_ARROW	252	白色箭头鼠标
SHARE_MOUSE_SHAPE_LASER_POINTER	255	激光笔， 默认该颜色

### 5.6.5.2 结构体定义

#### 5.6.5.2.1 AppInfo

app对象参数

参数名	类型	描述
appId	std::string	app Id， 最大长度限制128
exePath	std::string	包执行路径， 最大长度限制128

#### 5.6.5.2.2 LogInfo

日志对象参数

参数名	类型	描述
path	std::string	日志路径， 最大长度限制225
logLevel	<a href="#">4.6.5.1.1 LogLevel</a>	日志级别：错误， 警告， 信息， 调试
enable	bool	是否开启日志

#### 5.6.5.2.3 SharePolicy

投放音频策略参数

参数名	类型	描述
shareQuality	<a href="#">4.6.5.1.2 ShareAppDefinationQuality</a>	清晰度

参数名	类型	描述
isShareAudio	bool	0表示不共享音频, 1表示共享音频
isCompatibility	bool	0表示不开启兼容性, 1表示开启兼容性

#### 5.6.5.2.4 DiscoverResult

设备发现结果

参数名	类型	描述
result	int	设备发现结果 0 成功 非0 失败
password	std::string	设备连接码
addressList	std::list<std::string>	设备IP地址列表

#### 5.6.5.2.5 DeviceInfo

远端设备信息

参数名	类型	描述
deviceAddress	std::string	终端地址 ( IP地址 or 域名 )
deviceName	std::string	终端名(会场名)
isSupportRemote	unsigned int	终端侧是否支持遥控器 VOS_TRUE:支持,VOS_FALSE:不支持

#### 5.6.5.2.6 RemoteServiceStatus

远端设备状态

参数名	类型	描述
isMute	unsigned int	麦克风是否关闭 true 关闭 false 打开
isCameraMute	unsigned int	摄像头是否关闭 true 关闭 false 打开
volume	int	扬声器音量值 范围 0 ~21
isSpeakerMute	unsigned int	扬声器是否关闭 true 关闭 false 打开

参数名	类型	描述
isChairman	unsigned int	在会议中是否是主席 true 主席 false 非主席
isBase	unsigned int	是否是标准版 true 标准版本不带摄像头 false 非标准版本带摄像头

### 5.6.5.2.7 ShareTlsInfo

TLS证书设置状态

参数名	类型	描述
caCertpath	std::string	ca根证书路径，最大长度256
clientCertpath	std::string	客户端证书路径，最大长度256
verifyMode	ShareClientTlsVerifyMode	校验模式

### 5.6.5.3 SDK 接口头文件定义

#### 5.6.5.3.1 IEventHandler.h

抽象通知回调

```
//  
// Huawei ideaShare Wapper SDK  
//  
// Copyright (c) 2020 Huawei.io. All rights reserved.  
#ifndef IDEA_HANDLER_H  
#define IDEA_HANDLER_H  
#include "IdeaShareDef.h"  
#ifndef IDEA_SHARE_EXPORT  
#define IDEA_SHARE_EXPORT  
#endif  
#include <string>  
  
using namespace std;  
  
#ifdef __cplusplus  
#if __cplusplus  
extern "C" {  
#endif /* __cplusplus */  
#endif /* __cplusplus */  
  
class IDEA_SHARE_API IEventHandler {  
public:  
    virtual ~IEventHandler() {}  
};  
  
#ifdef __cplusplus  
#if __cplusplus  
}
```

```
#endif /* __cplusplus */  
#endif /* __cplusplus */  
  
#endif
```

### 5.6.5.3.2 IShareServiceController.h

SDK API 和 通知定义文件

```
//  
// Huawei IdeaShare SDK Engine SDK  
//  
// Copyright (c) 2020 Huawei.io. All rights reserved.  
//  
  
#ifndef IDEA_ENGINE_H  
#define IDEA_ENGINE_HS  
  
#ifndef IDEA_SHARE_EXPORT  
#define IDEA_SHARE_EXPORT  
#endif  
  
  
#include "IServiceController.h"  
#include "IEventHandler.h"  
#include <string>  
#include <list>  
#include <atomic>  
#include "IdeaShareDef.h"  
using namespace std;  
  
namespace Huawei {  
    namespace Idea {  
  
        class IDEA_SHARE_API IShareEventHandler : public IEventHandler {  
        public:  
            virtual ~IShareEventHandler() {}  
            /* Discover接口的回调接口 */  
            virtual void OnDiscoverResult(DiscoverResult* discoverResult) {}  
            virtual void OnDiscoverFailed(ShareAppErr errorCode, std::string errorMsg) {}  
  
            /* Connect接口的回调接口 */  
            virtual void OnConnectResult(DeviceInfo* deviceInfo) {}  
            virtual void OnConnectFailed(ShareAppErr errorCode, std::string errorMsg) {}  
  
            /* Disconnect接口的回调接口 */  
            virtual void OnDisconnectResult(int reason) {}  
            virtual void OnDisconnectFailed(ShareAppErr errorCode, std::string errorMsg) {}  
  
            /* StartSharing接口的回调接口 */  
            virtual void OnSharePlayResult() {}  
            virtual void OnSharePlayFailed(ShareAppErr errorCode, std::string errorMsg) {}  
  
            /* StopSharing接口的回调接口 */  
            virtual void OnShareStopResult(int reason) {}  
            virtual void OnShareStopFailed(ShareAppErr errorCode, std::string errorMsg) {}  
  
            /* GetRemoteServiceStatus接口的回调接口 */  
            virtual void OnGetRemoteServiceStatusResult(RemoteServiceStatus* remoteServiceStatus) {}  
            virtual void OnGetRemoteServiceStatusFailed(ShareAppErr errorCode, std::string errorMsg) {}  
  
            /* SetRemoteMicMute接口的回调接口 */  
            virtual void OnSetRemoteMicMuteResult(int result) {}  
            virtual void OnSetRemoteMicMuteFailed(ShareAppErr errorCode, std::string errorMsg) {}  
  
            /* SetRemoteSpeakerVolume接口的回调接口 */  
            virtual void OnSetRemoteSpeakerVolumeResult(int result) {}  
            virtual void OnSetRemoteSpeakerVolumeFailed(ShareAppErr errorCode, std::string errorMsg) {}  
    } // namespace Idea  
} // namespace Huawei
```

```
/* SetRemoteSpeakerMute接口的回调接口 */
virtual void OnSetRemoteSpeakerMuteResult(int result) {}
virtual void OnSetRemoteSpeakerMuteFailed(ShareAppErr errorCode, std::string errorMsg) {}

/* SetRemoteCameraMute接口的回调接口 */
virtual void OnSetRemoteCameraMuteResult(int result) {}
virtual void OnSetRemoteCameraMuteFailed(ShareAppErr errorCode, std::string errorMsg) {}

/* LeaveConference接口的回调接口 */
virtual void OnSetLeaveConferenceResult(int result) {}
virtual void OnSetLeaveConferenceFailed(ShareAppErr errorCode, std::string errorMsg) {}

/* EndConference接口的回调接口 */
virtual void OnSetEndConferenceResult(int result) {}
virtual void OnSetEndConferenceFailed(ShareAppErr errorCode, std::string errorMsg) {}

/* GetSDKLatestVersion接口的回调接口 */
virtual void OnGetSDKLatestVersionResult(std::string version) {}
virtual void OnGetSDKLatestVersionFailed(ShareAppErr errorCode, std::string errorMsg) {}

/* 大屏状态信息变化通知事件 */
virtual void OnRemoteServiceStatusChangedNotify(RemoteServiceStatus *remoteServiceStatus, int index) {}

/* 当前IdeaHub会议状态事件上报接口 */
virtual void OnDevConfStateNotify(ConfSate confState, AuxState auxState) {}

/* 投屏状态发生变化通知上报接口 */
virtual void OnShareStatusChangedNotify(ShareAppServerStatus status, ShareAppErr reason) {}

/* 各种事件失败时统一会上报的接口 */
virtual void OnError(ShareAppErr errorCode, std::string errorMsg, void* extraInfo) {}
};

class IDEA_SHARE_API ShareServiceController : public IServiceController {
public:
    static ShareServiceController* GetInstance();
    ~ShareServiceController();
    /* 日志设置 */
    int SetLog(const LogInfo &logInfo);
    /* TLS设置 */
    int SetTLS(const ShareTlsInfo &tlsInfo);
    /* 初始化 */
    int Create(const AppInfo &appInfo, IEventHandler* eventHandler);
    /* 去初始化 */
    int Destory();
    /* 设置投屏设备类型 */
    int SetDeviceType(ShareDeviceType shareDeviceType);
    /* 设置投屏策略 */
    int SetPolicy(const SharePolicy& sharePolicy);
    /* 投屏码解析ip请求 */
    int Discover(std::string castCode, std::string address, int port);
    /* 连接 */
    int Connect(list<std::string>addressList, int port, std::string password);
    /* 断开连接 */
    int Disconnect();
    /* 设置投屏对象 */
    int SetDisplayScreen(ShareAppScreenType screenChoice);
    /* 开始投屏 */
    int StartSharing();
    /* 停止投屏 */
    int StopSharing();
    /* 获取远程服务状态信息 */
    int GetRemoteServiceStatus();
    /* 打开\关闭麦克风 */
    int SetRemoteMicMute(bool isMute);
    /* 调节扬声器音量 */
    int SetRemoteSpeakerVolume(int volume);
    /* 打开\关闭扬声器 */
}
```

```
int SetRemoteSpeakerMute(bool isSpeakerMute);
/* 打开\关闭摄像机 */
int SetRemoteCameraMute(bool isCameraMute);
/* 离开会议 */
int LeaveConference();
/* 结束会议 */
int EndConference();
/* 获取投屏状态 */
int GetShareStatus();
/* 获取SDK版本 */
std::string GetSDKVersion();
/* 获取最新的服务器版本信息 */
int GetSDKLatestVersion();
/* 全局事件回调 */
IShareEventHandler* mEventHandler;
/* 友元函数 */
friend void ProcessNotifyCallback(uint32_t msgid, uint32_t param1, uint32_t param2, void *data);
/* 全局回调 */
void NotifyCallback(uint32_t msgid, uint32_t param1, uint32_t param2, void *data);
private:
/* 私有构造方法 */
ShareServiceController();
/* 全局事件回调 */
static ShareServiceController *mControllerInstance;
/* 是否正在共享 */
std::atomic<bool> m_IsSharing{ false };
};

} // namespace Idea
} // namespace Huawei

#endif
```

### 5.6.5.3.3 IdeaShareDef.h

#### SDK数据结构及宏定义头文件

```
/*
 * Copyright (c) Huawei Technologies Co., Ltd. 2020-2020. All rights reserved.
 */
/*
 * @file IdeaShareDef.h
 *
 * Copyright(C), 2012-2015, Huawei Tech. Co., Ltd. ALL RIGHTS RESERVED. \n
 *
 * 描述: ideashare wrapper sdk。 \n
 */
#ifndef __IDEAShare_CLIENT_DEF_H__
#define __IDEAShare_CLIENT_DEF_H__

#include<string>
#include<iostream>
#include <list>
#include <algorithm>
using namespace std;

#ifdef WIN32

#ifndef IDEA_SHARE_API
#define IDEA_SHARE_API __declspec(dllexport)
#else
#define IDEA_SHARE_API __declspec(dllimport)
#endif

#ifndef IDEA_SHARE_EXPORT
#define IDEA_SHARE_EXPORT __attribute__ ((visibility__("default")))
#endif
#endif
```

```
#ifdef __cplusplus
#if __cplusplus
extern "C" {
#endif /* __cplusplus */
#endif /* __cplusplus */

/* ****常量定义区**** */
/* ****枚举类型定义区**** */

/**
 * [en]This enumeration is used to describe the log level
 * [cn]日志级别
 */
typedef enum {
    IDEA_LEVEL_ERROR = 0,    /**< [en]Indicates the error level
                                <br>[cn]错误级别 */
    IDEA_LEVEL_WARNING,     /**< [en]Indicates the warning level
                                <br>[cn]警告级别 */
    IDEA_LEVEL_INFO,        /**< [en]Indicates the info level
                                <br>[cn]信息级别 */
    IDEA_LEVEL_DEBUG,       /**< [en]Indicates the debug level
                                <br>[cn]调试级别 */
} LogLevel;

/**
 * [cn]ERROR
 */
typedef enum {
    DISCOVER_ERROR = 0,      //设备发现错误 0
    CONNECT_ERROR,          //连接错误 1
    DISCONNECT_ERROR,        //断开连接错误 2
    SHARE_PLAY_ERROR,        //启动投屏错误 3
    SHARE_STOP_ERROR,        //停止投屏错误 4
    GET_REMOTE_SERVICE_STATUS_ERROR, //遥控器参数查询结果错误 5
    SET_REMOTE_MIC_MUTE_ERROR, //打开\关闭麦克风结果错误 6
    SET_REMOTE_SPEAKER_VOLUME_ERROR, //调节扬声器音量结果错误 7
    SET_REMOTE_SPEAKER_MUTE_ERROR, //打开\关闭扬声器结果错误 8
    SET_REMOTE_CAMERA_MUTE_ERROR, //打开\关闭摄像机结果错误 9
    LEAVE_CONFERENCE_ERROR, //离开会议结果错误 10
    END_CONFERENCE_ERROR, //结束会议结果错误 11
    GET_SDK_LATEST_VERSION_ERROR, //获取SDK最新版本 12
    PROCESS_CONNECTING_ERROR, //正在处理连接请求 13
    IN_CONNECTION_ERROR, //客户端已处于连接 14
    PROJECTION_CODE_ERROR, //投影码错误 数字+字母组合 15
    AUTH_ERROR, //鉴权失败 16
    CLIENT_IS_LOCKED_ERROR, //客户端被锁定 17
    CONNECT_BUSY_ERROR, //终端忙碌 18
    CONNECT_UPDATING_ERROR, //终端正在升级 19
    WEB_DENY_ERROR, //WEB被禁用 20
    NO_COMMON_DATA_CAP_ERROR, //和客户端没有辅流公共能力 21
    GET_TOKEN_FAILD_ERROR, //获取辅流令牌失败 22
    LOW_BANDWIDTH_ERROR, //带宽过低，无法发送辅流 23
    CONF_NO_DATA_CAP_ERROR, //会议无辅流能力 24
    MULTICONNECT_BUSY_ERROR, //终端正在处理辅流发送，暂且不接受辅流发送该请求 25
    EUA_AUTH_ERROR, //EUA鉴权失败 26
    EUA_IS_LOCKED_ERROR, //EUA客户端被锁定 27
} ShareAppErr;

/*服务端指示*/
typedef enum {
    SERVER_STATUS_CONNECT,
    SERVER_STATUS_SHARE,
    SERVER_STATUS_STOP_SHARE,
    SERVER_STATUS_DISCONNECT,
```

```
    SERVER_STATUS_BUTT
} ShareAppServerStatus;

/*服务端指示*/
typedef enum {
    SCREEN_PRIMARY,
    SCREEN_SECONDARY
} ShareAppScreenType;

typedef enum {
    FLUENCY_FIRST = 0, /* 流畅优先 */
    QUAITY_CLEAR_FIRST, /* 清晰优先 */
    QUAITY_ADAPTIVE, /* 自适应 */
} ShareAppDefinationQuality;
/* **** */
/* 结构体定义区
 * **** */
typedef struct {
    std::string appId;// 最大长度128
    std::string exePath;// 最大长度128
} AppInfo;

typedef struct {
    std::string path;//最大长度256
    LogLevel logLevel;
    bool enable; //是否开启日志
} LogInfo;

typedef struct {
    int result;
    std::string password;
    std::list<std::string> addressList;
} DiscoverResult;

typedef struct {
    unsigned int isMute;
    unsigned int isCameraMute;
    int volume;
    unsigned int isSpeakerMute;
    unsigned int isChairman;
    unsigned int isBase;
} RemoteServiceStatus;

typedef enum {
    IDEA_SHARE_CONNECTED, /* <连接 */
    IDEA_SHARE_SHARING, /* <共享 */
    IDEA_SHARE_STOPSHARING, /* <停止共享 */
    IDEA_SHARE_DISCONNECT, /* <断连 */
    IDEA_SHARE_BUTT
} IdeaShareStatus;

typedef enum {
    CONF_IDLE = 0,           // 空闲
    CONF_BUSY,               // 忙
} ConfState;

typedef enum {
    AUX_IDLE = 0,           // 空闲
    AUX_BUSY,               // 忙
} AuxState;

/*
 * 连接上报结果结构体
 */
typedef struct {
    std::string deviceAddress; /* <终端地址 (IP地址 or 域名) */
    std::string deviceName; /* <终端名(会场名) */
    unsigned int isSupportRemote; /* <终端侧是否支持遥控器 VOS_TRUE:支持,VOS_FALSE:不支持。TUP目前默认为VOS_TRUE, 支持 */
}
```

```
    } DeviceInfo;

    typedef struct {
        ShareAppDefinationQuality shareQuality; //清晰度
        bool isShareAudio; /* 0表示不共享音频, 1表示共享音频 */
        bool isCompatibility; /* 0表示不开启兼容性, 1表示开启兼容性 */
    } SharePolicy;

    typedef enum {
        NON_PROJECTOR = 0,
        PROJECTOR,
        DEVICE_BUTT
    } ShareDeviceType;

/* 认证模式*/
typedef enum {
    TLS_VERIFY_MODE_NONE = 0, /* * 不认证 */
    TLS_VERIFY_MODE_SERVER, /* * 认证服务端 */
    TLS_VERIFY_MODE_CLIENT, /* * 认证客户端 */
    TLS_VERIFY_MODE_BOTH, /* * 客户端, 服务器双向认证 */
    TLS_VERIFY_MODE_BUTT
} ShareClientTlsVerifyMode;

typedef struct {
    std::string caCertpath; /* < ca根证书,最大长度256 */
    std::string clientCertpath; /* < 客户端证书,最大长度256 */
    ShareClientTlsVerifyMode verifyMode; /* * < 校验模式 */
} ShareTlsInfo;

#ifndef __cplusplus
#if __cplusplus
}
#endif /* __cplusplus */
#endif /* __cplusplus */

#endif
```

#### 5.6.5.3.4 IServiceController.h

service虚类定义

```
///
// Huawei ideaShare Wapper SDK
//
// Copyright (c) 2020 Huawei.io. All rights reserved.

#ifndef IDEA_CONTROLLER_H
#define IDEA_CONTROLLER_H
#ifndef IDEA_SHARE_EXPORT
#define IDEA_SHARE_EXPORT
#endif
#endif

#include "IEventHandler.h"
#include "IdeaShareDef.h"
#include <string>
using namespace std;

class IDEA_SHARE_API IServiceController {
public:
    virtual int Create(const AppInfo &appInfo, IEventHandler* eventHandler) = 0;
    virtual int Destory() = 0;
    virtual ~IServiceController() {}
private:
    //预留接口, 适配EMUI
    virtual std::string GetKitVersion() { return "v1.0.0"; }
    virtual bool HasFeature(std::string feature) { return true; }
};

#endif
```

### 5.6.5.4 二进制公网地址说明

二进制文件中包含如下公网URL，为zlib开源组件自带链接，程序不会访问该链接。

<http://www.winimage.com/zLibDll>

## 5.7 Windows 接口参考（JavaScript）

### 5.7.1 概述

本文主要介绍投屏服务JavaScript接口参考。主要包含文档修改记录，接口函数的请求样例、响应样例以及事件上报说明、参数说明、附录等内容。

### 5.7.2 变更记录

表1-1变更记录

日期	版本	变更内容
2021-06-03	1.0.0	初稿完成

### 5.7.3 前言

IdeaShareSDK提供四个OS的SDK，其中windows开放的SDK接口是C++接口，为方便部分使用JavaScript语言（以下简称JS）集成SDK的客户直接调用C++接口，现特开发出与C++接口对应的JS形式的接口，方便用户进行调用，此文档用于描述和规定每个接口的JS格式的请求样例以及相应的响应样例

### 5.7.4 使用方法

Windows JS SDK中提供监听端口的IdeaShareAdapter.exe服务端程序，使用TCP(HTTP)进行通讯，启动后就会后台自动运行并持续监听8099端口，用户只需在客户端向该端口发送JSON格式的接口信息并监听回复消息即可完成接口的调用，其中TCP的前2个字节为消息length，后面跟消息内容。

### 5.7.5 接口参考

#### 5.7.5.1 基础配置

##### 5.7.5.1.1 初始化日志

请求样例：

```
{  
  "messId": "SetLog",  
  "params": ["D://temp", "DEBUG","1"]  
}
```

消息字段	取值	备注
messId	SetLog	严格按照取值填写
params	日志路径, 日志级别, 是否开启日志功能	数组 参数1: 日志设置路径 参数2: 日志设置级别 ( ERROR、WARNING、INFO、DEBUG从上到下 日志级别依次降低, 高级别向下包含低级别等级的日志 ) 参数3: 是否开启日志功能, 0: 不开启日志功能, 非0值: 开启日志功能 参数2不区分大小, 参数1路径长度不超过226

响应样例:

```
{  
  "messId": "SetLog",  
  "response": ["0"]  
}
```

消息字段	取值	备注
messId	SetLog	严格按照取值填写
response	0: 成功; 1: 失败	同步调用返回值, 数组

设置日志的C++原型接口为: int SetLog(const LogInfo &logInfo);

其中LogInfo结构体为相应的参数, 结构体内容如附录[1.5.8.2.1](#) 所示。

### 5.7.5.1.2 初始化 SDK

请求样例:

```
{  
  "messId": "Create",  
  "params": ["IdeaShare", "SDK"]  
}
```

消息字段	取值	备注
messId	Create	严格按照取值填写
params	参数名称固定IdeaShare, SDK	数组

响应样例：

```
{  
  "messId": "Create",  
  "response": ["0"]  
}
```

消息字段	取值	备注
messId	Create	严格按照取值填写
response	0: 成功; 1: 失败	数组

初始化的C++原型接口为：int Create(const AppInfo &appInfo, IEventHandler\* eventHandler);

其中AppInfo结构体为相应的参数，当前固定填充为"IdeaShare","SDK"，  
IEventHandler为注册的回调，由于JS格式采用TCP(HTTP)通讯方式，因此无需填充注册函数，在服务端会代为注册并将回调函数的信息返回给客户端。

### 5.7.5.1.3 去初始化 SDK

请求样例：

```
{  
  "messId": "Destroy",  
  "params": []  
}
```

消息字段	取值	备注
messId	Destroy	严格按照取值填写
params		没有参数

响应样例：

```
{  
  "messId": "Destroy",  
  "params": []  
}
```

```
"response": ["0"]  
}
```

消息字段	取值	备注
messId	Destroy	严格按照取值填写
response	0: 成功; 1: 失败。	数组

去初始化接口的C++原型接口为：int Destory();

下发了该命令后服务端会进行去初始化操作，销毁初始化的数据，如果只是建立连接但是未投屏，则会有接口响应和包括Disconnect接口的回调，如果是在投屏过程中下发该接口，则由于socket已经被清楚，因此不会收到Destroy接口的接口响应，但是依然会有Disconnect的回调。

### 5.7.5.2 连接设备

#### 5.7.5.2.1 设备发现

请求样例：

```
{  
  "messId": "Discover",  
  "params": ["22583967", "", "0"]  
}
```

消息字段	取值	备注
messId	Discover	严格按照取值填写
params	投影码, 在线模式IP, 在线模式端口	数组 参数1代表投影码 1) 离线模式下, 只需要投影码, 后两个参数依次为空和0 2) 在线模式下, 需要投影码, 同时需要EUA的IP和端口, 分别为第二, 第三参数

响应样例：

```
{  
  "messId": "Discover",  
  "response": ["0", "0", "10.159.154.88,169.159.154.88", "001"]
```

```
}
```

消息字段	取值	备注
messId	Discover	严格按照取值填写
response	0、接口调用结果、Discover解析结果：IP列表和鉴权码	数组 参数1：固定值 参数2：接口调用结果 0成功 1失败 参数3：解析的IP列表， 参数4：鉴权码。 接口调用失败通过onError返回具体详情

设备发现的C++原型接口为：int Discover(*std::string* castCode, *std::string* address, int port);

其中如果当前IdeaHub大屏设置为离线模式，则castCode为投影码，address为空，port为0；如果当前IdeaHub大屏设置为在线模式，则castCode为投影码，同时需要将当前大屏的IP填写到address参数处，以及大屏的当前端口填写到port处，三者缺一不可。调用该接口成功后服务端会收到底层的回调消息，回调内容为投影码解析的结果，其中包括解析出的IP列表，以及鉴权码，这两者是连接命令的必须参数。

### 5.7.5.2.2 TLS 证书校验设置

请求样例：

```
{
  "messId": "SetTLS",
  "params": ["D://temp", "", "1"]
}
```

消息字段	取值	备注
messId	SetTLS	严格按照取值填写
params	Ca根证书路径，客户端证书路径，校验模式	数组 参数1：根证书路径 参数2：客户端证书路径（参数1参数2只填一个，且路径长度不超过256） 参数3：校验模式，与参数1，参数相关 参数2参数3结构体格式详见附录 <a href="#">1.5.8.2.2 1.5.8.2.5</a>

响应样例：

```
{  
  "messId": "SetTLS",  
  "response": ["0"]  
}
```

消息字段	取值	备注
messId	SetTLS	严格按照取值填写
response	0: 成功; 1: 失败	数组

TLS证书校验设置的C++原型接口为：int SetTLS(const ShareTlsInfo &tlsInfo);

其中ShareTlsInfo结构体为相应的参数，结构体内容如附录[1.5.8.2.2](#) 所示。

### 5.7.5.2.3 连接

请求样例：

```
{  
  "messId": "Connect",  
  "params": ["10.159.154.88,169.159.154.88", "001"]  
}
```

消息字段	取值	备注
messId	Connect	严格按照取值填写
params	IP列表, 鉴权码	数组 参数1: IP列表 参数2: 鉴权码 此处的IP列表、以及鉴权码均来自Discover解析出来的结果

响应样例：

```
{  
  "messId": "Connect",  
  "response": ["0","10.159.157.85", "IdeaHub", "1"]  
}
```

消息字段	取值	备注
messId	Connect	严格按照取值填写
response	固定值、终端IP地址，终端名称，终端侧是否支持遥控器	数组 参数1：固定值 参数2：终端IP地址 参数3：终端名称 参数4：终端侧是否支持遥控器(0:不支持，1：支持)

连接的C++原型接口为：int Connect(*list<std::string>* addressList, int port, *std::string* password);

其中addressList是下发投影码解析命令后得到的回调消息中的IP地址列表，password为鉴权码，port口在服务端会代为填写，因此请求样例中参数只需要填IP列表和鉴权码；调用该接口成功后底层会收到相应的回调消息，回调消息内容为最终连接上的终端地址，终端当前的名称以及终端侧是否支持遥控器，这三个参数可用于更新上层UI的界面。

#### 5.7.5.2.4 断开连接

请求样例：

```
{  
  "messId": "Disconnect",  
  "params": []  
}
```

消息字段	取值	备注
messId	Disconnect	严格按照取值填写
params		没有参数

响应样例：

```
{  
  "messId": "Disconnect",  
  "response": ["0"]  
}
```

消息字段	取值	备注

messId	Disconnect	严格按照取值填写
response	0: 成功; 1: 失败。	数组

断开连接的C++原型接口为：int StopSharing();

下发该命令后如果当前正在投屏过程中，则会停止投屏并断开和大屏的连接，如果当前已经停止投屏，则会直接断开和大屏的连接，如果想要再次投屏，则需要依次重新下发投影码解析->连接->开始投屏命令

### 5.7.5.3 投屏

#### 5.7.5.3.1 开始投屏

请求样例：

```
{  
    "messId": "StartSharing",  
    "params": []  
}
```

消息字段	取值	备注
messId	StartSharing	严格按照取值填写
params		没有参数

响应样例：

```
{  
    "messId": "StartSharing",  
    "response": ["0"]  
}
```

消息字段	取值	备注
messId	StartSharing	严格按照取值填写
response	0: 成功; 1: 失败。	数组

开始投屏接口的C++原型接口为：int StartSharing();

在解析投影码，连接后就可以下发该命令进行投屏了，下发了该命令后底层会根据之前和大屏侧建立的连接进行投屏。

### 5.7.5.3.2 停止投屏

请求样例：

```
{  
  "messId": "StopSharing",  
  "params": []  
}
```

消息字段	取值	备注
messId	StopSharing	严格按照取值填写
params		没有参数

响应样例：

```
{  
  "messId": "StopSharing",  
  "response": ["0"]  
}
```

消息字段	取值	备注
messId	StopSharing	严格按照取值填写
response	0：成功；1：失败。	数组

停止投屏接口的C++原型接口为：int StopSharing();

无需传入参数，下发该命令后就会停止投屏，但是此使并没有和大屏端口连接，如果要重新投屏，则下发上述的开始投屏命令即可。

### 5.7.5.3.3 设置投屏对象

请求样例：

```
{  
  "messId": "SetDisplayScreen",  
  "params": ["SCREEN_PRIMARY"]  
}
```

消息字段	取值	备注
messId	SetPolicy	严格按照取值填写

params	SCREEN_PRIMARY或 SCREEN_SECONDARY	SCREEN_PRIMARY主屏； SCREEN_SECONDARY副屏
--------	-------------------------------------	---

响应样例：

```
{  
  "messId": "SetPolicy",  
  "response": ["0"]  
}
```

消息字段	取值	备注
messId	SetPolicy	严格按照取值填写
response	0: 成功； 1: 失败。	数组

设置投屏对象的C++原型接口为：int SetDisplayScreen(ShareAppScreenType screenChoice);

该接口主要用于设置投那块屏幕，当前只支持双屏中选择一个屏进行投屏，其中 ShareAppScreenType结构体为相应的参数信息，结构体内容如附录[1.5.8.2.6](#) 所示

#### 5.7.5.3.4 设置投屏策略

请求样例：

```
{  
  "messId": "SetPolicy",  
  "params": ["2", "1", "1"]  
}
```

消息字段	取值	备注
messId	SetPolicy	严格按照取值填写
params	投屏策略(0,1,2)； 音频共享(0,1)； 是否开启兼容性(0,1)	数组 参数1： 0 流畅优先 1 清晰优先 2 自适应； 参数2： 0 不共享音频 1 共享音频 参数3： 0 不开启兼容性 1 开启兼容性

响应样例：

```
{  
  "messId": "SetPolicy",  
  "response": ["0"]  
}
```

消息字段	取值	备注
messId	SetPolicy	严格按照取值填写
response	0: 成功; 1: 失败。	数组

设置投屏策略接口的C++原型接口为: int SetPolicy(const SharePolicy& sharePolicy);

该接口主要是在投屏前或者投屏中设置当前的投屏策略，包括投屏的流畅度以及是否共享音频，其中SharePolicy结构体为相应的参数，结构体内容如附录[1.5.8.2.3](#) 所示。

#### 5.7.5.3.5 获取投屏状态

请求样例:

```
{  
  "messId": "GetShareStatus",  
  "params": []  
}
```

消息字段	取值	备注
messId	GetShareStatus	严格按照取值填写
params		没有参数

响应样例:

```
{  
  "messId": "GetShareStatus",  
  "response": ["0","1"]  
}
```

消息字段	取值	备注
messId	GetShareStatus	严格按照取值填写

response	接口调用结果，当前的共享状态	数组 参数1：接口调用结果， 0：成功，1：失败 参数2：当前的共享状态，具体的值所对应的状 态参考结构体 <a href="#">1.5.8.2.7</a>
----------	----------------	---

获取投屏状态接口的C++原型接口为：int GetShareStatus();

该接口用于获取当前的投屏状态。

#### 5.7.5.4 遥控器

遥控器功能是指可以通过客服端下发指令设置/查询IdeaHub大屏侧得到设备的状态，比如设置/获取大屏侧的音量值，麦克风的状态并且打开/关闭麦克风等等，该功能可以配合UI界面使用

#### 5.7.5.5 查询远端设备状态

请求样例：

```
{  
    "messId": "GetRemoteServiceStatus",  
    "params": []  
}
```

消息字段	取值	备注
messId	GetRemoteServiceStatus	严格按照取值填写
params		没有参数

响应样例：

```
{  
    "messId": "GetRemoteServiceStatus",  
    "response": ["0","1","0","12","0","1","0"]  
}
```

消息字段	取值	备注
messId	GetRemoteServiceStatus	严格按照取值填写

response	接口调用结果，麦克风是否关闭，摄像头是否关闭，扬声器音量值，扬声器是否静音，在会议中是否是主席，是否是标准版	数组 参数1：接口调用结果 0 成功 1失败 参数2：麦克风是否关闭 1 关闭，0 打开 参数3：摄像头是否关闭 1 关闭，0 打开 参数4：扬声器音量值 参数5：扬声器是否静音 1 静音，0 打开 参数6：在会议中是否是主席 1是，0 不是 参数7：是否是标准版 1标准版不带摄像头，0 非标准版带摄像头
----------	--	--

查询远端设备状态的C++原型接口为：int GetRemoteServiceStatus();

该接口主要用于查询大屏侧的基本信息，包括大屏当前的音量，摄像头开关状态，扬声器是否静音，当前会议者是否是主席，大屏版本是否支持摄像头等。

### 5.7.5.6 打开/关闭麦克风

请求样例：

```
{  
  "messId": "SetRemoteMicMute",  
  "params": ["1"]  
}
```

消息字段	取值	备注
messId	SetRemoteMicMute	严格按照取值填写
params	0/1	0 设置不静音，1 设置为静音

响应样例：

```
{  
  "messId": "SetRemoteMicMute",  
  "response": ["0","1"]  
}
```

消息字段	取值	备注
messId	SetRemoteMicMute	严格按照取值填写
response	接口调用结果，设置成功与否	数组 参数1：接口调用结果 0 成功 1 失败 参数2：设置是否成功 0 失败 1 成功

打开/关闭麦克风接口的C++原型接口为：int SetRemoteMicMute(bool isMute);

该接口用于远程控制IdeaHub大屏侧的麦克风的开/关，可以配合UI界面使用。

#### 5.7.5.7 调节扬声器音量

请求样例：

```
{  
    "messId": "SetRemoteSpeakerVolume",  
    "params": ["13"]  
}
```

消息字段	取值	备注
messId	SetRemoteSpeakerVolume	严格按照取值填写
params	音量值	将当前IdeaHub侧的扬声器音量值设置为参数值，范围为0~21

响应样例：

```
{  
    "messId": "SetRemoteSpeakerVolume",  
    "response": ["0","1"]  
}
```

消息字段	取值	备注
messId	SetRemoteSpeakerVolume	严格按照取值填写

response	接口调用结果，设置成功与否	数组 参数1：接口调用结果 0 成功 1 失败 参数2：设置是否成功 0 失败 1 成功
----------	---------------	--

打开/关闭麦克风接口的C++原型接口为：int SetRemoteSpeakerVolume(int volume);  
该接口用于远程控制IdeaHub大屏侧的扬声器的音量值，可以配合UI界面使用。

### 5.7.5.8 打开/关闭扬声器

请求样例：

```
{  
    "messId": "SetRemoteSpeakerMute",  
    "params": []  
}
```

消息字段	取值	备注
messId	SetRemoteSpeakerMute	严格按照取值填写
params	0/1	0打开，1关闭

响应样例：

```
{  
    "messId": "SetRemoteSpeakerMute",  
    "response": ["0","1"]  
}
```

消息字段	取值	备注
messId	SetRemoteSpeakerMute	严格按照取值填写
response	接口调用结果，设置成功与否	数组 参数1：接口调用结果 0 成功 1 失败 参数2：设置是否成功 0 失败 1 成功

打开/关闭扬声器接口的C++原型接口为：int SetRemoteSpeakerMute(bool isSpeakerMute);

该接口用于远程控制IdeaHub大屏侧的扬声器的开/关，可以配合UI界面使用。

### 5.7.5.9 打开/关闭摄像头

请求样例：

```
{  
  "messId": "SetRemoteCameraMute",  
  "params": []  
}
```

消息字段	取值	备注
messId	SetRemoteCameraMute	严格按照取值填写
params	0/1	0打开，1关闭

响应样例：

```
{  
  "messId": " SetRemoteCameraMute",  
  "response": ["0"]  
}
```

消息字段	取值	备注
messId	SetRemoteCameraMute	严格按照取值填写
response	接口调用结果，设置成功与否	数组 参数1：接口调用结果 0 成功 1 失败 参数2：设置是否成功 0 失败 1 成功

打开/关闭摄像头接口的C++原型接口为：int SetRemoteCameraMute(bool isCameraMute);

该接口用于远程控制IdeaHub大屏侧的摄像头开/关，可以配合UI界面使用。

### 5.7.5.10 离开会议

请求样例：

```
{  
  "messId": "LeaveConference",  
  "params": []  
}
```

}

消息字段	取值	备注
messId	LeaveConference	严格按照取值填写
params		没有参数

响应样例：

```
{  
  "messId": "LeaveConference",  
  "response": ["0"]  
}
```

消息字段	取值	备注
messId	LeaveConference	严格按照取值填写
response	接口调用结果，设置成功与否	数组 参数1：接口调用结果 0 成功 1 失败 参数2：设置是否成功 0 失败 1 成功

离开会议接口的C++原型接口为：int LeaveConference();

该接口用于在会议中投屏时设置离开会议，可以配合UI界面使用。

### 5.7.5.11 结束会议

请求样例：

```
{  
  "messId": "EndConference",  
  "params": []  
}
```

消息字段	取值	备注
messId	EndConference	严格按照取值填写
params		没有参数

响应样例：

```
{  
  "messId": "EndConference",  
  "response": ["0"]  
}
```

消息字段	取值	备注
messId	EndConference	严格按照取值填写
response	接口调用结果，设置成功与否	数组 参数1：接口调用结果 0 成功 1 失败 参数2：设置是否成功 0 失败 1 成功

结束会议的C++原型接口为：int EndConference();

该接口用于在会议中投屏时结束会议，可以配合UI界面使用。

### 5.7.5.12 版本信息查询

#### 5.7.5.12.1 获取 SDK 版本信息

请求样例：

```
{  
  "messId": "GetSDKVersion",  
  "params": []  
}
```

消息字段	取值	备注
messId	GetSDKVersion	严格按照取值填写
params		没有参数

响应样例：

```
{  
  "messId": "GetSDKVersion",  
  "response": ["0", "1.0.0.001"]  
}
```

消息字段	取值	备注

messId	GetSDKVersion	严格按照取值填写
response	接口调用结果 sdk版本号	数组 参数1：接口调用结果 0 成功 1 失败 参数2：sdk版本号

获取SDK版本接口的C++原型接口为：`std::string GetSDKVersion();`

该接口用于获取当前的SDK接口的版本信息，可以用于配合UI界面使用。

#### 5.7.5.12.2 获取大屏侧服务端版本信息

请求样例：

```
{  
    "messId": "GetSDKLatestVersion",  
    "params": []  
}
```

消息字段	取值	备注
messId	GetSDKLatestVersion	严格按照取值填写
params		没有参数

响应样例：

```
{  
    "messId": "GetSDKLatestVersion",  
    "response": ["0","1.0.0.003"]  
}
```

消息字段	取值	备注
messId	GetSDKLatestVersion	严格按照取值填写
response	接口调用结果 大屏侧服务端版本号	数组 参数1：接口调用结果 0 成功 1 失败 参数2：大屏侧服务端版 本号

获取最新的服务器版本信息的C++原型接口为：`int GetSDKLatestVersion();`

该接口用于获取IdeaHub大屏侧服务端的版本信息，可以配合UI界面使用。

### 5.7.5.13 事件上报/接口回调结果

上述接口皆为主动下发接口，但是大屏侧的状态发生改变时，也会主动通知上报给客户端，以便客户端及时刷新UI，比如当大屏侧的扬声器音量发生了改变，会上报给应用层，然后通知客户端发生了该事件，客户端可以据此刷新UI侧所显示的大屏音量的音量值；所有事件上报/接口回调结果皆为json格式。

#### 5.7.5.13.1 投屏状态发生变化事件上报

上报样例：

```
{  
    "messId": "OnShareStatusChangedNotify",  
    "response": ["0","1","0"]  
}
```

消息字段	取值	备注
messId	OnShareStatusChangedNotify	严格按照取值填写
response	0, status, reason	数组 参数1：固定值 参数2：当前投屏状态 参数3：原因

### 5.7.5.14 大屏状态信息变化通知事件

上报样例：

```
{  
    "messId": "OnRemoteServiceStatusChangedNotify",  
    "response": ["0","0","0","12","0","0","0","4"]  
}
```

消息字段	取值	备注
messId	OnRemoteServiceStatusChangedNotify	严格按照取值填写

response	0, isMute, isCameraMute, volume, isSpeakerMute, isChairman,index,	数组 参数1：固定值 参数2：麦克风开/关（1/0） 参数3：摄像头开关（1/0） 参数4：扬声器音量值（范围：0~21） 参数5：扬声器开/关（1/0） 参数6：主席状态 参数7：是否是标准版本 参数8：index,指示前5个参数中哪个参数的值发生了改变；“1”代表参数2改变；“2”代表参数2改变；“4”代表参数3改变；“8”代表参数4改变；“16”代表参数5改变； 注意：一次只会有一个值改变，同一个值多次改变（如改变音量）会上报多次事件
----------	---	---

### 5.7.5.15 设备推送辅流和会议状态通知

上报样例：

```
{  
    "messId": "OnConfStateChangedNotify",  
    "response": ["0","0","1"]  
}
```

消息字段	取值	备注
messId	OnConfStateChangedNotify	严格按照取值填写
response	0, confState, auxState	数组 参数1：固定值 参数2：会议状态0空闲，1忙 参数3：辅流状态0 空闲，1忙

### 5.7.5.16 接口调用错误回调

当异步接口回调失败的时，会调用错误失败接口，会调用统一的错误接口（onError），传入相应的错误码，以便知道是哪个接口失败，同时也会调用对应接口的失败函数，传入错误码和对应的调用接口信息，使用者只需要处理其中的一个响应就行。

#### 5.7.5.16.1 OnError 错误事件上报

上报样例：

```
{  
    "messId": "OnError",  
    "response": ["0", "1", "12", "OnGetSDKLatestVersionFailed"]  
}
```

消息字段	取值	备注
messId	OnError	严格按照取值填写
response	0、错误标志、错误码、对应的接口函数名称	数组 参数1：固定值 参数2：错误标志 参数3：错误码 参数4：对应的接口函数名称 错误码和对应的调用接口名称的对应关系详见附录

## 5.7.6 附录

### 5.7.6.1 错误码一对应接口表

编号	错误码名称	说明	对应调用接口名称
0	DISCOVER_ERROR	设备发现错误	Discover
1	CONNECT_ERROR	连接错误	Connect
2	DISCONNECT_ERROR	断开连接错误	Disconnect
3	SHARE_PLAY_ERROR	启动投屏错误	StartSharing
4	SHARE_STOP_ERROR	停止投屏错误	StopSharing
5	GET_REMOTE_SERVICE_STATUS_ERROR	遥控器参数查询结果错误	GetRemoteServiceStatus

6	SET_REMOTE_MIC_MUTE_ERROR	打开/关闭麦克风错误	SetRemoteMicMute
7	SET_REMOTE_SPEAKER_VOLUME_ERROR	调节扬声器音量结果错误	SetRemoteSpeakerVolume
8	SET_REMOTE_SPEAKER_MUTE_ERROR	打开/关闭扬声器结果错误	SetRemoteSpeakerMute
9	SET_REMOTE_CAMERA_MUTE_ERROR	打开/关闭摄像头结果错误	SetRemoteCameraMute
10	LEAVE_CONFERENCE_ERROR	离开会议结果错误	LeaveConference
11	END_CONFERENCE_ERROR	结束会议结果错误	EndConference
12	GET_SDK_LATEST_VERSION_ERROR	获取最新服务端版本信息结果错误	GetSDKLatestVersion
13	PROCESS_CONNECTING_ERROR	正在处理连接请求	Connect
14	IN_CONNECTION_ERROR	客户端已处于连接	Connect
15	PROJECTION_CODE_ERROR	投影码错误 数字+字母组合	Discover
16	AUTH_ERROR	鉴权失败	Connect
17	CLIENT_IS_LOCKED_ERROR	客户端被锁定	Connect
18	CONNECT_BUSY_ERROR	终端忙碌	Connect
19	CONNECT_UPDATING_ERROR	终端正在升级	Connect
20	WEB_DENY_ERROR	WEB被禁用	Connect
21	NO_COMMON_DATA_CAP_ERROR	和客户端没有辅流公共能力	StartSharing
22	GET_TOKEN_FAILD_ERROR	获取辅流令牌失败	StartSharing
23	LOW_BANDWIDTH_ERROR	带宽过低，无法发送辅流	StartSharing
24	CONF_NO_DATA_CAP_ERROR	会议无辅流能力	StartSharing
25	MULTICONNECT_BUSY_ERROR	终端正在处理辅流发送，暂且不接受辅流发送该请求	StartSharing
26	EUA_AUTH_ERROR	EUA鉴权失败	Discover

27	EUA_IS_LOCKED_ERROR	EUA客户端被锁定	Discover
----	---------------------	-----------	----------

## 5.7.6.2 结构体描述

### 5.7.6.2.1 LogInfo 结构体

参数名	类型	描述
path	std::string	日志路径，最大长度限制225
logLevel	LogLevel枚举类型	日志级别：ERROR、WARN、INFO、DEBUG
enable	bool	是否开启日志记录功能

### 5.7.6.2.2 ShareTlsInfo 结构体

参数名	类型	描述
caCertpath	std::string	Ca根证书路径，最大长度256
clientCertpath	std::string	客户端证书路径，最大长度256
verifyMode	ShareClientTlsVerifyMode枚举类型	校验模式

### 5.7.6.2.3 SharePolicy 结构体

参数名	类型	描述
shareQuality	ShareAppDefinationQuality 枚举变量	投屏的清晰度
isShareAudio	bool	0表示不开启音频共享，1表示开启音频共享。

### 5.7.6.2.4 LogLevel 结构体

枚举值	对应值	描述
IDEA_LEVEL_ERROR	0	错误级别
IDEA_LEVEL_WARNING	1	警告级别
IDEA_LEVEL_INFO	2	信息级别
IDEA_LEVEL_DEBUG	3	调试级别

### 5.7.6.2.5 ShareClientTlsVerifyMode 结构体

枚举值	对应值	描述
TLS_VERIFY_MODE_NONE	0	不认证
TLS_VERIFY_MODE_SERVER	1	认证服务端
TLS_VERIFY_MODE_CLIENT	2	认证客户端
TLS_VERIFY_MODE_BOTH	3	客户端，服务端双向认证
TLS_VERIFY_MODE_BUTT	4	BUTT

### 5.7.6.2.6 ShareAppScreenType 结构体

枚举值	对应值	描述
SCREEN_PRIMARY	0	主屏默认
SCREEN_SECONDARY	1	扩展屏

### 5.7.6.2.7 IdeaShareStatus 结构体

枚举值	对应值	描述
IDEA_SHARE_CONNECTED	0	连接状态
IDEA_SHARE_SHARING	1	投屏状态
IDEA_SHARE_STOPSHARING	2	停止投屏状态
IDEA_SHARE_DISCONNECT	3	断连状态
IDEA_SHARE_BUTT	4	BUTT值

## 5.8 错误码参考

## 5.8.1 错误码

参照[4.6.5.1.7](#) 错误码定义。

# 6 个人数据处理说明

个人数据清单	使用目的	存留期
用户屏幕图像	投屏中屏幕镜像信息投放	IdeaShare不保存个人数据
用户音频输出口	投屏中音频信息投放	
用户IP地址	投屏连接	

- 
- 
-