直播 SDK

客户端 API

腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

客户端 API
Android
API 概述

最近更新时间：2026-04-20 17:34:06

AtomicXCore API 概览 (Kotlin)

AtomicXCore SDK 是最新推出的面向视频直播、语聊房等场景的全新一代基于响应式的 API。您可以非常快速的基于这组 API 构建自己的 UI 页面。它支持房间管理、屏幕分享、成员管理、麦位控制、基础美颜等丰富功能。本页面包含 AtomicXCore SDK 的所有的功能模块，并分类展示如下。

用户登录

类名	说明
LoginStore	登录相关接口，管理用户登录、登出、用户信息设置等操作。

设备管理

类名	说明
AudioEffectStore	音效设置相关接口，管理主播的变声、混响和耳返等音效功能。
BaseBeautyStore	基础美颜相关接口，管理磨皮、美白、红润等基础美颜效果的调节和状态同步。
DeviceStore	设备相关接口，操作麦克风、摄像头等。

直播互动

类名	说明
BattleStore	直播 PK 管理相关接口，管理 PK 的创建、加入、离开等操作。
CoGuestStore	直播连麦管理相关接口，管理主播与观众之间的连麦申请、邀请、接受、拒绝等操作。
CoHostStore	直播主播连线管理相关接口，管理主播间相互连线的创建、加入、离开等操作。
LikeStore	点赞相关接口，管理直播间/语音聊天房内的点赞发送、点赞状态同步及点赞事件监听等操作。

LiveAudienceStore	直播观众相关接口，管理观众列表、权限设置等操作。
LiveListStore	直播列表相关接口，管理直播房间的创建、加入、离开等操作。
LiveSeatStore	直播麦位管理相关接口，管理麦位的上下麦、锁麦、释放麦位等操作。

弹幕消息

类名	说明
BarrageStore	弹幕相关接口，管理直播间/语音聊天房内的弹幕发送、弹幕状态同步及弹幕事件监听等操作。

礼物消息

类名	说明
GiftStore	礼物相关接口，管理直播间/语音聊天房内的礼物发送、礼物状态同步及礼物事件监听等操作。

视图组件

类名	说明
CameraView	调用开始摄像头测试时，会将摄像头的渲染画面显示到此 CameraView 上。
LiveCoreView	直播核心视图组件，提供直播推流和播放的视图容器，支持多人连麦、PK 等功能。

Define

最近更新时间：2026-04-20 17:34:06

简介

此模块定义了跨平台通用的基础类型，为 API 调用提供统一的错误处理和回调机制。

功能特性

- **回调机制**：提供统一的完成回调闭包类型。

数据结构

CompletionHandler

提供基础类型定义，包括错误处理结构、回调闭包类型和响应式状态管理工具。
方法
onSuccess: 操作成功时调用。

```
fun onSuccess()
```

onFailure: 操作失败时调用。

```
fun onFailure(code: Int, desc: String)
```

参数名	类型	说明
code	Int	错误码。
desc	String	错误描述。

ListResultCompletionHandler

列表结果完成回调接口。
用于返回分页列表的异步操作结果回调接口。
方法
onSuccess: 操作成功时调用，返回列表结果和游标。

```
fun onSuccess(
    result: List<T>,
    cursor: String
```

```
)
```

参数名	类型	说明
result	List<T>	结果列表。
cursor	String	分页游标。

onFailure: 操作失败时调用。

```
fun onFailure(
    code: Int,
    desc: String
)
```

参数名	类型	说明
code	Int	错误码。
desc	String	错误描述。

AudioEffectStore

最近更新时间：2026-04-20 17:34:06

简介

`AudioEffectStore` 提供了一套完整的音效管理 API，包括变声效果、混响效果和耳返功能。 通过该类，主播可以在直播过程中实时调整自己的声音效果，提升直播体验。

> **重要：**
> 使用 **AudioEffectStore.shared** 单例获取 `AudioEffectStore` 实例。设置的音效在退出房间后会自动失效，下次进房需要重新设置。

> **说明：**
> 音效状态更新通过 **audioEffectState** 发布者传递。订阅它以接收有关变声、混响和耳返状态的实时更新。

> **警告：**
> 由于蓝牙耳机的硬件延迟非常高，在主播佩戴蓝牙耳机时无法开启耳返功能。请在用户界面上提示主播佩戴有线耳机。

功能特性

- **变声效果**：支持多种变声特效，如熊孩子、小女孩、大叔等。
- **混响效果**：支持多种混响特效，如 KTV、小房间、大会堂等。
- **耳返功能**：主播可在耳机中听到自己的声音，适用于唱歌场景。
- **音量控制**：支持耳返音量的精细调节。

可订阅数据

AudioEffectState 的字段描述如下：

属性名	类型	描述
audioChangerType	StateFlow<AudioChangerType>	变声状态。
audioReverbType	StateFlow<AudioReverbType>	混响状态。

isEarMonitorOpened	StateFlow<Boolean>	耳返开启。
earMonitorVolume	StateFlow<Int>	耳返音量，取值范围0 – 100。 如果将 volume 设置成100之后感觉音量还是太小，可以将 volume 最大设置成150，但超过100的 volume 会有爆音的风险，请谨慎操作。

API 列表

函数名	描述
AudioEffectStore.shared	获取单例实例。
setAudioChangerType	设置变声效果。
setAudioReverbType	设置混响效果。
setVoiceEarMonitorEnable	开启/关闭耳返。
setVoiceEarMonitorVolume	设置耳返音量。
reset	重置为默认状态。

获取实例

AudioEffectStore.shared

获取单例实例。

变声设置

setAudioChangerType

设置变声效果

```
abstract fun setAudioChangerType(type: AudioChangerType)
```

通过该接口您可以设置人声的变声特效。

变声特效可以作用于人声之上，通过声学算法对人声进行二次处理，以获得与原始声音所不同的音色。

版本信息

从3.5版本开始支持。

调用时机

进入房间后，且需要使用变声效果时调用。

注意事项

> **说明：**
> 设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

参数说明

参数名	类型	是否必填	描述
type	AudioChangerType	必填	变声效果类型。

混响设置

setAudioReverbType

设置混响效果

```
abstract fun setAudioReverbType(type: AudioReverbType)
```

通过该接口您可以设置人声的混响效果。

混响特效可以作用于人声之上，通过声学算法对声音进行叠加处理，模拟出各种不同环境下的临场感受。

版本信息

从3.5版本开始支持。

调用时机

进入房间后，且需要使用混响效果时调用。

注意事项

> **说明：**
> 设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

参数说明

参数名	类型	是否必填	描述

type	AudioReverbType	必填	混响效果类型。

耳返设置

setVoiceEarMonitorEnable

开启/关闭耳返

```
abstract fun setVoiceEarMonitorEnable(enable: Boolean)
```

主播开启耳返后，可以在耳机里听到麦克风采集到的自己发出的声音，该特效适用于主播唱歌的应用场景中。

版本信息

从3.5版本开始支持。

适用场景

适用于主播唱歌场景，让主播能够实时听到自己的声音以便调整演唱效果。

调用时机

进入房间后，且主播佩戴有线耳机时调用。

注意事项

> **警告：**
> 由于蓝牙耳机的硬件延迟非常高，所以在主播佩戴蓝牙耳机时无法开启此特效，请尽量在用户界面上提示主播佩戴有线耳机。

参数说明

参数名	类型	是否必填	描述
enable	Boolean	必填	是否开启耳返。

setVoiceEarMonitorVolume

设置耳返音量

```
abstract fun setVoiceEarMonitorVolume(volume: Int)
```

通过该接口您可以设置耳返特效中声音的音量大小。

版本信息

从3.5版本开始支持。

调用时机

开启耳返功能后调用。

注意事项

> **说明：**
> 如果将 volume 设置成100之后感觉音量还是太小，可以将 volume 最大设置成150，但超过100的 volume 会有爆音的风险，请谨慎操作。

参数说明

参数名	类型	是否必填	描述
volume	Int	必填	耳返音量。（取值范围：0 – 100（超过100可能导致爆音））（默认值：100）。

重置

reset

重置为默认状态

```
abstract fun reset()
```

将所有音效设置重置为默认值，包括关闭变声效果、关闭混响效果、关闭耳返并重置耳返音量。
版本信息
从3.5版本开始支持。
调用时机
需要恢复默认音效设置时调用，例如退出直播间前。

数据结构

AudioChangerType

变声效果类型。

枚举值	值	说明
NONE	0	关闭特效。
CHILD	1	熊孩子。
LITTLE_GIRL	2	小女孩。
MAN	3	大叔。

HEAVY_METAL	4	重金属。
COLD	5	感冒。
FOREIGNER	6	外语腔。
TRAPPED_BEAST	7	困兽。
FATSO	8	肥宅。
STRONG_CURRENT	9	强电流。
HEAVY_MACHINERY	10	重机械。
ETHEREAL	11	空灵。

AudioReverbType

混响效果类型。

枚举值	值	说明
NONE	0	关闭特效。
KTV	1	KTV。
SMALL_ROOM	2	小房间。
AUDITORIUM	3	大会堂。
DEEP	4	低沉。
LOUD	5	洪亮。
METALLIC	6	金属声。
MAGNETIC	7	磁性。

AudioEffectState

AudioEffectStore 对外提供的音效相关状态数据。

属性	类型	说明
audioChangerType	StateFlow<AudioChangerType>	变声状态。

audioReverbType	StateFlow<AudioReverbType>	混响状态。
isEarMonitorOpened	StateFlow<Boolean>	耳返开启。
earMonitorVolume	StateFlow<Int>	耳返音量，取值范围0 – 100。 如果将 volume 设置成100之后感觉音量还是太小，可以将 volume 最大设置成150，但超过100的 volume 会有爆音的风险，请谨慎操作。

使用示例

```
// 获取单例实例
val store = AudioEffectStore.shared()
// 订阅状态变化
lifecycleScope.launch {
    store.audioEffectState.audioChangerType.collect { type ->
        println("当前变声效果：$type")
    }
}
lifecycleScope.launch {
    store.audioEffectState.audioReverbType.collect { type ->
        println("当前混响效果：$type")
    }
}
// 设置变声效果
store.setAudioChangerType(AudioChangerType.LITTLE_GIRL)
// 设置混响效果
store.setAudioReverbType(AudioReverbType.KTV)
// 开启耳返
store.setVoiceEarMonitorEnable(true)
store.setVoiceEarMonitorVolume(80)
```

BarrageStore

最近更新时间：2026-04-20 17:34:06

简介

`BarrageStore` 提供了一套完整的弹幕管理 API，包括发送文本弹幕、发送自定义弹幕和添加本地提示消息。 通过该类，可以在直播间内实现弹幕互动功能。

> **重要：**
> 使用 **BarrageStore.create** 工厂方法创建 `BarrageStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 弹幕状态更新通过 **barrageState** 发布者传递。订阅它以接收房间内弹幕数据的实时更新。

功能特性

- **文本弹幕**：支持发送纯文本弹幕消息。
- **自定义弹幕**：支持发送自定义格式的弹幕（如带特效的弹幕）。
- **本地提示**：支持添加仅本地可见的提示消息。

可订阅数据

BarrageState 的字段描述如下：

属性名	类型	描述
messageList	StateFlow<List< Barrage >>	当前房间的弹幕消息列表，支持实时更新并可被订阅监听。

API 列表

函数名	描述
BarrageStore.create	创建弹幕管理实例。
customMessageEvent	自定义消息事件发布者。
sendTextMessage	发送文本弹幕。

sendCustomMessage	发送自定义弹幕。
appendLocalTip	添加本地提示消息。

创建实例

BarrageStore.create

创建弹幕管理实例。

观察事件

customMessageEvent

自定义消息事件发布者。

发送弹幕

sendTextMessage

发送文本类型弹幕。

```
abstract fun sendTextMessage(
        text: String?,
        extensionInfo: Map<String, String>?,
        completion: CompletionHandler?
    )
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
text	String?	必填	文本弹幕内容。
extensionInfo	Map<String, String>?	必填	扩展信息，可包含自定义字段（如指定弹幕颜色、字体大小等）。
completion	CompletionHandler?	必填	完成回调（成功/失败状态）。

sendCustomMessage

发送自定义类型弹幕。

```
abstract fun sendCustomMessage(
        businessID: String?,
        data: String?,
        completion: CompletionHandler?
    )
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
businessID	String?	必填	业务标识 ID，用于区分不同业务场景的自定义弹幕。
data	String?	必填	自定义数据内容，通常为 JSON 格式字符串，用于传递业务自定义的数据。
completion	CompletionHandler?	必填	完成回调（成功/失败状态）。

本地消息

appendLocalTip

添加本地提示消息（在本地添加提示或操作反馈消息，仅当前客户端可见）。

```
abstract fun appendLocalTip(message: Barrage)
```

版本信息

从3.5版本开始支持。

注意事项

> **说明：**
> 该消息仅在本地显示，不会通过网络发送给其他用户。

参数说明

参数名	类型	是否必填	描述

message	Barrage	必填	本地弹幕消息（如系统提示、操作反馈等，仅当前用户可见）。

数据结构

BarrageType

弹幕类型枚举，用于区分不同的弹幕消息种类。

枚举值	值	说明
TEXT	0	文本类型弹幕，包含纯文字内容。
CUSTOM	1	自定义类型弹幕，支持业务自定义数据格式（如带特效的弹幕、互动消息等）。

Barrage

弹幕数据模型，包含单条弹幕的完整属性信息。

属性	类型	说明
liveID	String	弹幕所属直播间/语音聊天房的唯一标识 ID。
sender	LiveUserInfo	弹幕发送者的用户信息（如用户 ID、昵称、头像等）。
sequence	Long	弹幕消息的唯一序列 ID，用于消息排序和去重。
timestampInSecond	Long	弹幕发送时间戳（单位：秒），用于展示发送时间顺序。
messageType	BarrageType	弹幕消息类型（文本或自定义）。
textContent	String	文本类型弹幕的消息内容，即弹幕的文本内容。
extensionInfo	Map<String, String>	弹幕扩展信息，可自定义字段（如显示样式、优先级等）。当 messageType 为 TEXT 时有效。
businessID	String	自定义类型弹幕的业务标识 ID，用于区分不同业务场景的自定义弹幕。

data	String	自定义类型弹幕的具体数据内容（通常为 JSON 格式字符串），当 messageType 为 CUSTOM 时有效。

BarrageState

弹幕状态，管理当前房间的弹幕数据状态。

属性	类型	说明
messageList	StateFlow<List<Barrage>>	当前房间的弹幕消息列表，支持实时更新并可被订阅监听。

BaseBeautyStore

最近更新时间：2026-04-20 17:34:06

简介

基础美颜功能通过简单易用的 API 实现实时美颜效果调节。`BaseBeautyStore` 提供了一套完整的接口来管理美颜效果的设置和状态订阅。

> **说明：**
> 美颜状态更新通过 **baseBeautyState** 发布者传递。订阅它以接收有关美颜效果级别的实时更新。

功能特性

- **磨皮效果**：支持0 – 9级别的磨皮效果调节。
- **美白效果**：支持0 – 9级别的美白效果调节。
- **红润效果**：支持0 – 9级别的红润效果调节。
- **状态订阅**：实时订阅美颜状态变化，同步 UI 显示与实际效果。

可订阅数据

BaseBeautyState 的字段描述如下：

属性名	类型	描述
smoothLevel	StateFlow<Float>	磨皮级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。
whitenessLevel	StateFlow<Float>	美白级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。
ruddyLevel	StateFlow<Float>	红润级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。

API 列表

函数名	描述
BaseBeautyStore.shared	获取单例实例。
setSmoothLevel	设置磨皮级别。

setWhitenessLevel	设置美白级别。
setRuddyLevel	设置红润级别。
reset	重置为默认状态。

获取实例

BaseBeautyStore.shared

获取单例实例。

美颜调节

setSmoothLevel

设置磨皮级别

```
abstract fun setSmoothLevel(smoothLevel: Float)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
smoothLevel	Float	必填	磨皮级别，取值范围 0, 9；0表示关闭，9表示效果最明显。

setWhitenessLevel

设置美白级别

```
abstract fun setWhitenessLevel(whitenessLevel: Float)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
whitenessLevel	Float	必填	美白级别，取值范围 0, 9；0表示关闭，9表示效果最明显。

setRuddyLevel

设置红润级别

```
abstract fun setRuddyLevel(ruddyLevel: Float)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
ruddyLevel	Float	必填	红润级别，取值范围 0, 9；0表示关闭，9表示效果最明显。

reset

将所有美颜参数（磨皮、美白、红润）重置为默认关闭状态（值为0）。

```
abstract fun reset()
```

版本信息
从3.5版本开始支持。

数据结构

BaseBeautyState

基础美颜状态，管理磨皮、美白、红润等美颜效果的级别数据。支持订阅以同步 UI 显示与实际效果。

属性	类型	说明
smoothLevel	StateFlow<Float>	磨皮级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。
whitenessLevel	StateFlow<Float>	美白级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。
ruddyLevel	StateFlow<Float>	红润级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。

使用示例

```
// 获取单例实例
val store = BaseBeautyStore.shared()
// 订阅状态变化
lifecycleScope.launch {
    store.baseBeautyState.smoothLevel.collect { level ->
        println("磨皮级别：$level")
    }
}
// 设置美颜效果
store.setSmoothLevel(5f)
store.setWhitenessLevel(3f)
store.setRuddyLevel(2f)
// 重置所有美颜效果
store.reset()
```

BattleStore

最近更新时间：2026-04-20 17:34:06

简介

PK 功能实现主播之间的实时互动对战。`BattleStore` 提供了一套全面的 API 来管理整个 PK 生命周期。

> **重要：**
> 请始终使用工厂方法 **BattleStore.create** 并提供有效的直播间 ID 来创建 `BattleStore` 实例。不要尝试直接初始化。

> **说明：**
> PK 状态更新通过 **battleState** 发布者传递。订阅它以接收有关 PK 信息、参与用户和分数的实时更新。

> **警告：**
> 如果 PK 请求在指定的超时时间内未收到响应，将触发超时事件。请始终在 UI 中处理超时场景。

功能特性

- **PK 请求管理**：主播可以发起 PK 请求，被邀请方可以接受或拒绝。
- **状态管理**：实时跟踪 PK 信息、参与用户和分数。
- **事件驱动架构**：提供完整的 PK 事件回调。
- **超时处理**：为 PK 请求提供内置超时机制。

可订阅数据

BattleState 的字段描述如下：

属性名	类型	描述
currentBattleInfo	StateFlow<BattleInfo?>	当前 PK 信息。
battleUsers	StateFlow<List<SeatUserInfo>>	PK 用户列表。
battleScore	StateFlow<Map<String, Int>>	PK 分数的映射。

API 列表

函数名	描述
BattleStore.create	创建 BattleStore 实例。
addBattleListener	新增 PK 事件回调。
removeBattleListener	移除 PK 事件回调。
requestBattle	发起 PK 请求。
cancelBattleRequest	取消 PK 请求。
acceptBattle	接受 PK 请求。
rejectBattle	拒绝 PK 请求。
exitBattle	退出 PK。

创建实例

BattleStore.create

创建 BattleStore 实例。

观察状态和事件

addBattleListener

新增 PK 事件回调

```
abstract fun addBattleListener(listener: BattleListener?)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	BattleListener?	必填	监听器。

removeBattleListener

移除 PK 事件回调

```
abstract fun removeBattleListener(listener: BattleListener?)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	BattleListener?	必填	监听器。

PK 操作

requestBattle

发起 PK 请求

```
abstract fun requestBattle(
    config: BattleConfig,
    userIDList: List<String>,
    timeout: Int,
    completion: BattleRequestCallback?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
config	BattleConfig	必填	PK 配置。
userIDList	List<String>	必填	需要进入 PK 的用户 ID 列表。
timeout	Int	必填	请求超时时间。
completion	BattleRequestCallback?	必填	发起请求成功的回调。

cancelBattleRequest

取消 PK 请求

```
abstract fun cancelBattleRequest(
    battleID: String?,
    userIDList: List<String>,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
battleID	String?	必填	PK ID。
userIDList	List<String>	必填	用户 ID 列表。
completion	CompletionHandler?	必填	取消请求成功的回调。

acceptBattle

接受 PK 请求

```
abstract fun acceptBattle(
    battleID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
battleID	String?	必填	PK ID。
completion	CompletionHandler?	必填	接受成功的回调。

rejectBattle

拒绝 PK 请求

```
abstract fun rejectBattle(
    battleID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
battleID	String?	必填	PK ID。
completion	CompletionHandler?	必填	拒绝成功的回调。

exitBattle

退出 PK

```
abstract fun exitBattle(
    battleID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
battleID	String?	必填	PK ID。
completion	CompletionHandler?	必填	退出 PK 成功的回调。

数据结构

BattleEndedReason

正在 PK 中用户收到 PK 结束的原因。

枚举值	值	说明
TIME_OVER	0	PK 倒计时结束。
ALL_MEMBER_EXIT	1	所有 PK 成员退出。

BattleListener

收到 PK 相关的回调事件。

方法

方法名	说明
onBattleStarted	当 PK 正式开始时触发此回调，通知所有参与者 PK 已经开始。
onBattleEnded	当 PK 结束时触发此回调。
onUserJoinBattle	当有用户加入 PK 时触发此回调。
onUserExitBattle	当有用户退出 PK 时触发此回调。
onBattleRequestReceived	当收到 PK 请求时触发此回调。
onBattleRequestCancelled	当 PK 请求被取消时触发此回调。
onBattleRequestTimeout	当 PK 请求超时时触发此回调。
onBattleRequestAccept	当 PK 请求被接受时触发此回调。
onBattleRequestReject	当 PK 请求被拒绝时触发此回调。

BattleConfig

发送 PK 请求时，设置的 PK 配置信息。

属性	类型	说明
duration	Int	PK 持续时间（单位：秒）。
needResponse	Boolean	被邀请用户是否需要回复同意/拒绝。

extensionInfo	String	扩展信息。

BattleInfo

PK 信息。

属性	类型	说明
battleID	String	PK ID。
config	BattleConfig	发送 PK 请求时，设置的 PK 配置信息。
startTime	Long	PK 开始标记时间戳（单位：秒）。
endTime	Long	PK 结束标记时间戳（单位：秒）。

BattleState

BattleStore 对外提供的 PK 相关状态数据。

属性	类型	说明
currentBattleInfo	StateFlow<BattleInfo?>	当前 PK 信息。
battleUsers	StateFlow<List<SeatUserInfo>>	PK 用户列表。
battleScore	StateFlow<Map<String, Int>>	PK 分数的映射。

BattleRequestCallback

PK 请求回调。
PK 请求的回调接口，用于处理 PK 请求的成功或失败结果。
方法
onSuccess: 成功回调。

```
fun onSuccess(battleInfo: BattleInfo, resultMap: Map<String, Int>)
```

参数名	类型	说明
battleInfo	BattleInfo	PK 信息，包含 PK 的详细配置和状态。

resultMap	Map<String, Int>	PK 请求的响应结果回调。

onError: 失败回调。

```
fun onError(code: Int, desc: String)
```

参数名	类型	说明
code	Int	错误码。
desc	String	错误描述。

使用示例

```
// 创建 store 实例
val store = BattleStore.create("live_room_123")
// 订阅状态变化
lifecycleScope.launch {
    store.battleState.currentBattleInfo.collect { battleInfo ->
        battleInfo?.let {
            println("当前 PK ID: ${it.battleID}")
        }
    }
}
// 添加 PK 事件监听器
store.addBattleListener(object : BattleListener() {
    override fun onBattleStarted(battleInfo: BattleInfo, inviter:
SeatUserInfo, invitees: List<SeatUserInfo>) {
        println("PK 开始，发起者: ${inviter.userName}")
    }
    override fun onBattleEnded(battleInfo: BattleInfo, reason:
BattleEndedReason?) {
        println("PK 结束，原因: $reason")
    }
})
// 发起 PK 请求
val config = BattleConfig(duration = 300, needResponse = true)
store.requestBattle(config, listOf("user_456"), 30, object :
BattleRequestCallback {
```

```
    override fun onSuccess(battleInfo: BattleInfo, resultMap:
Map<String, Int>) {
        println("PK 请求成功: ${battleInfo.battleID}")
    }
    override fun onError(code: Int, desc: String) {
        println("PK 请求失败: $desc")
    }
})
```

CameraView

最近更新时间：2026-04-20 17:34:06

简介

`CameraView` 是一个专门用于显示摄像头预览画面的视图组件。 当调用开始摄像头测试时，摄像头的渲染画面会显示到此视图上。

> **说明：**
> 此类仅在 Android 平台可用。

功能特性

- **摄像头预览**：用于显示摄像头测试时的实时画面。
- **多种构造方式**：支持多种构造函数，适应不同的使用场景。

API 列表

函数名	描述
CameraView(Context)	构造函数。
CameraView(SurfaceView)	构造函数。
CameraView(Context, AttributeSet)	构造函数。
CameraView(Context, AttributeSet, SurfaceView)	构造函数。

创建实例

CameraView(Context)

构造函数。

CameraView(SurfaceView)

构造函数。

CameraView(Context, AttributeSet)

构造函数。

CameraView(Context, AttributeSet, SurfaceView)

构造函数。

使用示例

```
// 方式一：通过 Context 创建
val cameraView = CameraView(context)
// 方式二：通过 SurfaceView 创建
val surfaceView = SurfaceView(context)
val cameraView = CameraView(surfaceView)
// 方式三：在 XML 布局中使用
// <io.trtc.tuikit.atomicxcore.api.view.CameraView
//     android:id="@+id/camera_view"
//     android:layout_width="match_parent"
//     android:layout_height="match_parent" />
```

CoGuestStore

最近更新时间：2026-04-20 17:34:06

简介

连麦功能通过基于麦位的系统实现主播和观众成员之间的实时互动。`CoGuestStore` 提供了一套全面的 API 来管理整个连麦生命周期。

> **重要：**
> 请始终使用工厂方法 **CoGuestStore.create** 并提供有效的直播间 ID 来创建 `CoGuestStore` 实例。不要尝试直接初始化。

> **说明：**
> 连麦状态更新通过 **coGuestState** 发布者传递。订阅它以接收有关已连接用户、邀请和申请的实时更新。

> **警告：**
> 如果连麦请求在指定的超时时间内未收到响应，将触发带有 `NoResponseReason.timeout` 的事件。请始终在 UI 中处理超时场景。

功能特性

- **双向邀请**：主播可以邀请观众成员，观众成员也可以申请加入。
- **状态管理**：实时跟踪已连接用户、邀请和申请。
- **事件驱动架构**：为主播和观众角色提供独立的事件流。
- **超时处理**：为邀请和申请提供内置超时机制。

可订阅数据

CoGuestState 的字段描述如下：

属性名	类型	描述
connected	StateFlow<List<SeatUserInfo>>	已经在麦位上的用户列表。
invitees	StateFlow<List<LiveUserInfo>>	主播发出邀请的用户列表。
applicants	StateFlow<List<LiveUserInfo>>	主播收到申请连麦的用户列表。

candidates	StateFlow<List<LiveUserInfo>>	候选连麦的用户列表。

API 列表

函数名	描述
CoGuestStore.create	创建对象实例。
addHostListener	主播端事件回调。
removeHostListener	主播端事件回调。
addGuestListener	观众端事件回调。
removeGuestListener	观众端事件回调。
applyForSeat	观众申请连麦。
cancelApplication	观众取消申请。
acceptApplication	主播接受申请。
rejectApplication	主播拒绝申请。
inviteToSeat	主播邀请观众连麦。
cancelInvitation	主播取消邀请。
acceptInvitation	观众接受邀请。
rejectInvitation	观众拒绝邀请。
disconnect	结束连麦会话。

创建实例

CoGuestStore.create

创建对象实例。

观察状态和事件

addHostListener

添加主播侧事件回调监听器

```
abstract fun addHostListener(
    listener: HostListener?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	HostListener?	必填	监听器。

removeHostListener

移除主播侧事件回调监听器

```
abstract fun removeHostListener(
    listener: HostListener?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	HostListener?	必填	监听器。

addGuestListener

添加观众侧事件回调监听器

```
abstract fun addGuestListener(
    listener: GuestListener?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	GuestListener?	必填	监听器。

removeGuestListener

移除观众侧事件回调监听器

```
abstract fun removeGuestListener(
    listener: GuestListener?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	GuestListener?	必填	监听器。

观众操作

applyForSeat

申请上麦

```
abstract fun applyForSeat(
    seatIndex: Int,
    timeout: Int,
    extraInfo: String?,
    completion: CompletionHandler?
)
```

以观众身份请求加入连麦会话。

调用此方法后，会向直播间内的所有主播发送连麦请求。请求将保持活动状态，直到：

- 主播通过 acceptApplication 接受

- 主播通过 rejectApplication 拒绝
- 超时时间到期
- 您通过 cancelApplication 取消

版本信息

从3.5版本开始支持。

注意事项

> **说明：**
> 如果在超时时间内没有主播响应，将触发带有 NoResponseReason.TIMEOUT 的 {ref2} 事件。

参数说明

参数名	类型	是否必填	描述
seatIndex	Int	必填	麦位索引，-1表示自动分配麦位。
timeout	Int	必填	超时时间（单位：秒）。
extraInfo	String?	必填	额外信息。
completion	CompletionHandler?	必填	完成回调。

cancelApplication

取消上麦申请

```
abstract fun cancelApplication(
    completion: CompletionHandler?
)
```

取消之前发送的连麦申请。调用此方法后，所有主播将收到申请取消的通知。

版本信息

从3.5版本开始支持。

注意事项

> **说明：**
> 如果申请已被主播处理，取消操作可能无效。

参数说明

参数名	类型	是否必填	描述

completion	CompletionHandler?	必填	完成回调。

acceptApplication

接受上麦申请

```
abstract fun acceptApplication(
    userID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String?	必填	用户 ID。
completion	CompletionHandler?	必填	完成回调。

rejectApplication

拒绝上麦申请

```
abstract fun rejectApplication(
    userID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String?	必填	用户 ID。
completion	CompletionHandler?	必填	完成回调。

主播操作

inviteToSeat

邀请观众上麦

```
abstract fun inviteToSeat(
    inviteeID: String?,
    seatIndex: Int,
    timeout: Int,
    extraInfo: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
inviteeID	String?	必填	被邀请用户 ID。
seatIndex	Int	必填	麦位索引，-1表示自动分配麦位。
timeout	Int	必填	超时时间（单位：秒）。
extraInfo	String?	必填	额外信息。
completion	CompletionHandler?	必填	完成回调。

cancelInvitation

取消上麦邀请

```
abstract fun cancelInvitation(
    inviteeID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
inviteeID	String?	必填	被邀请用户 ID。
completion	Completion Handler?	必填	完成回调。

acceptInvitation

接受上麦邀请

```
abstract fun acceptInvitation(
    inviterID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
inviterID	String?	必填	邀请者用户 ID。
completion	Completion Handler?	必填	完成回调。

rejectInvitation

拒绝上麦邀请

```
abstract fun rejectInvitation(
    inviterID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

inviterID	String?	必填	邀请者用户 ID。
completion	Completion Handler?	必填	完成回调。

连接控制

disconnect

结束连麦会话。

数据结构

NoResponseReason

主播发出的连麦邀请或者观众发起连麦请求的无响应原因。

枚举值	值	说明
TIMEOUT	0	请求超时。
ALREADY_SEATED	1	用户已在麦位上。

HostListener

主播侧收到的回调事件。

方法

方法名	说明
onGuestApplicationReceived	当观众申请连麦时触发此回调。
onGuestApplicationCancelled	当观众取消连麦申请时触发此回调。
onGuestApplicationProcessedByOtherHost	当观众的连麦申请被其他主播处理时触发此回调。
onHostInvitationResponded	当主播发出的连麦邀请收到观众响应时触发此回调。
onHostInvitationNoResponse	当主播发出的连麦邀请无响应时触发此回调。

GuestListener

观众侧收到的回调事件。

方法

方法名	说明
onHostInvitationReceived	当收到主播的连麦邀请时触发此回调。
onHostInvitationCancelled	当主播取消连麦邀请时触发此回调。
onGuestApplicationResponded	当观众的连麦申请收到主播响应时触发此回调。
onGuestApplicationNoResponse	当观众的连麦申请无响应时触发此回调。
onKickedOffSeat	当观众被主播踢下麦位时触发此回调。

CoGuestState

CoGuestStore 对外提供的连麦相关状态数据。

属性	类型	说明
connected	StateFlow<List<SeatUserInfo>>	已经在麦位上的用户列表。
invitees	StateFlow<List<LiveUserInfo>>	主播发出邀请的用户列表。
applicants	StateFlow<List<LiveUserInfo>>	主播收到申请连麦的用户列表。
candidates	StateFlow<List<LiveUserInfo>>	候选连麦的用户列表。

使用示例

```
// 创建 store 实例
val store = CoGuestStore.create("live_room_123")
// 订阅状态变化
lifecycleScope.launch {
```

```
    store.coGuestState.connected.collect { connected ->
        println("已连接用户: ${connected.size}")
    }
}
// 添加主播事件监听器（主播使用）
store.addHostListener(object : HostListener() {
    override fun onGuestApplicationReceived(guestUser: LiveUserInfo) {
        println("收到来自 ${guestUser.userName} 的申请")
        // 显示接受/拒绝的 UI
    }
    override fun onHostInvitationResponded(isAccept: Boolean, guestUser:
LiveUserInfo) {
        println("观众 ${guestUser.userName} ${if (isAccept) "接受了" else
"拒绝了"}")
    }
})
// 主播：接受申请
store.acceptApplication("user_456") { code, message ->
    if (code == 0) {
        println("申请接受成功")
    }
}
```

CoHostStore

最近更新时间：2026-04-20 17:34:06

简介

跨房连线功能允许不同直播间的主播进行实时互动。`CoHostStore` 提供了一套全面的 API 来管理整个跨房连线生命周期。

> **重要：**
> 请始终使用工厂方法 **CoHostStore.create** 并提供有效的直播间 ID 来创建 `CoHostStore` 实例。不要尝试直接初始化。

> **说明：**
> 连线状态更新通过 **coHostState** 发布者传递。订阅它以接收有关连线状态、已连接主播、邀请和申请的实时更新。

> **警告：**
> 如果连线请求在指定的超时时间内未收到响应，将触发超时事件。请始终在 UI 中处理超时场景。

功能特性

- **双向连线**：主播可以向其他主播发起连线请求，也可以接收其他主播的连线请求。
- **状态管理**：实时跟踪连线状态、已连接主播、邀请列表和申请者。
- **事件驱动架构**：提供连线事件流用于监听各种连线状态变化。
- **布局模板**：支持多种连线布局模板，如动态网格布局和1对6布局。

可订阅数据

CoHostState 的字段描述如下：

属性名	类型	描述
coHostStatus	StateFlow<CoHostStatus>	跨房连线的实时状态。
connected	StateFlow<List<SeatUserInfo>>	正在和当前直播间连线的主播列表。
invitees	StateFlow<List<SeatUserInfo>>	向其他直播间发出请求的主播列表。

applicant	StateFlow<SeatUserInfo?>	向当前直播间发起连线请求的主播。
candidatesCursor	StateFlow<String>	推荐用户列表游标。
candidates	StateFlow<List<SeatUserInfo>>	推荐用户列表。

API 列表

函数名	描述
CoHostStore.create	创建对象实例。
addCoHostListener	连线事件回调。
removeCoHostListener	连线事件回调。
requestHostConnection	发起连线请求。
cancelHostConnection	取消连线请求。
acceptHostConnection	接受连线请求。
rejectHostConnection	拒绝连线请求。
exitHostConnection	退出连线。
muteRemoteHostAudio	禁音/取消禁音远端主播的音频。
getCoHostCandidates	获取推荐主播列表。

创建实例

CoHostStore.create

创建对象实例。

观察状态和事件

addCoHostListener

添加连线回调监听器

```
abstract fun addCoHostListener(
    listener: CoHostListener?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	CoHostListener?	必填	监听器。

removeCoHostListener

移除连线回调监听器

```
abstract fun removeCoHostListener(
    listener: CoHostListener?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	CoHostListener?	必填	监听器。

连线操作

requestHostConnection

发起主播连线请求

```
abstract fun requestHostConnection(
```

```
    targetHostLiveID: String?,
    layoutTemplate: CoHostLayoutTemplate,
    timeout: Int,
    extraInfo: String?,
    completion: CompletionHandler?
)
```

向目标主播发起跨房连线请求。

调用此方法后，会向目标主播发送连线请求。请求将保持活动状态，直到：
- 目标主播通过 **acceptHostConnection(fromHostLiveID:completion:)** 接受
- 目标主播通过 **rejectHostConnection(fromHostLiveID:completion:)** 拒绝
- 超时时间到期
- 您通过 **cancelHostConnection(toHostLiveID:completion:)** 取消

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
targetHostLiveID	String?	必填	目标主播的直播间 ID。
layoutTemplate	CoHostLayoutTemplate	必填	连线布局模板。
timeout	Int	必填	请求超时时间（单位：秒）。
extraInfo	String?	必填	扩展信息。
completion	CompletionHandler?	必填	发起请求成功的回调。

cancelHostConnection

取消主播连线请求

```
abstract fun cancelHostConnection(
    toHostLiveID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
toHostLiveID	String?	必填	目标主播的直播间 ID。
completion	Completion Handler?	必填	取消请求成功的回调。

acceptHostConnection

接受主播连线请求

```
abstract fun acceptHostConnection(
    fromHostLiveID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
fromHostLiveID	String?	必填	发起连线请求的主播直播间 ID。
completion	Completion Handler?	必填	接受成功的回调。

rejectHostConnection

拒绝主播连线请求

```
abstract fun rejectHostConnection(
    fromHostLiveID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
fromHostLiveID	String?	必填	发起连线请求的主播直播间 ID。
completion	Completion Handler?	必填	拒绝成功的回调。

exitHostConnection

退出主播连线

```
abstract fun exitHostConnection(
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
completion	Completion Handler?	必填	退出连线成功的回调。

muteRemoteHostAudio

禁音/取消禁音远端主播的音频

```
abstract fun muteRemoteHostAudio(
    liveID: String,
    isMuted: Boolean,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveID	String	必填	远端主播的直播间 ID。

isMuted	Boolean	必填	是否禁音远端主播的音频。`true` 表示禁音，`false` 表示取消禁音。
completion	CompletionHandler?	必填	操作结果的回调。

getCoHostCandidates

获取可以与当前主播连线的推荐主播列表

```
abstract fun getCoHostCandidates(
    cursor: String,
    completion: CompletionHandler?
)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
cursor	String	必填	游标。
completion	CompletionHandler?	必填	完成回调。

数据结构

CoHostStatus

当前用户的跨房连线状态。

枚举值	值	说明
CONNECTED	0	和其他主播正在连线中。
DISCONNECTED	1	没有和其他主播连线。

CoHostLayoutTemplate

连线布局模板。

枚举值	值	说明

HOST_VOICE_CONNECTION	2	语聊房连线布局。
HOST_DYNAMIC_GRID	600	主播动态网格布局。
HOST_DYNAMIC_1V6	601	主播动态1对6布局。
HOST_VIDEO_LEFT_FOCUS_9_SEATS	602	主播视频左焦点9席位布局。
HOST_VIDEO_UNIFORM_GRID_9_SEATS	603	主播视频均匀网格9席位布局。

CoHostListener

连线请求的回调事件。

方法

方法名	说明
onCoHostRequestReceived	当收到连线请求时触发此回调。
onCoHostRequestCancelled	当连线请求被取消时触发此回调。
onCoHostRequestAccepted	当连线请求被接受时触发此回调。
onCoHostRequestRejected	当连线请求被拒绝时触发此回调。
onCoHostRequestTimeout	当连线请求超时时触发此回调。
onCoHostUserJoined	当用户加入连线时触发此回调。
onCoHostUserLeft	当用户离开连线时触发此回调。

CoHostState

CoHostStore 对外提供的跨房连线相关状态数据。

属性	类型	说明
coHostStatus	StateFlow<CoHostStatus>	跨房连线的实时状态。
connected	StateFlow<List<SeatUserInfo>>	正在和当前直播间连线的主播列表。
invitees	StateFlow<List<SeatUserInfo>>	向其他直播间发出请求的主播列表。
applicant	StateFlow<SeatUserInfo?>	向当前直播间发起连线请求的主播。
candidatesCursor	StateFlow<String>	推荐用户列表游标。
candidates	StateFlow<List<SeatUserInfo>>	推荐用户列表。

使用示例

```
// 创建 store 实例
val store = CoHostStore.create("live_room_123")
// 订阅状态变化
lifecycleScope.launch {
    store.coHostState.coHostStatus.collect { status ->
        println("连线状态: $status")
    }
}
lifecycleScope.launch {
    store.coHostState.connected.collect { connected ->
        println("已连接主播: ${connected.size}")
    }
}
// 添加连线事件监听器
store.addCoHostListener(object : CoHostListener() {
    override fun onCoHostRequestReceived(inviter: SeatUserInfo,
extensionInfo: String) {
        println("收到来自 ${inviter.userName} 的连线请求")
        // 显示接受/拒绝的 UI
    }
    override fun onCoHostRequestAccepted(invitee: SeatUserInfo) {
        println("连线请求被 ${invitee.userName} 接受")
    }
```

```
    override fun onCoHostUserJoined(userInfo: SeatUserInfo) {
        println("主播 ${userInfo.userName} 加入连线")
    }
})
// 发起连线请求
store.requestHostConnection(
    targetHostLiveID = "target_live_id",
    layoutTemplate = CoHostLayoutTemplate.HOST_DYNAMIC_GRID,
    timeout = 30,
    extraInfo = "",
    completion = { code, message ->
        if (code == 0) {
            println("连线请求发送成功")
        }
    }
)
```

DeviceStore

最近更新时间：2026-04-20 17:34:06

简介

`DeviceStore` 提供了一套全面的 API 来管理音视频设备，包括麦克风、摄像头和屏幕分享等功能。

> **重要：**
> 请使用 **DeviceStore.shared** 单例获取 `DeviceStore` 实例。不要尝试直接初始化。

> **说明：**
> 设备状态更新通过 **deviceState** 发布者传递。订阅它以接收有关麦克风、摄像头、网络等状态的实时更新。

功能特性

- **麦克风管理**：打开/关闭麦克风，设置采集音量和输出音量。
- **摄像头管理**：打开/关闭摄像头，切换前后摄像头，设置镜像和视频质量。
- **音频路由**：切换扬声器和听筒。
- **屏幕分享**：开启和关闭屏幕分享功能。
- **网络状态**：实时监控网络质量信息。

可订阅数据

DeviceState 的字段描述如下：

属性名	类型	描述
microphoneStatus	StateFlow<DeviceStatus>	麦克风状态。
microphoneLastError	StateFlow<DeviceError>	麦克风错误，用于出现报错时提取错误信息。
captureVolume	StateFlow<Int>	采集音量，取值范围 0, 100。
currentMicVolume	StateFlow<Int>	当前用户实际输出音量。
outputVolume	StateFlow<Int>	最大输出音量，取值范围 0, 100。
cameraStatus	StateFlow<DeviceStatus>	摄像头状态。

cameraLastError	StateFlow<DeviceError>	摄像头错误，用于出现报错时提取错误信息。
isFrontCamera	StateFlow<Boolean>	是否为前置摄像头。
localMirrorType	StateFlow<MirrorType>	镜像状态。
localVideoQuality	StateFlow<VideoQuality>	本地视频质量。
currentAudioRoute	StateFlow<AudioRoute>	当前音频路由位置。
screenStatus	StateFlow<DeviceStatus>	屏幕分享状态。
networkInfo	StateFlow<NetworkInfo>	网络信息。
networkType	StateFlow<NetworkType>	当前网络类型。

API 列表

函数名	描述
DeviceStore.shared	单例对象。
openLocalMicrophone	打开本地麦克风。
closeLocalMicrophone	关闭本地麦克风。
setCaptureVolume	设置采集音量。
setOutputVolume	设置输出音量。
setAudioRoute	设置音频路由。
startCameraTest	开始摄像头测试。
stopCameraTest	停止摄像头测试。
openLocalCamera	打开本地摄像头。
closeLocalCamera	关闭本地摄像头。
switchCamera	切换摄像头。

switchMirror	切换镜像状态。
updateVideoQuality	更新视频质量。
startScreenShare	开启屏幕分享。
stopScreenShare	关闭屏幕分享。
reset	重置为默认状态。

获取实例

DeviceStore.shared

单例对象。

麦克风操作

openLocalMicrophone

打开本地麦克风

```
abstract fun openLocalMicrophone(completion: CompletionHandler?)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
completion	CompletionHandler?	必填	操作是否成功。

closeLocalMicrophone

关闭本地麦克风

```
abstract fun closeLocalMicrophone()
```

版本信息
从3.5版本开始支持。

setCaptureVolume

设置采集音量

```
abstract fun setCaptureVolume(volume: Int)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
volume	Int	必填	采集音量，取值范围 0, 100。

setOutputVolume

设置最大输出音量

```
abstract fun setOutputVolume(volume: Int)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
volume	Int	必填	最大音量，取值范围 0, 100。

音频路由

setAudioRoute

设置音频路由

```
abstract fun setAudioRoute(audioRoute: AudioRoute)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
audioRoute	AudioRoute	必填	路由位置。

摄像头操作

startCameraTest

开始摄像头测试，如果摄像头打开成功，会将画面渲染到设置的 CameraView 上

```
abstract fun startCameraTest(cameraView: CameraView, completion:
CompletionHandler?)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
cameraView	CameraView	必填	摄像头采集画面的渲染视图。
completion	CompletionHandler?	必填	操作是否成功。

stopCameraTest

停止摄像头测试

```
abstract fun stopCameraTest()
```

版本信息
从3.5版本开始支持。

openLocalCamera

打开本地摄像头

```
abstract fun openLocalCamera(isFront: Boolean, completion:
CompletionHandler?)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述

isFront	Boolean	必填	是否前置摄像头。
completion	Completion Handler?	必填	操作是否成功。

closeLocalCamera

关闭本地摄像头

```
abstract fun closeLocalCamera()
```

版本信息

从3.5版本开始支持。

switchCamera

切换摄像头

```
abstract fun switchCamera(isFront: Boolean)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
isFront	Boolean	必填	是否前置摄像头。

switchMirror

切换镜像状态

```
abstract fun switchMirror(mirrorType: MirrorType)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
mirrorType	MirrorType	必填	镜像状态。

updateVideoQuality

更新视频质量

```
abstract fun updateVideoQuality(quality: VideoQuality)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
quality	VideoQuality	必填	视频质量。

屏幕分享

startScreenShare

开启屏幕分享

```
abstract fun startScreenShare()
```

版本信息
从3.5版本开始支持。

stopScreenShare

关闭屏幕采集

```
abstract fun stopScreenShare()
```

版本信息
从3.5版本开始支持。

重置

reset

重置为默认状态

```
abstract fun reset()
```

版本信息

从3.5版本开始支持。

数据结构

DeviceType

设备类型。

枚举值	值	说明
MICROPHONE	0	麦克风类型。
CAMERA	1	摄像头类型。
SCREEN_SHARE	2	屏幕分享类型。

DeviceError

设备相关错误码。

枚举值	值	说明
NO_ERROR	0	操作成功。
NO_DEVICE_DETECTED	1	未检测到设备。
NO_SYSTEM_PERMISSION	2	没有系统权限。
NOT_SUPPORT_CAPTURE	3	不支持采集。
OCCUPIED_ERROR	4	设备已占用。
UNKNOWN_ERROR	5	未知错误。

DeviceStatus

设备开启状态。

枚举值	值	说明
OFF	0	关闭。

ON	1	开启。

AudioRoute

音频路由。

枚举值	值	说明
SPEAKERPHONE	0	扬声器，使用扬声器播放（即"免提"），扬声器位于手机底部，声音偏大，适合外放音乐。
EARPIECE	1	听筒，使用听筒播放，听筒位于手机顶部，声音偏小，适合需要保护隐私的通话场景。

VideoQuality

视频质量。

枚举值	值	说明
QUALITY_360P	1	360P。
QUALITY_540P	2	540P。
QUALITY_720P	3	720P。
QUALITY_1080P	4	1080P。

NetworkQuality

网络质量。

枚举值	值	说明
UNKNOWN	0	未知网络。
EXCELLENT	1	极佳。
GOOD	2	良好。
POOR	3	较差。
BAD	4	差。
VERY_BAD	5	极差。

DOWN	6	中断。

MirrorType

摄像头镜像状态。

枚举值	值	说明
AUTO	0	自动，前置摄像头镜像，后置摄像头不镜像。
ENABLE	1	前后摄像头均镜像。
DISABLE	2	前后摄像头均不镜像。

NetworkType

网络类型。

枚举值	值	说明
UNKNOWN	0	未知或未识别的网络类型。
WIFI	1	WiFi 网络连接。
CELLULAR	2	蜂窝/移动数据网络连接。

DeviceFocusOwner

设备焦点。

枚举值	说明
call	语音通话场景。
live	直播场景。
room	房间场景。
none	未设置。

NetworkInfo

网络信息。

属性	类型	说明

userID	String	用户唯一 ID。
quality	NetworkQuality	网络质量。
upLoss	Int	上行丢包率，取值范围 0, 100。
downLoss	Int	下行丢包率，取值范围 0, 100。
delay	Int	延迟（单位：毫秒）。

DeviceState

设备状态。

属性	类型	说明
microphoneStatus	StateFlow<DeviceStatus>	麦克风状态。
microphoneLastError	StateFlow<DeviceError>	麦克风错误，用于出现报错时提取错误信息。
captureVolume	StateFlow<Int>	采集音量，取值范围 0, 100。
currentMicVolume	StateFlow<Int>	当前用户实际输出音量。
outputVolume	StateFlow<Int>	最大输出音量，取值范围 0, 100。
cameraStatus	StateFlow<DeviceStatus>	摄像头状态。
cameraLastError	StateFlow<DeviceError>	摄像头错误，用于出现报错时提取错误信息。
isFrontCamera	StateFlow<Boolean>	是否为前置摄像头。
localMirrorType	StateFlow<MirrorType>	镜像状态。
localVideoQuality	StateFlow<VideoQuality>	本地视频质量。
currentAudioRoute	StateFlow<AudioRoute>	当前音频路由位置。
screenStatus	StateFlow<DeviceStatus>	屏幕分享状态。
networkInfo	StateFlow<NetworkInfo>	网络信息。
networkType	StateFlow<NetworkType>	当前网络类型。

使用示例

```
// 获取单例实例
val store = DeviceStore.shared()
// 订阅状态变化
lifecycleScope.launch {
    store.deviceState.microphoneStatus.collect { status ->
        println("麦克风状态：$status")
    }
}
lifecycleScope.launch {
    store.deviceState.cameraStatus.collect { status ->
        println("摄像头状态：$status")
    }
}
// 打开麦克风
store.openLocalMicrophone { code, message ->
    if (code == 0) {
        println("麦克风打开成功")
    }
}
// 打开前置摄像头
store.openLocalCamera(isFront = true) { code, message ->
    if (code == 0) {
        println("摄像头打开成功")
    }
}
```

GiftStore

最近更新时间：2026-04-20 17:34:06

简介

`GiftStore` 提供了一套完整的礼物管理 API，包括发送礼物、刷新礼物列表、设置语言和监听礼物事件。 通过该类，可以在直播间内实现礼物互动功能。

> **重要：**
> 使用 GiftStore.create 工厂方法创建 `GiftStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 礼物状态更新通过 giftState 发布者传递。订阅它以接收房间内礼物数据的实时更新。

功能特性

- **礼物发送**：支持向当前房间发送指定礼物。
- **礼物列表**：获取和刷新当前房间可用的礼物列表。
- **语言设置**：设置礼物信息的展示语言。
- **事件监听**：监听礼物接收事件。

可订阅数据

GiftState 的字段描述如下：

属性名	类型	描述
usableGifts	StateFlow<List< GiftCategory >>	当前房间可用的所有礼物分类及礼物列表。

API 列表

函数名	描述
GiftStore.create	创建礼物管理实例。
addGiftListener	礼物事件回调。
removeGiftListener	礼物事件回调。

sendGift	发送礼物。
refreshUsableGifts	刷新可用礼物列表。
setLanguage	设置展示语言。

创建实例

GiftStore.create

创建礼物管理实例。

观察状态和事件

addGiftListener

礼物事件回调

removeGiftListener

礼物事件回调

礼物操作

sendGift

向当前房间发送指定礼物

```
abstract fun sendGift(
        giftID: String,
        count: Int,
        completion: CompletionHandler?
    )
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
giftID	String	必填	要发送的礼物唯一标识 ID。
count	Int	必填	单次发送的礼物数量。

completion	Completion Handler?	必填	完成回调（成功/失败状态）。

refreshUsableGifts

手动刷新当前房间的可用礼物列表。

```
abstract fun refreshUsableGifts(completion: CompletionHandler?)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
completion	Completion Handler?	必填	完成回调（成功时可通过 state 获取最新礼物列表，失败时返回错误信息）。

setLanguage

设置礼物信息的展示语言。

```
abstract fun setLanguage(language: String)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
language	String	必填	语言代码（"zh-CN" 表示中文，"en" 表示英文），设置完成展示界面刷新后礼物名称、描述等会同步更新为对应语言。

数据结构

Gift

礼物数据模型，包含单个礼物的完整属性信息。

属性	类型	说明
giftID	String	礼物 ID。

name	String	礼物名称。
desc	String	礼物描述。
iconURL	String	礼物图标图片的网络 URL，用于加载礼物缩略图。
resourceURL	String	礼物动效资源文件的网络 URL，用于加载礼物展示动效。
level	Long	礼物等级，用于区分礼物稀有度或价值层级。
coins	Long	礼物价格（金币）。
extensionInfo	Map<String, String>	礼物扩展信息，可自定义字段（如特效类型、赠送限制等）。

GiftCategory

礼物分类。

属性	类型	说明
categoryID	String	分类唯一标识 ID，用于区分不同礼物分类。
name	String	分类展示名称，用于 UI 分类显示（如 "热门礼物"，"高级礼物"）。
desc	String	分类描述信息，用于说明该分类的特点。
extensionInfo	Map<String, String>	分类扩展信息，包含自定义字段（如排序权重、显示样式等）。
giftList	List< Gift >	当前分类下的所有礼物列表。

GiftState

礼物状态，管理当前房间的礼物数据状态，支持实时更新并可被订阅监听。

属性	类型	说明
usableGifts	StateFlow<List< GiftCategory >>	当前房间可用的所有礼物分类及礼物列表。

GiftListener

礼物事件，用于接收直播间/语音聊天房内的礼物动态。

方法

onReceiveGift: 收到新礼物消息的事件回调。当直播间/语音聊天房内有其他观众发送礼物时，会触发该事件并返回相关信息。

```
open fun onReceiveGift(liveID: String, gift: Gift, count: Int, sender:
LiveUserInfo) {}
```

参数名	类型	说明
liveID	String	直播间 ID。
gift	Gift	礼物信息。
count	Int	礼物数量。
sender	LiveUserInfo	礼物发送者信息。

LikeStore

最近更新时间：2026-04-20 17:34:06

简介

`LikeStore` 提供了一套完整的点赞管理 API，包括发送点赞、监听点赞事件和获取点赞状态。通过该类，可以在直播间内实现点赞互动功能。

> **重要：**
> 使用 LikeStore.create 工厂方法创建 `LikeStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 点赞状态更新通过 likeState 发布者传递。订阅它以接收房间内点赞数据的实时更新。

功能特性

- **点赞发送**：支持向当前房间发送点赞。
- **点赞状态**：获取当前房间的累计点赞数。
- **事件监听**：监听点赞接收事件。

数据结构

LikeState

点赞状态，用于展示和订阅直播间/语音聊天房的点赞信息。

属性	类型	说明
totalLikeCount	StateFlow<Long>	当前直播间/语音聊天房的累计总点赞数，支持实时更新并可被订阅监听。

LikeListener

点赞事件，用于接收直播间/语音聊天房内的点赞动态。
此监听器用于接收直播间/语音聊天房内的点赞动态。
方法
onReceiveLikesMessage: 收到新点赞消息的事件回调。当直播间/语音聊天房内有其他观众发送点赞时，会触发该事件并返回相关信息。

```
open fun onReceiveLikesMessage(liveID: String, totalLikesReceived: Long,
sender: LiveUserInfo) {}
```

参数名	类型	说明
liveID	String	直播间 ID。
totalLikesReceived	Long	本次收到的新点赞数。
sender	LiveUserInfo	点赞发送者信息。

LiveAudienceStore

最近更新时间：2026-04-20 17:34:06

简介

`LiveAudienceStore` 提供了一套完整的观众管理 API，包括获取观众列表、设置管理员、踢出用户、禁言等功能。 通过该类，可以在直播间内实现观众管理功能。

> **重要：**
> 使用 LiveAudienceStore.create 工厂方法创建 `LiveAudienceStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 观众状态更新通过 liveAudienceState 发布者传递。订阅它以接收房间内观众数据的实时更新。

功能特性

- **观众列表**：获取和管理当前房间的观众列表。
- **权限管理**：设置和撤销管理员权限。
- **用户管理**：踢出用户、禁言等操作。
- **事件监听**：监听房主、管理员、观众加入/离开等事件。

可订阅数据

LiveAudienceState 的字段描述如下：

属性名	类型	描述
audienceList	StateFlow<List<LiveUserInfo>>	观众列表。
audienceCount	StateFlow<Int>	观众数量。
messageBannedUserList	StateFlow<List<LiveUserInfo>>	消息被禁言的用户列表。

API 列表

函数名	描述

LiveAudienceStore.create	创建观众管理实例。
addLiveAudienceListener	观众事件回调。
removeLiveAudienceListener	观众事件回调。
fetchAudienceList	获取观众列表。
setAdministrator	设置管理员。
revokeAdministrator	撤销管理员。
kickUserOutOfRoom	踢出用户。
disableSendMessage	禁言/解禁用户。

创建实例

LiveAudienceStore.create

创建观众管理实例。

观察状态和事件

addLiveAudienceListener

添加观众事件监听器

```
abstract fun addLiveAudienceListener(listener: LiveAudienceListener)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	LiveAudienceListener	必填	监听器。

removeLiveAudienceListener

移除观众事件监听器

```
abstract fun removeLiveAudienceListener(listener: LiveAudienceListener)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	LiveAudienceListener	必填	监听器。

观众管理

fetchAudienceList

获取观众列表

```
abstract fun fetchAudienceList(completion: CompletionHandler?)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
completion	CompletionHandler?	必填	完成回调。

setAdministrator

设置管理员

```
abstract fun setAdministrator(userID: String?, completion:
CompletionHandler?)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

userID	String?	必填	要设置为管理员的用户 ID。
completion	Completion Handler?	必填	完成回调。

revokeAdministrator

撤销管理员

```
abstract fun revokeAdministrator(userID: String?, completion:
CompletionHandler?)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String?	必填	要撤销管理员权限的用户 ID。
completion	Completion Handler?	必填	完成回调。

kickUserOutOfRoom

将用户踢出房间

```
abstract fun kickUserOutOfRoom(userID: String?, completion:
CompletionHandler?)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String?	必填	要踢出的用户 ID。
completion	Completion Handler?	必填	完成回调。

disableSendMessage

禁用/解禁用户发送消息

```
abstract fun disableSendMessage(
    userID: String?,
    isDisable: Boolean,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String?	必填	目标用户 ID。
isDisable	Boolean	必填	true 表示禁用发送消息，false 表示解禁。
completion	CompletionHandler?	必填	完成回调。

数据结构

Role

用户角色。

枚举值	说明
OWNER	房主。
ADMIN	管理员。
GENERAL_USER	普通用户。

LiveUserInfo

直播用户信息。

属性	类型	说明
userID	String	用户唯一标识 ID。

userName	String	用户名称。
avatarURL	String	用户头像 URL。

LiveAudienceState

直播观众状态。

属性	类型	说明
audienceList	StateFlow<List<LiveUserInfo>>	观众列表。
audienceCount	StateFlow<Int>	观众数量。
messageBannedUserList	StateFlow<List<LiveUserInfo>>	消息被禁言的用户列表。

LiveAudienceListener

直播观众事件。
此监听器用于接收直播间内全角色（房主、管理员、观众）的动态事件。
方法
onOwnerJoined: 房主加入事件。

```
open fun onOwnerJoined(owner: LiveUserInfo) {}
```

参数名	类型	说明
owner	LiveUserInfo	加入的房主信息。

onOwnerLeft: 房主离开事件。

```
open fun onOwnerLeft(owner: LiveUserInfo) {}
```

参数名	类型	说明
owner	LiveUserInfo	离开的房主信息。

onAdminJoined: 管理员加入事件。

```
open fun onAdminJoined(admin: LiveUserInfo) {}
```

参数名	类型	说明
admin	LiveUserInfo	加入的管理员信息。

onAdminLeft: 管理员离开事件。

```
open fun onAdminLeft(admin: LiveUserInfo) {}
```

参数名	类型	说明
admin	LiveUserInfo	离开的管理员信息。

onAudienceJoined: 观众加入事件。

```
open fun onAudienceJoined(audience: LiveUserInfo) {}
```

参数名	类型	说明
audience	LiveUserInfo	加入的观众信息。

onAudienceLeft: 观众离开事件。

```
open fun onAudienceLeft(audience: LiveUserInfo) {}
```

参数名	类型	说明
audience	LiveUserInfo	离开的观众信息。

onAudienceMessageDisabled: 观众被禁止发言事件。

```
open fun onAudienceMessageDisabled(audience: LiveUserInfo, isDisable:
Boolean) {}
```

参数名	类型	说明
audience	LiveUserInfo	观众信息。
isDisable	Boolean	是否被禁止发言。

LiveCoreView

最近更新时间：2026-04-20 17:34:06

简介

`LiveCoreView` 提供了直播推流和播放的视图容器，支持多人连麦、PK 等功能。 通过该组件，可以实现直播间的视频渲染和交互。

> **重要：**
> 使用前需要先调用 **setLiveID** 设置直播间 ID。

功能特性

- **视频渲染**：提供直播推流和播放的视图容器。
- **连麦支持**：支持多人连麦功能。
- **PK 支持**：支持主播 PK 功能。
- **房间外预览**：支持在进入房间前预览直播流。

数据结构

CoreViewType

核心视图类型。

枚举值	说明
PLAY_VIEW	播放视图。
PUSH_VIEW	推流视图。

ViewLayer

视图层级。

枚举值	说明
FOREGROUND	前景层。
BACKGROUND	背景层。

VideoViewAdapter

视频视图适配器协议。

方法

createCoGuestView: 创建连麦视图。

```
fun createCoGuestView(seatInfo: SeatInfo?, viewLayer: ViewLayer?): View?
```

参数名	类型	说明
seatInfo	SeatInfo?	连麦用户的麦位信息。
viewLayer	ViewLayer?	视图层级，前景层或背景层。

createCoHostView: 创建跨房连麦视图。

```
fun createCoHostView(seatInfo: SeatInfo?, viewLayer: ViewLayer?): View?
```

参数名	类型	说明
seatInfo	SeatInfo?	跨房连麦用户的麦位信息。
viewLayer	ViewLayer?	视图层级，前景层或背景层。

createBattleView: 创建 PK 视图。

```
fun createBattleView(seatInfo: SeatInfo?): View?
```

参数名	类型	说明
seatInfo	SeatInfo?	PK 用户的麦位信息。

createBattleContainerView: 创建 PK 容器视图。

```
fun createBattleContainerView(): View?
```

LiveListStore

最近更新时间：2026-04-20 17:34:06

简介

`LiveListStore` 提供了一套完整的直播间管理 API，包括开播、加入直播、离开直播、结束直播等功能。 通过该类，可以实现直播间的生命周期管理。主播调用 startLive/endLive 来开播和解散房间，观众调用 joinLive/leaveLive 来加入和退出直播间。

> **重要：**
> 使用 **LiveListStore.shared** 单例对象获取 `LiveListStore` 实例。

> **说明：**
> 直播状态更新通过 **liveState** 发布者传递。订阅它以接收直播数据的实时更新。

功能特性

- **直播列表**：获取和管理直播间列表。
- **开播**：主播开播。如果是新的直播间 ID，将创建直播间并开播；如果是服务端已创建的直播间 ID，将加入该直播间并开播。
- **直播加入**：加入已存在的直播间。
- **直播管理**：更新直播信息、结束直播等操作。
- **事件监听**：监听直播结束、被踢出等事件。

可订阅数据

LiveListState 的字段描述如下：

属性名	类型	描述
liveList	StateFlow<List<LiveInfo>>	直播列表。
liveListCursor	StateFlow<String>	直播列表游标。
currentLive	StateFlow<LiveInfo>	当前直播信息。

API 列表

函数名	描述

LiveListStore.shared	单例对象。
addLiveListListener	直播列表事件回调。
removeLiveListListener	直播列表事件回调。
fetchLiveList	获取直播列表。
fetchLiveInfo	获取直播信息。
startLive	主播开播。如果是新的直播间 ID，将创建直播间并开播；如果是服务端已创建的直播间 ID，将加入该直播间并开播。
joinLive	加入直播（观众调用）。
leaveLive	离开直播（观众调用）。
endLive	结束直播（主播调用）。
updateLiveInfo	更新直播信息。
queryMetaData	查询元数据。
updateLiveMetaData	更新元数据。

获取实例

LiveListStore.shared

单例对象。

观察状态和事件

addLiveListListener

添加直播列表事件监听器

```
abstract fun addLiveListListener(listener: LiveListListener)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	LiveListListener	必填	监听器。

removeLiveListListener

移除直播列表事件监听器

```
abstract fun removeLiveListListener(listener: LiveListListener)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	LiveListListener	必填	监听器。

直播列表

fetchLiveList

获取直播列表

```
abstract fun fetchLiveList(
    cursor: String?,
    count: Int,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
cursor	String?	必填	游标。
count	Int	必填	数量。

completion	CompletionHandler?	必填	完成回调。

fetchLiveInfo

获取直播信息

```
abstract fun fetchLiveInfo(
    liveID: String,
    completion: LiveInfoCompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播间 ID。
completion	LiveInfoCompletionHandler?	必填	完成回调。

直播操作

startLive

开播

```
abstract fun startLive(
    liveInfo: LiveInfo,
    completion: LiveInfoCompletionHandler?
)
```

主播开播。如果是新的直播间 ID，将创建直播间并开播；如果是服务端已创建的直播间 ID，将加入该直播间并开播。

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

liveInfo	LiveInfo	必填	直播信息。
completion	LiveInfoCompletionHandler?	必填	完成回调。

joinLive

加入直播

```
abstract fun joinLive(
    liveID: String?,
    completion: LiveInfoCompletionHandler?
)
```

观众调用此接口加入已存在的直播间。

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveID	String?	必填	直播 ID。
completion	LiveInfoCompletionHandler?	必填	完成回调。

leaveLive

离开直播

```
abstract fun leaveLive(completion: CompletionHandler?)
```

观众调用此接口离开当前直播间。

若主播仅需退出房间但不解散直播间，也可调用此接口。

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

completion	CompletionHandler?	必填	完成回调。

endLive

结束直播（主播调用）

```
abstract fun endLive(completion: StopLiveCompletionHandler?)
```

主播调用此接口结束当前直播并解散房间。

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
completion	StopLiveCompletionHandler?	必填	完成回调。

updateLiveInfo

更新直播信息

```
abstract fun updateLiveInfo(
    liveInfo: LiveInfo,
    modifyFlagList: List<LiveInfo.ModifyFlag>,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveInfo	LiveInfo	必填	直播信息。
modifyFlag	List<LiveInfo.ModifyFlag>	必填	修改标志。

completion	CompletionHandler?	必填	完成回调。

元数据操作

queryMetaData

查询元数据

```
abstract fun queryMetaData(
    keys: List<String>,
    completion: MetaDataCompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
keys	List<String>	必填	键列表。
completion	MetaDataCompletionHandler?	必填	完成回调。

updateLiveMetaData

更新直播元数据

```
abstract fun updateLiveMetaData(
    metaData: HashMap<String, String>,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

metaData	HashMap<String, String>	必填	元数据。
completion	Completion Handler?	必填	完成回调。

数据结构

TakeSeatMode

上麦模式。

枚举值	说明
FREE	自由上麦。
APPLY	申请上麦。

SeatLayoutTemplate

麦位布局模板，用于简化创建直播间时的麦位配置。

枚举值	说明
VideoDynamicGrid9Seats	竖屏动态九宫格，适用于视频直播场景。
VideoDynamicFloat7Seats	竖屏动态1v6浮动布局，适用于视频直播场景。
VideoLeftFocus9Seats	竖屏左侧焦点九宫格布局，适用于视频直播场景。
VideoUniformGrid9Seats	竖屏均匀九宫格布局，适用于视频直播场景。
VideoFixedGrid9Seats	竖屏静态九宫格，适用于视频直播场景。
VideoFixedFloat7Seats	竖屏静态1v6浮动布局，适用于视频直播场景。
VideoLandscape4Seats	横屏4人麦位布局，适用于视频直播场景。
Karaoke	语音 KTV 布局，适用于 K 歌场景，可指定麦位数量。

AudioSalon	语音沙龙布局，适用于语聊场景，可指定麦位数量。

LiveEndedReason

直播结束原因。

枚举值	说明
ENDED_BY_HOST	主播主动结束。
ENDED_BY_SERVER	服务器结束。

LiveKickedOutReason

被踢出直播间原因。

枚举值	说明
BY_ADMIN	被管理员踢出。
BY_LOGGED_ON_OTHER_DEVICE	在其他设备登录。
BY_SERVER	被服务器踢出。
FOR_NETWORK_DISCONNECTED	网络断开连接。
FOR_JOIN_ROOM_STATUS_INVALID_DURING_OFFLINE	离线期间房间状态无效。
FOR_COUNT_OF_JOINED_ROOMS_EXCEED_LIMIT	加入房间数量超过限制。

LiveInfo

直播信息。

属性	类型	说明
liveID	String	直播 ID。

liveName	String	直播名称。
notice	String	直播公告。
isMessageDisable	Boolean	是否禁用消息。
isPublicVisible	Boolean	是否公开可见。
isSeatEnabled	Boolean	是否启用麦位。
keepOwnerOnSeat	Boolean	房主是否保持在麦位上。
maxSeatCount	Int	最大麦位数量。
seatMode	TakeSeatMode	上麦模式。
seatTemplate	SeatLayoutTemplate	麦位布局模板，用于简化麦位配置。
seatLayoutTemplateID	Int	麦位布局模板 ID。
coverURL	String	封面 URL。
backgroundURL	String	背景 URL。
categoryList	List<Int>	分类列表。
activityStatus	Int	活动状态。
liveOwner	LiveUserInfo	直播房主信息。
createTime	Long	创建时间。
totalViewerCount	Int	总观看人数。
isGiftEnabled	Boolean	是否启用礼物。
metaData	Map<String, String>	元数据。

LiveListState

直播列表状态。

属性	类型	说明
liveList	StateFlow<List< LiveInfo >>	直播列表。

liveListCursor	StateFlow<String>	直播列表游标。
currentLive	StateFlow<LiveInfo>	当前直播信息。

LiveListListener

直播列表事件。

方法

onLiveEnded: 直播结束事件。

```
open fun onLiveEnded(liveID: String, reason: LiveEndedReason, message:
String) {}
```

参数名	类型	说明
liveID	String	直播 ID。
reason	LiveEndedReason	结束原因。
message	String	消息。

onKickedOutOfLive: 被踢出直播间事件。

```
open fun onKickedOutOfLive(liveID: String, reason: LiveKickedOutReason,
message: String) {}
```

参数名	类型	说明
liveID	String	直播 ID。
reason	LiveKickedOutReason	被踢出原因。
message	String	消息。

LiveInfoCompletionHandler

直播信息完成回调接口。

方法

onSuccess: 成功回调。

```
fun onSuccess(liveInfo: LiveInfo)
```

参数名	类型	说明
liveInfo	LiveInfo	直播信息。

onFailure: 失败回调。

```
fun onFailure(code: Int, desc: String)
```

参数名	类型	说明
code	Int	错误码。
desc	String	错误描述。

StopLiveCompletionHandler

停止直播完成回调接口。
方法
onSuccess: 成功回调。

```
fun onSuccess(statisticsData: TUILiveListManager.LiveStatisticsData)
```

参数名	类型	说明
statisticsData	TUILiveListManager.LiveStatisticsData	直播统计数据。

onFailure: 失败回调。

```
fun onFailure(code: Int, desc: String)
```

参数名	类型	说明
code	Int	错误码。
desc	String	错误描述。

MetaDataCompletionHandler

元数据完成回调接口。
方法

onSuccess: 成功回调。

```
fun onSuccess(metaData: HashMap<String, String>)
```

参数名	类型	说明
metaData	HashMap<String, String>	元数据。

onFailure: 失败回调。

```
fun onFailure(code: Int, desc: String)
```

参数名	类型	说明
code	Int	错误码。
desc	String	错误描述。

LiveSeatStore

最近更新时间：2026-04-20 17:34:06

简介

`LiveSeatStore` 提供了一套完整的麦位管理 API，包括上麦、下麦、锁麦、解锁麦位、踢用户下麦、远程控制设备等功能。 通过该类，可以在直播间内实现麦位管理功能。

> **重要：**
> 使用 **LiveSeatStore.create** 工厂方法创建 `LiveSeatStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 麦位状态更新通过 **liveSeatState** 发布者传递。订阅它以接收房间内麦位数据的实时更新。

功能特性

- **麦位管理**：上麦、下麦、锁麦、解锁麦位等操作。
- **用户管理**：踢用户下麦、移动用户到指定麦位。
- **设备控制**：远程控制用户的摄像头和麦克风。
- **事件监听**：监听麦位相关事件。

可订阅数据

LiveSeatState 的字段描述如下：

属性名	类型	描述
seatList	StateFlow<List<SeatInfo>>	麦位列表。
canvas	StateFlow<LiveCanvas>	画布信息。
speakingUsers	StateFlow<MutableMap<String, Int>>	正在说话的用户。
avStatistics	StateFlow<List<AVStatistics>>	音视频相关统计信息。

API 列表

函数名	描述

LiveSeatStore.create	创建麦位管理实例。
addLiveSeatEventListener	麦位事件回调。
removeLiveSeatEventListener	麦位事件回调。
takeSeat	上麦。
leaveSeat	下麦。
lockSeat	锁麦。
unlockSeat	解锁麦位。
kickUserOutOfSeat	踢用户下麦。
moveUserToSeat	移动用户。
openRemoteCamera	开启远程摄像头。
closeRemoteCamera	关闭远程摄像头。
openRemoteMicrophone	开启远程麦克风。
closeRemoteMicrophone	关闭远程麦克风。

创建实例

LiveSeatStore.create

创建麦位管理实例。

观察状态和事件

addLiveSeatEventListener

添加麦位事件监听器

```
abstract fun addLiveSeatEventListener(
```

```
    listener: LiveSeatListener?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	LiveSeatListener?	必填	监听器。

removeLiveSeatEventListener

移除麦位事件监听器

```
abstract fun removeLiveSeatEventListener(
    listener: LiveSeatListener?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	LiveSeatListener?	必填	监听器。

麦位操作

takeSeat

上麦

```
abstract fun takeSeat(
    seatIndex: Int,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
seatIndex	Int	必填	麦位索引。
completion	Completion Handler?	必填	完成回调。

leaveSeat

下麦

```
abstract fun leaveSeat(completion: CompletionHandler?)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
completion	Completion Handler?	必填	完成回调。

lockSeat

锁定麦位

```
abstract fun lockSeat(
    seatIndex: Int,
    completion: CompletionHandler?
)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
seatIndex	Int	必填	麦位索引。
completion	Completion Handler?	必填	完成回调。

unlockSeat

解锁麦位

```
abstract fun unlockSeat(
    seatIndex: Int,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
seatIndex	Int	必填	麦位索引。
completion	CompletionHandler?	必填	完成回调。

用户管理

kickUserOutOfSeat

踢用户下麦

```
abstract fun kickUserOutOfSeat(
    userID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String?	必填	用户 ID。
completion	CompletionHandler?	必填	完成回调。

moveUserToSeat

移动用户到麦位

```
abstract fun moveUserToSeat(
    userID: String?,
    targetIndex: Int,
    policy: MoveSeatPolicy? = MoveSeatPolicy.ABORT_WHEN_OCCUPIED,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String?	必填	用户 ID。
targetIndex	Int	必填	目标麦位索引。
policy	MoveSeatPolicy?	必填	移动策略。
completion	CompletionHandler?	必填	完成回调。

远程设备控制

openRemoteCamera

开启远程摄像头

```
abstract fun openRemoteCamera(
    userID: String?,
    policy: DeviceControlPolicy,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

userID	String?	必填	用户 ID。
policy	DeviceControlPolicy	必填	设备控制策略。
completion	CompletionHandler?	必填	完成回调。

closeRemoteCamera

关闭远程摄像头

```
abstract fun closeRemoteCamera(
    userID: String?,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String?	必填	用户 ID。
completion	CompletionHandler?	必填	完成回调。

openRemoteMicrophone

开启远程麦克风

```
abstract fun openRemoteMicrophone(
    userID: String?,
    policy: DeviceControlPolicy,
    completion: CompletionHandler?
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String?	必填	用户 ID。
policy	DeviceControlPolicy	必填	设备控制策略。
completion	CompletionHandler?	必填	完成回调。

closeRemoteMicrophone

关闭远程麦克风

```
abstract fun closeRemoteMicrophone(
    userID: String?,
    completion: CompletionHandler?
)
internal val hasAudioStreamUserList: MutableSet<String> = mutableSetOf()
internal val hasVideoStreamUserList: MutableSet<String> = mutableSetOf()
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String?	必填	用户 ID。
completion	CompletionHandler?	必填	完成回调。

数据结构

MoveSeatPolicy

移动麦位策略。

枚举值	说明
ABORT_WHEN_OCCUPIED	被占用时中止。
FORCE_REPLACE	强制替换。

SWAP_POSITION	交换位置。

DeviceControlPolicy

设备控制策略。

枚举值	说明
UNLOCK_ONLY	仅解锁。

SuspendStatus

用户的挂起状态。

枚举值	说明
NONE	未挂起。
IN_BACKGROUND	用户进入后台挂起。
IN_CALLING	用户正在接听电话。

LiveSeatListener

麦位相关的回调事件。

方法

方法名	说明
onLocalCameraOpenedByAdmin	当本地摄像头被管理员开启时触发此回调。
onLocalCameraClosedByAdmin	当本地摄像头被管理员关闭时触发此回调。
onLocalMicrophoneOpenedByAdmin	当本地麦克风被管理员开启时触发此回调。
onLocalMicrophoneClosedByAdmin	当本地麦克风被管理员关闭时触发此回调。

SeatUserInfo

麦位用户信息。

属性	类型	说明

userID	String	用户 ID。
userName	String	用户名。
avatarURL	String	头像 URL。
role	Role	用户角色。
liveID	String	直播间 ID。
microphoneStatus	DeviceStatus	麦克风状态。
allowOpenMicrophone	Boolean	是否允许开启麦克风。
cameraStatus	DeviceStatus	摄像头状态。
allowOpenCamera	Boolean	是否允许开启摄像头。
userSuspendStatus	SuspendStatus	用户挂起状态。

RegionInfo

麦位视图坐标信息。

属性	类型	说明
x	Int	X 坐标。
y	Int	Y 坐标。
w	Int	宽度。
h	Int	高度。
zorder	Int	层级顺序。

AVStatistics

音视频相关统计信息。

属性	类型	说明
userID	String	用户 ID。
videoBitrate	Int	本地视频的码率。

videoWidth	Int	本地视频的宽度。
videoHeight	Int	本地视频的高度。
frameRate	Int	本地视频的帧率。
audioSampleRate	Int	音频的采样率。
audioBitrate	Int	音频码率。

SeatInfo

麦位信息。

属性	类型	说明
index	Int	麦位索引。
isLocked	Boolean	是否锁定。
userInfo	SeatUserInfo	用户信息。
region	RegionInfo	区域信息。

LiveCanvas

直播画布。

属性	类型	说明
w	Int	宽度。
h	Int	高度。
templateID	Int	模板 ID。

LiveSeatState

LiveSeatStore 对外提供的麦位状态数据。

属性	类型	说明
seatList	StateFlow<List< SeatInfo >>	麦位列表。
canvas	StateFlow< LiveCanvas >	画布信息。
speakingUsers	StateFlow<MutableMap<Str	正在说话的用户。

	ing, Int>>	
avStatistics	StateFlow<List<AVStatistics>>	音视频相关统计信息。

LoginStore

最近更新时间：2026-04-20 17:34:06

简介

`LoginStore` 提供了一套完整的登录管理 API，包括用户登录、登出、设置个人信息等功能。 通过该类，可以管理用户的登录状态和用户资料。

> **重要：**
> 使用 **LoginStore.shared** 单例对象访问 `LoginStore` 实例。

> **说明：**
> 登录状态更新通过 **loginState** 发布者传递。订阅它以接收登录状态的实时更新。

功能特性

- **用户登录**：支持使用 SDK 应用 ID、用户 ID 和用户签名进行登录。
- **用户登出**：支持用户登出操作。
- **个人信息设置**：支持设置用户昵称、头像、性别等个人资料。

可订阅数据

LoginState 的字段描述如下：

属性名	类型	描述
loginStatus	StateFlow<LoginStatus>	登录状态。
loginUserInfo	StateFlow<UserProfile?>	登录用户信息。

API 列表

函数名	描述
LoginStore.shared	单例对象。
addLoginListener	登录事件监听器。
removeLoginListener	登录事件监听器。

login	登录。
logout	登出。
setSelfInfo	设置个人信息。

获取实例

LoginStore.shared

单例对象。

观察事件

addLoginListener

添加登录监听器

```
abstract fun addLoginListener(listener: LoginListener)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	LoginListener	必填	登录监听器。

removeLoginListener

移除登录监听器

```
abstract fun removeLoginListener(listener: LoginListener)
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	LoginListener	必填	登录监听器。

登录操作

login

登录

```
abstract fun login(
    context: Context,
    sdkAppID: Int,
    userID: String,
    userSig: String,
    completion: CompletionHandler? = null
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
context	Context	必填	上下文。
sdkAppID	Int	必填	SDK 应用 ID。
userID	String	必填	用户 ID。
userSig	String	必填	用户签名。
completion	CompletionHandler?	必填	完成回调。

logout

登出

```
abstract fun logout(completion: CompletionHandler? = null)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

completion	CompletionHandler?	必填	完成回调。

setSelfInfo

设置个人信息

```
abstract fun setSelfInfo(
    userProfile: UserProfile,
    completion: CompletionHandler? = null
)
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userProfile	UserProfile	必填	用户资料。
completion	CompletionHandler?	必填	完成回调。

数据结构

LoginStatus

登录状态。

枚举值	说明
UNLOGIN	未登录。
LOGINED	已登录。

AllowType

好友验证方式。

枚举值	值	说明
ALLOW_ANY	0	允许任何人。
NEED_CONFIRM	1	需要验证。

DENY_ANY	2	拒绝任何人。

Gender

性别。

枚举值	值	说明
UNKNOWN	0	未知。
MALE	1	男性。
FEMALE	2	女性。

LoginListener

登录事件。

方法

方法名	说明
onKickedOffline	当前用户被踢下线。
onLoginExpired	登录票据过期。

UserProfile

用户资料。

属性	类型	说明
userID	String	用户 ID。
nickname	String?	昵称。
avatarURL	String?	头像 URL。
selfSignature	String?	个性签名。
gender	Gender?	性别。
role	Int?	角色。
level	Int?	等级。
birthday	Long?	生日。

allowType	AllowType?	好友验证方式。
customInfo	Map<String, ByteArray>?	自定义信息。

LoginState

登录状态。

属性	类型	说明
loginStatus	StateFlow<LoginStatus>	登录状态。
loginUserInfo	StateFlow<UserProfile?>	登录用户信息。

iOS
API 概述

最近更新时间：2026-04-20 17:34:06

AtomicXCore API 概览 (Swift)

AtomicXCore SDK 是最新推出的面向视频直播、语聊房等场景的全新一代基于响应式的 API。您可以非常快速的基于这组 API 构建自己的 UI 页面。它支持房间管理、屏幕分享、成员管理、麦位控制、基础美颜等丰富功能。本页面包含 AtomicXCore SDK 的所有的功能模块，并分类展示如下。

用户登录

类名	说明
LoginStore	登录相关接口，管理用户登录、登出、用户信息设置等操作。

设备管理

类名	说明
AudioEffectStore	音效设置相关接口，管理主播的变声、混响和耳返等音效功能。
BaseBeautyStore	基础美颜相关接口，管理磨皮、美白、红润等基础美颜效果的调节和状态同步。
DeviceStore	设备相关接口，操作麦克风、摄像头等。

直播互动

类名	说明
BattleStore	直播 PK 管理相关接口，管理 PK 的创建、加入、离开等操作。
CoGuestStore	直播连麦管理相关接口，管理主播与观众之间的连麦申请、邀请、接受、拒绝等操作。
CoHostStore	直播主播连线管理相关接口，管理主播间相互连线的创建、加入、离开等操作。
LikeStore	点赞相关接口，管理直播间/语音聊天房内的点赞发送、点赞状态同步及点赞事件监听等操作。
LiveAudienceStore	直播观众相关接口，管理观众列表、权限设置等操作。

LiveListStore	直播列表相关接口，管理直播房间的创建、加入、离开等操作。
LiveSeatStore	直播麦位管理相关接口，管理麦位的上下麦、锁麦、释放麦位等操作。

弹幕消息

类名	说明
BarrageStore	弹幕相关接口，管理直播间/语音聊天房内的弹幕发送、弹幕状态同步及弹幕事件监听等操作。

礼物消息

类名	说明
GiftStore	礼物相关接口，管理直播间/语音聊天房内的礼物发送、礼物状态同步及礼物事件监听等操作。

视图组件

类名	说明
LiveCoreView	直播核心视图组件，提供直播推流和播放的视图容器，支持多人连麦、PK 等功能。

Define

最近更新时间：2026-04-20 17:34:06

简介

此模块定义了跨平台通用的基础类型，为 API 调用提供统一的错误处理和回调机制。

功能特性

- **回调机制**：提供统一的完成回调闭包类型。

数据结构

CompletionClosure

提供基础类型定义，包括错误处理结构、回调闭包类型和响应式状态管理工具。

```
public typealias CompletionClosure = (Result<Void, ErrorInfo>) -> Void
```

ErrorInfo

错误信息结构。

属性	类型	说明
code	Int	错误码。
message	String	错误消息描述。

StatePublisher

状态发布者。

StatePropertySelector

状态属性选择器协议。

StatePublisherSelector

状态发布器选择器。

ListResultCompletionClosure

列表结果完成回调接口。
用于返回分页列表的异步操作结果回调接口。

AudioEffectStore

最近更新时间：2026-04-20 17:34:06

简介

`AudioEffectStore` 提供了一套完整的音效管理 API，包括变声效果、混响效果和耳返功能。 通过该类，主播可以在直播过程中实时调整自己的声音效果，提升直播体验。

> **重要：**
> 使用 **shared** 单例获取 `AudioEffectStore` 实例。设置的音效在退出房间后会自动失效，下次进房需要重新设置。

> **说明：**
> 音效状态更新通过 **state** 发布者传递。订阅它以接收有关变声、混响和耳返状态的实时更新。

> **警告：**
> 由于蓝牙耳机的硬件延迟非常高，在主播佩戴蓝牙耳机时无法开启耳返功能。请在用户界面上提示主播佩戴有线耳机。

功能特性

- **变声效果**：支持多种变声特效，如熊孩子、小女孩、大叔等。
- **混响效果**：支持多种混响特效，如 KTV、小房间、大会堂等。
- **耳返功能**：主播可在耳机中听到自己的声音，适用于唱歌场景。
- **音量控制**：支持耳返音量的精细调节。

可订阅数据

AudioEffectState 的字段描述如下：

属性名	类型	描述
audioChangerType	AudioChangerType	变声状态。
audioReverbType	AudioReverbType	混响状态。
isEarMonitorOpened	Bool	耳返开启。

earMonitorVolume	Int	耳返音量，取值范围0 – 100。 如果将 volume 设置成100之后感觉音量还是太小，可以将 volume 最大设置成150，但超过100的 volume 会有爆音的风险，请谨慎操作。

API 列表

函数名	描述
shared	获取单例实例。
setAudioChangerType	设置变声效果。
setAudioReverbType	设置混响效果。
setVoiceEarMonitorEnable	开启/关闭耳返。
setVoiceEarMonitorVolume	设置耳返音量。
reset	重置为默认状态。

获取实例

shared

获取单例实例。

变声设置

setAudioChangerType

设置变声效果

```
public func setAudioChangerType(type: AudioChangerType) {
    fatalError("\(#function) must be overridden by subclass")
}
```

通过该接口您可以设置人声的变声特效。

变声特效可以作用于人声之上，通过声学算法对人声进行二次处理，以获得与原始声音所不同的音色。

版本信息
从3.5版本开始支持。
调用时机
进入房间后，且需要使用变声效果时调用。
注意事项

> **说明：**
> 设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

参数说明

参数名	类型	是否必填	描述
type	AudioChangerType	必填	变声效果类型。

混响设置

setAudioReverbType

设置混响效果

```
public func setAudioReverbType(type: AudioReverbType) {
    fatalError("\(#function) must be overridden by subclass")
}
```

通过该接口您可以设置人声的混响效果。

混响特效可以作用于人声之上，通过声学算法对声音进行叠加处理，模拟出各种不同环境下的临场感受。
版本信息
从3.5版本开始支持。
调用时机
进入房间后，且需要使用混响效果时调用。
注意事项

> **说明：**
> 设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

参数说明

参数名	类型	是否必填	描述

type	AudioReverbType	必填	混响效果类型。

耳返设置

setVoiceEarMonitorEnable

开启/关闭耳返

```
public func setVoiceEarMonitorEnable(enable: Bool) {
    fatalError("\(#function) must be overridden by subclass")
}
```

主播开启耳返后，可以在耳机里听到麦克风采集到的自己发出的声音，该特效适用于主播唱歌的应用场景中。

版本信息

从3.5版本开始支持。

适用场景

适用于主播唱歌场景，让主播能够实时听到自己的声音以便调整演唱效果。

调用时机

进入房间后，且主播佩戴有线耳机时调用。

注意事项

> **警告：**
> 由于蓝牙耳机的硬件延迟非常高，所以在主播佩戴蓝牙耳机时无法开启此特效，请尽量在用户界面上提示主播佩戴有线耳机。

参数说明

参数名	类型	是否必填	描述
enable	Bool	必填	是否开启耳返。

setVoiceEarMonitorVolume

设置耳返音量

```
public func setVoiceEarMonitorVolume(volume: Int) {
    fatalError("\(#function) must be overridden by subclass")
}
```

通过该接口您可以设置耳返特效中声音的音量大小。

版本信息
从3.5版本开始支持。
调用时机
开启耳返功能后调用。
注意事项

> **说明：**
> 如果将 volume 设置成100之后感觉音量还是太小，可以将 volume 最大设置成150，但超过100的 volume 会有爆音的风险，请谨慎操作。

参数说明

参数名	类型	是否必填	描述
volume	Int	必填	耳返音量。（取值范围：0 – 100（超过100可能导致爆音））（默认值：100）。

重置

reset

重置为默认状态

```
public func reset() {
    fatalError("\(#function) must be overridden by subclass")
}
```

将所有音效设置重置为默认值，包括关闭变声效果、关闭混响效果、关闭耳返并重置耳返音量。
版本信息
从3.5版本开始支持。
调用时机
需要恢复默认音效设置时调用，例如退出直播间前。

数据结构

AudioChangerType

变声效果类型。

枚举值	值	说明

none	0	关闭特效。
child	1	熊孩子。
littleGirl	2	小女孩。
man	3	大叔。
heavyMetal	4	重金属。
cold	5	感冒。
foreigner	6	外语腔。
trappedBeast	7	困兽。
fatso	8	肥宅。
strongCurrent	9	强电流。
heavyMachinery	10	重机械。
ethereal	11	空灵。

AudioReverbType

混响效果类型。

枚举值	值	说明
none	0	关闭特效。
ktv	1	KTV。
smallRoom	2	小房间。
auditorium	3	大会堂。
deep	4	低沉。
loud	5	洪亮。
metallic	6	金属声。
magnetic	7	磁性。

AudioEffectState

AudioEffectStore 对外提供的音效相关状态数据。

属性	类型	说明
audioChangerType	AudioChangerType	变声状态。
audioReverbType	AudioReverbType	混响状态。
isEarMonitorOpened	Bool	耳返开启。
earMonitorVolume	Int	耳返音量，取值范围0 – 100。 如果将 volume 设置成100之后感觉音量还是太小，可以将 volume 最大设置成150，但超过100的 volume 会有爆音的风险，请谨慎操作。

使用示例

```
// 获取单例实例
let store = AudioEffectStore.shared
// 订阅状态变化
store.state.subscribe { state in
    print("当前变声效果：\(state.audioChangerType)")
    print("当前混响效果：\(state.audioReverbType)")
    print("耳返开启：\(state.isEarMonitorOpened)")
}
// 设置变声效果
store.setAudioChangerType(type: .littleGirl)
// 设置混响效果
store.setAudioReverbType(type: .ktv)
// 开启耳返
store.setVoiceEarMonitorEnable(enable: true)
store.setVoiceEarMonitorVolume(volume: 80)
```

BarrageStore

最近更新时间：2026-04-20 17:34:06

简介

`BarrageStore` 提供了一套完整的弹幕管理 API，包括发送文本弹幕、发送自定义弹幕和添加本地提示消息。 通过该类，可以在直播间内实现弹幕互动功能。

> **重要：**
> 使用 **create(liveID:)** 工厂方法创建 `BarrageStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 弹幕状态更新通过 **state** 发布者传递。订阅它以接收房间内弹幕数据的实时更新。

功能特性

- **文本弹幕**：支持发送纯文本弹幕消息。
- **自定义弹幕**：支持发送自定义格式的弹幕（如带特效的弹幕）。
- **本地提示**：支持添加仅本地可见的提示消息。

可订阅数据

BarrageState 的字段描述如下：

属性名	类型	描述
messageList	[Barrage]	当前房间的弹幕消息列表，支持实时更新并可被订阅监听。

API 列表

函数名	描述
create	创建弹幕管理实例。
CustomMessageEventPublisher	自定义消息事件发布者。
sendTextMessage	发送文本弹幕。
sendCustomMess	发送自定义弹幕。

age	
appendLocalTip	添加本地提示消息。

创建实例

create

弹幕管理核心类，用于处理直播间/语音聊天房内的弹幕相关业务逻辑。

```
public static func create(liveID: String) -> BarrageStore {
    let store: BarrageStoreImpl = StoreFactory.shared.getStore(liveId:
liveID)
    return store
}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播间 ID。

观察事件

CustomMessageEventPublisher

自定义消息事件发布者。

发送弹幕

sendTextMessage

发送文本类型弹幕。

```
public func sendTextMessage(text: String,
                            extensionInfo: [String: String]?,
                            completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
text	String	必填	文本弹幕内容。
extensionInfo	[String: String]?	必填	扩展信息，可包含自定义字段（如指定弹幕颜色、字体大小等）。
completion	Completion Closure?	必填	完成回调（成功/失败状态）。

sendCustomMessage

发送自定义类型弹幕。

```
public func sendCustomMessage(businessID: String,
                              data: String,
                              completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
businessID	String	必填	业务标识 ID，用于区分不同业务场景的自定义弹幕。
data	String	必填	自定义数据内容，通常为 JSON 格式字符串，用于传递业务自定义的数据。
completion	Completion Closure?	必填	完成回调（成功/失败状态）。

本地消息

appendLocalTip

添加本地提示消息（在本地添加提示或操作反馈消息，仅当前客户端可见）。

```
public func appendLocalTip(message: Barrage) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

注意事项

> **说明：**
> 该消息仅在本地显示，不会通过网络发送给其他用户。

参数说明

参数名	类型	是否必填	描述
message	Barrage	必填	本地弹幕消息（如系统提示、操作反馈等，仅当前用户可见）。

数据结构

BarrageType

弹幕类型枚举，用于区分不同的弹幕消息种类。

枚举值	值	说明
text	0	文本类型弹幕，包含纯文字内容。
custom	1	自定义类型弹幕，支持业务自定义数据格式（如带特效的弹幕、互动消息等）。

Barrage

弹幕数据模型，包含单条弹幕的完整属性信息。

属性	类型	说明
liveID	String	弹幕所属直播间/语音聊天房的唯一标识 ID。
sender	LiveUserInfo	弹幕发送者的用户信息（如用户 ID、昵称、头像等）。

sequence	Int	弹幕消息的唯一序列 ID，用于消息排序和去重。
timestampInSecond	TimeInterval	弹幕发送时间戳（单位：秒），用于展示发送时间顺序。
messageType	BarrageType	弹幕消息类型（文本或自定义）。
textContent	String	文本类型弹幕的消息内容，即弹幕的文本内容。
extensionInfo	[String: String]?	弹幕扩展信息，可自定义字段（如显示样式、优先级等）。当 messageType 为 TEXT 时有效。
businessID	String	自定义类型弹幕的业务标识 ID，用于区分不同业务场景的自定义弹幕。
data	String	自定义类型弹幕的具体数据内容（通常为 JSON 格式字符串），当 messageType 为 CUSTOM 时有效。

BarrageState

弹幕状态，管理当前房间的弹幕数据状态。

属性	类型	说明
messageList	[Barrage]	当前房间的弹幕消息列表，支持实时更新并可被订阅监听。

BaseBeautyStore

最近更新时间：2026-04-20 17:34:06

简介

基础美颜功能通过简单易用的 API 实现实时美颜效果调节。`BaseBeautyStore` 提供了一套完整的接口来管理美颜效果的设置和状态订阅。

> **说明：**
> 美颜状态更新通过 **state** 发布者传递。订阅它以接收有关美颜效果级别的实时更新。

功能特性

- **磨皮效果**：支持0 – 9级别的磨皮效果调节。
- **美白效果**：支持0 – 9级别的美白效果调节。
- **红润效果**：支持0 – 9级别的红润效果调节。
- **状态订阅**：实时订阅美颜状态变化，同步 UI 显示与实际效果。

可订阅数据

BaseBeautyState 的字段描述如下：

属性名	类型	描述
smoothLevel	Float	磨皮级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。
whitenessLevel	Float	美白级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。
ruddyLevel	Float	红润级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。

API 列表

函数名	描述
shared	获取单例实例。
setSmoothLevel	设置磨皮级别。
setWhitenessLeve	设置美白级别。

setRuddyLevel	设置红润级别。
reset	重置为默认状态。

获取实例

shared

获取单例实例。

美颜调节

setSmoothLevel

设置磨皮级别

```
public func setSmoothLevel(smoothLevel: Float) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
smoothLevel	Float	必填	磨皮级别，取值范围 0, 9；0表示关闭，9表示效果最明显。

setWhitenessLevel

设置美白级别

```
public func setWhitenessLevel(whitenessLevel: Float) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
whitenessLevel	Float	必填	美白级别，取值范围 0, 9；0表示关闭，9表示效果最明显。

setRuddyLevel

设置红润级别

```
public func setRuddyLevel(ruddyLevel: Float) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
ruddyLevel	Float	必填	红润级别，取值范围 0, 9；0表示关闭，9表示效果最明显。

reset

将所有美颜参数（磨皮、美白、红润）重置为默认关闭状态（值为0）。

```
public func reset() {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息
从3.5版本开始支持。

数据结构

BaseBeautyState

基础美颜状态，管理磨皮、美白、红润等美颜效果的级别数据。支持订阅以同步 UI 显示与实际效果。

属性	类型	说明
smoothLevel	Float	磨皮级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。

whitenessLevel	Float	美白级别，取值范围 0 - 9；0表示关闭，9表示效果最明显。
ruddyLevel	Float	红润级别，取值范围 0 - 9；0表示关闭，9表示效果最明显。

使用示例

```
// 获取单例实例
let store = BaseBeautyStore.shared
// 订阅状态变化
store.state.subscribe { state in
    print("磨皮级别: \(state.smoothLevel)")
    print("美白级别: \(state.whitenessLevel)")
    print("红润级别: \(state.ruddyLevel)")
}
// 设置美颜效果
store.setSmoothLevel(smoothLevel: 5)
store.setWhitenessLevel(whitenessLevel: 3)
store.setRuddyLevel(ruddyLevel: 2)
// 重置所有美颜效果
store.reset()
```

BattleStore

最近更新时间：2026-04-20 17:34:06

简介

PK 功能实现主播之间的实时互动对战。`BattleStore` 提供了一套全面的 API 来管理整个 PK 生命周期。

> **重要：**
> 请始终使用工厂方法 **create(liveID:)** 并提供有效的直播间 ID 来创建 `BattleStore` 实例。不要尝试直接初始化。

> **说明：**
> PK 状态更新通过 **state** 发布者传递。订阅它以接收有关 PK 信息、参与用户和分数的实时更新。

> **警告：**
> 如果 PK 请求在指定的超时时间内未收到响应，将触发超时事件。请始终在 UI 中处理超时场景。

功能特性

- **PK 请求管理**：主播可以发起 PK 请求，被邀请方可以接受或拒绝。
- **状态管理**：实时跟踪 PK 信息、参与用户和分数。
- **事件驱动架构**：提供完整的 PK 事件回调。
- **超时处理**：为 PK 请求提供内置超时机制。

可订阅数据

BattleState 的字段描述如下：

属性名	类型	描述
currentBattleInfo	BattleInfo?	当前 PK 信息。
battleUsers	[SeatUserInfo]	PK 用户列表。
battleScore	[String: UInt]	PK 分数的映射。

API 列表

函数名	描述

create	创建 BattleStore 实例。
battleEventPublisher	PK 事件发布者。
requestBattle	发起 PK 请求。
cancelBattleRequest	取消 PK 请求。
acceptBattle	接受 PK 请求。
rejectBattle	拒绝 PK 请求。
exitBattle	退出 PK。

创建实例

create

创建 BattleStore 实例

```
public static func create(liveID: String) -> BattleStore {
    let store: BattleStoreImpl = StoreFactory.shared.getStore(liveId:
liveID)
    return store
}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播间 ID。

观察状态和事件

battleEventPublisher

PK 事件发布者

PK 操作

requestBattle

发起 PK 请求

```
public func requestBattle(config: BattleConfig, userIDList: [String],
timeout: TimeInterval, completion: BattleRequestClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
config	BattleConfig	必填	PK 配置。
userIDList	[String]	必填	需要进入 PK 的用户 ID 列表。
timeout	TimeInterval	必填	请求超时时间。
completion	BattleRequestClosure?	必填	发起请求成功的回调。

cancelBattleRequest

取消 PK 请求

```
public func cancelBattleRequest(battleId: String, userIdList: [String],
completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
battleId	String	必填	PK ID。
userIdList	[String]	必填	用户 ID 列表。

completion	Completion Closure?	必填	取消请求成功的回调。

acceptBattle

接受 PK 请求

```
public func acceptBattle(battleID: String, completion:
CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
battleID	String	必填	PK ID。
completion	Completion Closure?	必填	接受成功的回调。

rejectBattle

拒绝 PK 请求

```
public func rejectBattle(battleID: String, completion:
CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
battleID	String	必填	PK ID。
completion	Completion Closure?	必填	拒绝成功的回调。

exitBattle

退出 PK

```
public func exitBattle(battleID: String, completion: CompletionClosure?)
{
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
battleID	String	必填	PK ID。
completion	CompletionClosure?	必填	退出 PK 成功的回调。

数据结构

BattleEndedReason

正在 PK 中用户收到 PK 结束的原因。

枚举值	值	说明
timeOver	0	PK 倒计时结束。
allMemberExit	1	所有 PK 成员退出。

BattleEvent

收到 PK 相关的回调事件。

枚举值	说明
onBattleStarted	当 PK 正式开始时触发此回调，通知所有参与者 PK 已经开始。
onBattleEnded	当 PK 结束时触发此回调。
onUserJoinBattle	当有用户加入 PK 时触发此回调。
onUserExitBattle	当有用户退出 PK 时触发此回调。

onBattleRequestReceived	当收到 PK 请求时触发此回调。
onBattleRequestCancelled	当 PK 请求被取消时触发此回调。
onBattleRequestTimeout	当 PK 请求超时时触发此回调。
onBattleRequestAccept	当 PK 请求被接受时触发此回调。
onBattleRequestReject	当 PK 请求被拒绝时触发此回调。

BattleConfig

发送 PK 请求时，设置的 PK 配置信息。

属性	类型	说明
duration	TimeInterval	PK 持续时间（单位：秒）。
needResponse	Bool	被邀请用户是否需要回复同意/拒绝。
extensionInfo	String	扩展信息。

BattleInfo

PK 信息。

属性	类型	说明
battleID	String	PK ID。
config	BattleConfig	发送 PK 请求时，设置的 PK 配置信息。
startTime	UInt	PK 开始标记时间戳（单位：秒）。
endTime	UInt	PK 结束标记时间戳（单位：秒）。

BattleState

BattleStore 对外提供的 PK 相关状态数据。

属性	类型	说明

currentBattleInfo	BattleInfo?	当前 PK 信息。
battleUsers	[SeatUserInfo]	PK 用户列表。
battleScore	[String: UInt]	PK 分数的映射。

BattleRequestCallback

PK 请求回调。

PK 请求的回调接口，用于处理 PK 请求的成功或失败结果。

方法

onSuccess: 成功回调。

```
fun onSuccess(battleInfo: BattleInfo, resultMap: Map<String, Int>)
```

参数名	类型	说明
battleInfo	BattleInfo	PK 信息，包含 PK 的详细配置和状态。
resultMap	Map<String, Int>	PK 请求的响应结果回调。

onError: 失败回调。

```
fun onError(code: Int, desc: String)
```

参数名	类型	说明
code	Int	错误码。
desc	String	错误描述。

使用示例

```
// 创建 store 实例
let store = BattleStore.create(liveID: "live_room_123")
// 订阅状态变化
store.state.subscribe { state in
    if let battleInfo = state.currentBattleInfo {
        print("当前 PK ID: \(battleInfo.battleID)")
    }
    print("PK 用户数: \(state.battleUsers.count)")
```

```
}
// 订阅 PK 事件
store.battleEventPublisher.sink { event in
    switch event {
    case .onBattleStarted(let battleInfo, let inviter, let invitees):
        print("PK 开始，发起者: \(inviter.userName)")
    case .onBattleEnded(let battleInfo, let reason):
        print("PK 结束，原因: \(reason)")
    default:
        break
    }
}
// 发起 PK 请求
let config = BattleConfig(duration: 300, needResponse: true)
store.requestBattle(config: config, userIDList: ["user_456"], timeout:
30) { result in
    switch result {
    case .success(let (battleInfo, resultMap)):
        print("PK 请求成功: \(battleInfo.battleID)")
    case .failure(let error):
        print("PK 请求失败: \(error)")
    }
}
```

CoGuestStore

最近更新时间：2026-04-20 17:34:06

简介

连麦功能通过基于麦位的系统实现主播和观众成员之间的实时互动。`CoGuestStore` 提供了一套全面的 API 来管理整个连麦生命周期。

> **重要：**
> 请始终使用工厂方法 **create(liveID:)** 并提供有效的直播间 ID 来创建 `CoGuestStore` 实例。不要尝试直接初始化。

> **说明：**
> 连麦状态更新通过 **state** 发布者传递。订阅它以接收有关已连接用户、邀请和申请的实时更新。

> **警告：**
> 如果连麦请求在指定的超时时间内未收到响应，将触发带有 `NoResponseReason.timeout` 的事件。请始终在 UI 中处理超时场景。

功能特性

- **双向邀请**：主播可以邀请观众成员，观众成员也可以申请加入。
- **状态管理**：实时跟踪已连接用户、邀请和申请。
- **事件驱动架构**：为主播和观众角色提供独立的事件流。
- **超时处理**：为邀请和申请提供内置超时机制。

可订阅数据

CoGuestState 的字段描述如下：

属性名	类型	描述
connected	[SeatUserInfo]	已经在麦位上的用户列表。
invitees	[LiveUserInfo]	主播发出邀请的用户列表。
applicants	[LiveUserInfo]	主播收到申请连麦的用户列表。
candidates	[LiveUserInfo]	候选连麦的用户列表。

API 列表

函数名	描述
create	创建对象实例。
hostEventPublisher	主播端事件发布者。
guestEventPublisher	观众端事件发布者。
applyForSeat	观众申请连麦。
cancelApplication	观众取消申请。
acceptApplication	主播接受申请。
rejectApplication	主播拒绝申请。
inviteToSeat	主播邀请观众连麦。
cancelInvitation	主播取消邀请。
acceptInvitation	观众接受邀请。
rejectInvitation	观众拒绝邀请。
disConnect	结束连麦会话。

创建实例

create

创建 CoGuestStore 实例

```
public static func create(liveID: String) -> CoGuestStore {
    let store: CoGuestStoreImpl = StoreFactory.shared.getStore(liveId:
liveID)
    return store
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播间 ID。

观察状态和事件

hostEventPublisher

主播端事件发布者

guestEventPublisher

观众端事件发布者

观众操作

applyForSeat

申请上麦

```
public func applyForSeat(seatIndex: Int = -1, timeout: TimeInterval,
extraInfo: String?, completion: CompletionClosure?) { fatalError("\
(#function) must be overridden by subclass") }
```

以观众身份请求加入连麦会话。

调用此方法后，会向直播间内的所有主播发送连麦请求。请求将保持活动状态，直到：
- 主播通过 **acceptApplication** 接受
- 主播通过 **rejectApplication** 拒绝
- 超时时间到期
- 您通过 **cancelApplication** 取消

版本信息
从3.5版本开始支持。
注意事项

> **说明：**
> 如果在超时时间内没有主播响应，将触发带有 **NoResponseReason/timeout** 的 {ref2} 事件。

参数说明

参数名	类型	是否必填	描述
seatIndex	Int	必填	麦位索引，-1表示自动分配麦位。

timeout	TimeInterval	必填	超时时间（单位：秒）。
extraInfo	String?	必填	额外信息。
completion	CompletionClosure?	必填	完成回调。

cancelApplication

取消上麦申请

```
public func cancelApplication(completion: CompletionClosure?) {
fatalError("\(#function) must be overridden by subclass") }
```

取消之前发送的连麦申请。调用此方法后，所有主播将收到申请取消的通知。

版本信息

从3.5版本开始支持。

注意事项

> **说明：**
> 如果申请已被主播处理，取消操作可能无效。

参数说明

参数名	类型	是否必填	描述
completion	CompletionClosure?	必填	完成回调。

acceptApplication

接受上麦申请

```
public func acceptApplication(userID: String, completion:
CompletionClosure?) { fatalError("\(#function) must be overridden by
subclass") }
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。
completion	Completion Closure?	必填	完成回调。

rejectApplication

拒绝上麦申请

```
public func rejectApplication(userID: String, completion:
CompletionClosure?) { fatalError("\(#function) must be overridden by
subclass") }
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。
completion	Completion Closure?	必填	完成回调。

主播操作

inviteToSeat

邀请观众上麦

```
public func inviteToSeat(userID: String, seatIndex: Int = -1, timeout:
TimeInterval, extraInfo: String?, completion: CompletionClosure?) {
fatalError("\(#function) must be overridden by subclass") }
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
userID	String	必填	被邀请用户 ID。

seatIndex	Int	必填	麦位索引，-1表示自动分配麦位。
timeout	TimeInterval	必填	超时时间（单位：秒）。
extraInfo	String?	必填	额外信息。
completion	CompletionClosure?	必填	完成回调。

cancelInvitation

取消上麦邀请

```
public func cancelInvitation(inviteeID: String, completion:
CompletionClosure?) { fatalError("\(#function) must be overridden by
subclass") }
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
inviteeID	String	必填	被邀请用户 ID。
completion	CompletionClosure?	必填	完成回调。

acceptInvitation

接受上麦邀请

```
public func acceptInvitation(inviterID: String, completion:
CompletionClosure?) { fatalError("\(#function) must be overridden by
subclass") }
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述

inviterID	String	必填	邀请者用户 ID。
completion	Completion Closure?	必填	完成回调。

rejectInvitation

拒绝上麦邀请

```
public func rejectInvitation(inviterID: String, completion:
CompletionClosure?) { fatalError("\(#function) must be overridden by
subclass") }
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
inviterID	String	必填	邀请者用户 ID。
completion	Completion Closure?	必填	完成回调。

连接控制

disConnect

断开连麦

```
public func disConnect(completion: CompletionClosure?) { fatalError("\
(#function) must be overridden by subclass") }
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
completion	Completion Closure?	必填	完成回调。

数据结构

NoResponseReason

主播发出的连麦邀请或者观众发起连麦请求的无响应原因。

枚举值	值	说明
timeout	0	请求超时。
alreadySeated	1	用户已在麦位上。

HostEvent

主播侧收到的回调事件。

枚举值	说明
onGuestApplicationReceived	当观众申请连麦时触发此回调。
onGuestApplicationCancelled	当观众取消连麦申请时触发此回调。
onGuestApplicationProcessedByOtherHost	当观众的连麦申请被其他主播处理时触发此回调。
onHostInvitationResponded	当主播发出的连麦邀请收到观众响应时触发此回调。
onHostInvitationNoResponse	当主播发出的连麦邀请无响应时触发此回调。

GuestEvent

观众侧收到的回调事件。

枚举值	说明
onHostInvitationReceived	当收到主播的连麦邀请时触发此回调。
onHostInvitationCancelled	当主播取消连麦邀请时触发此回调。
onGuestApplicationResponded	当观众的连麦申请收到主播响应时触发此回调。

onGuestApplicationNoResponse	当观众的连麦申请无响应时触发此回调。
onKickedOffSeat	当观众被主播踢下麦位时触发此回调。

CoGuestState

CoGuestStore 对外提供的连麦相关状态数据。

属性	类型	说明
connected	[SeatUserInfo]	已经在麦位上的用户列表。
invitees	[LiveUserInfo]	主播发出邀请的用户列表。
applicants	[LiveUserInfo]	主播收到申请连麦的用户列表。
candidates	[LiveUserInfo]	候选连麦的用户列表。

使用示例

```
// 创建 store 实例
let store = CoGuestStore.create(liveID: "live_room_123")
// 订阅状态变化
store.state.subscribe { state in
    print("已连接用户: \(state.connected.count)")
    print("待处理申请: \(state.applicants.count)")
}
// 订阅主播事件（主播使用）
store.hostEventPublisher.sink { event in
    switch event {
    case .onGuestApplicationReceived(let guestUser):
        print("收到来自 \(guestUser.userName) 的申请")
        // 显示接受/拒绝的 UI
    case .onHostInvitationResponded(let isAccept, let guestUser):
        print("观众 \(guestUser.userName) \(isAccept ? "接受了" : "拒绝了")")
    default:
        break
    }
}
// 主播：接受申请
```

```
store.acceptApplication(userID: "user_456") { code, message in
    if code == 0 {
        print("申请接受成功")
    }
}
```

CoHostStore

最近更新时间：2026-04-20 17:34:06

简介

跨房连线功能允许不同直播间的主播进行实时互动。`CoHostStore` 提供了一套全面的 API 来管理整个跨房连线生命周期。

> **重要：**
> 请始终使用工厂方法 **create(liveID:)** 并提供有效的直播间 ID 来创建 `CoHostStore` 实例。不要尝试直接初始化。

> **说明：**
> 连线状态更新通过 **state** 发布者传递。订阅它以接收有关连线状态、已连接主播、邀请和申请的实时更新。

> **警告：**
> 如果连线请求在指定的超时时间内未收到响应，将触发超时事件。请始终在 UI 中处理超时场景。

功能特性

- **双向连线**：主播可以向其他主播发起连线请求，也可以接收其他主播的连线请求。
- **状态管理**：实时跟踪连线状态、已连接主播、邀请列表和申请者。
- **事件驱动架构**：提供连线事件流用于监听各种连线状态变化。
- **布局模板**：支持多种连线布局模板，如动态网格布局和1对6布局。

可订阅数据

CoHostState 的字段描述如下：

属性名	类型	描述
coHostStatus	CoHostStatus	跨房连线的实时状态。
connected	[SeatUserInfo]	正在和当前直播间连线的主播列表。
invitees	[SeatUserInfo]	向其他直播间发出请求的主播列表。
applicant	SeatUserInfo?	向当前直播间发起连线请求的主播。

candidatesCursor	String	推荐用户列表游标。
candidates	[SeatUserInfo]	推荐用户列表。

API 列表

函数名	描述
create	创建对象实例。
coHostEventPublisher	连线事件发布者。
requestHostConnection	发起连线请求。
cancelHostConnection	取消连线请求。
acceptHostConnection	接受连线请求。
rejectHostConnection	拒绝连线请求。
exitHostConnection	退出连线。
muteRemoteHostAudio	禁音/取消禁音远端主播的音频。
getCoHostCandidates	获取推荐主播列表。

创建实例

create

创建 CoHostStore 实例

```
public static func create(liveID: String) -> CoHostStore {
    let store: CoHostStoreImpl = StoreFactory.shared.getStore(liveId:
liveID)
    return store
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播间 ID。

观察状态和事件

coHostEventPublisher

连线事件发布者

连线操作

requestHostConnection

发起主播连线请求

```
public func requestHostConnection(targetHost liveId: String,
                                  layoutTemplate: CoHostLayoutTemplate,
                                  timeout: TimeInterval,
                                  extraInfo: String = "",
                                  completion: CompletionClosure?)
{
    fatalError("\(#function) must be overridden by subclass")
}
```

向目标主播发起跨房连线请求。

调用此方法后，会向目标主播发送连线请求。请求将保持活动状态，直到：

- 目标主播通过 acceptHostConnection(fromHostLiveID:completion:) 接受
- 目标主播通过 rejectHostConnection(fromHostLiveID:completion:) 拒绝
- 超时时间到期
- 您通过 cancelHostConnection(toHostLiveID:completion:) 取消

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

liveId	String	必填	目标主播的直播间 ID。
layoutTemplate	CoHostLayoutTemplate	必填	连线布局模板。
timeout	TimeInterval	必填	请求超时时间（单位：秒）。
extraInfo	String	必填	扩展信息。
completion	CompletionClosure?	必填	发起请求成功的回调。

cancelHostConnection

取消主播连线请求

```
public func cancelHostConnection(toHostLiveID: String, completion:
CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
toHostLiveID	String	必填	目标主播的直播间 ID。
completion	CompletionClosure?	必填	取消请求成功的回调。

acceptHostConnection

接受主播连线请求

```
public func acceptHostConnection(fromHostLiveID: String, completion:
CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
fromHostLiveID	String	必填	发起连线请求的主播直播间 ID。
completion	Completion Closure?	必填	接受成功的回调。

rejectHostConnection

拒绝主播连线请求

```
public func rejectHostConnection(fromHostLiveID: String, completion:
CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
fromHostLiveID	String	必填	发起连线请求的主播直播间 ID。
completion	Completion Closure?	必填	拒绝成功的回调。

exitHostConnection

退出主播连线

```
public func exitHostConnection(completion: CompletionClosure? = nil) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
completion	Completion Closure?	必填	退出连线成功的回调。

muteRemoteHostAudio

禁音/取消禁音远端主播的音频

```
public func muteRemoteHostAudio(liveID: String,
                                isMuted: Bool,
                                completion: CompletionClosure?)
{
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveID	String	必填	远端主播的直播间 ID。
isMuted	Bool	必填	是否禁音远端主播的音频。`true` 表示禁音，`false` 表示取消禁音。
completion	Completion Closure?	必填	操作结果的回调。

getCoHostCandidates

获取可以与当前主播连线的推荐主播列表

```
public func getCoHostCandidates(cursor: String,
                                completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
cursor	String	必填	游标。
completion	Completion Closure?	必填	完成回调。

数据结构

CoHostStatus

当前用户的跨房连线状态。

枚举值	值	说明
connected	0	和其他主播正在连线中。
disconnected	1	没有和其他主播连线。

CoHostLayoutTemplate

连线布局模板。

枚举值	值	说明
hostVoiceConnection	2	语聊房连线布局。
hostDynamicGrid	600	主播动态网格布局。
hostDynamic1v6	601	主播动态1对6布局。
hostVideoLeftFocus9Seats	602	主播视频左焦点9席位布局。
hostVideoUniformGrid9Seats	603	主播视频均匀网格9席位布局。

CoHostEvent

连线请求的回调事件。

枚举值	说明
onCoHostRequestReceived	当收到连线请求时触发此回调。

onCoHostRequestCancelled	当连线请求被取消时触发此回调。
onCoHostRequestAccepted	当连线请求被接受时触发此回调。
onCoHostRequestRejected	当连线请求被拒绝时触发此回调。
onCoHostRequestTimeout	当连线请求超时时触发此回调。
onCoHostUserJoined	当用户加入连线时触发此回调。
onCoHostUserLeft	当用户离开连线时触发此回调。

CoHostState

CoHostStore 对外提供的跨房连线相关状态数据。

属性	类型	说明
coHostStatus	CoHostStatus	跨房连线的实时状态。
connected	[SeatUserInfo]	正在和当前直播间连线的主播列表。
invitees	[SeatUserInfo]	向其他直播间发出请求的主播列表。
applicant	SeatUserInfo?	向当前直播间发起连线请求的主播。
candidatesCursor	String	推荐用户列表游标。
candidates	[SeatUserInfo]	推荐用户列表。

使用示例

```
// 创建 store 实例
let store = CoHostStore.create(liveID: "live_room_123")
// 订阅状态变化
store.state.subscribe { state in
    print("连线状态: \(state.coHostStatus)")
    print("已连接主播: \(state.connected.count)")
}
// 订阅连线事件
```

```
store.coHostEventPublisher.sink { event in
    switch event {
    case .onCoHostRequestReceived(let inviter, let extensionInfo):
        print("收到来自 \(inviter.userName) 的连线请求")
        // 显示接受/拒绝的 UI
    case .onCoHostRequestAccepted(let invitee):
        print("连线请求被 \(invitee.userName) 接受")
    case .onCoHostUserJoined(let userInfo):
        print("主播 \(userInfo.userName) 加入连线")
    default:
        break
    }
}
// 发起连线请求
store.requestHostConnection(
    targetHost: "target_live_id",
    layoutTemplate: .hostDynamicGrid,
    timeout: 30,
    extraInfo: "",
    completion: { code, message in
        if code == 0 {
            print("连线请求发送成功")
        }
    }
)
```

DeviceStore

最近更新时间：2026-04-20 17:34:06

简介

`DeviceStore` 提供了一套全面的 API 来管理音视频设备，包括麦克风、摄像头和屏幕分享等功能。

> **重要：**
> 请使用 **shared** 单例获取 `DeviceStore` 实例。不要尝试直接初始化。

> **说明：**
> 设备状态更新通过 **state** 发布者传递。订阅它以接收有关麦克风、摄像头、网络等状态的实时更新。

功能特性

- **麦克风管理**：打开/关闭麦克风，设置采集音量和输出音量。
- **摄像头管理**：打开/关闭摄像头，切换前后摄像头，设置镜像和视频质量。
- **音频路由**：切换扬声器和听筒。
- **屏幕分享**：开启和关闭屏幕分享功能。
- **网络状态**：实时监控网络质量信息。

可订阅数据

DeviceState 的字段描述如下：

属性名	类型	描述
microphoneStatus	DeviceStatus	麦克风状态。
microphoneLastError	DeviceError	麦克风错误，用于出现报错时提取错误信息。
captureVolume	Int	采集音量，取值范围 0, 100。
currentMicVolume	Int	当前用户实际输出音量。
outputVolume	Int	最大输出音量，取值范围 0, 100。
cameraStatus	DeviceStatus	摄像头状态。
cameraLastError	DeviceError	摄像头错误，用于出现报错时提取错误信息。

isFrontCamera	Bool	是否为前置摄像头。
localMirrorType	MirrorType	镜像状态。
localVideoQuality	VideoQuality	本地视频质量。
currentAudioRoute	AudioRoute	当前音频路由位置。
screenStatus	DeviceStatus	屏幕分享状态。
networkInfo	NetworkInfo	网络信息。
networkType	NetworkType	当前网络类型。

API 列表

函数名	描述
shared	单例对象。
openLocalMicrophone	打开本地麦克风。
closeLocalMicrophone	关闭本地麦克风。
setCaptureVolume	设置采集音量。
setOutputVolume	设置输出音量。
setAudioRoute	设置音频路由。
startCameraTest	开始摄像头测试。
stopCameraTest	停止摄像头测试。
openLocalCamera	打开本地摄像头。
closeLocalCamera	关闭本地摄像头。
switchCamera	切换摄像头。
switchMirror	切换镜像状态。
updateVideoQuality	更新视频质量。

startScreenShare	开启屏幕分享。
stopScreenShare	关闭屏幕分享。
reset	重置为默认状态。

获取实例

shared

单例对象。

麦克风操作

openLocalMicrophone

打开本地麦克风

```
public func openLocalMicrophone(completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
completion	CompletionClosure?	必填	操作是否成功。

closeLocalMicrophone

关闭本地麦克风

```
public func closeLocalMicrophone() {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息
从3.5版本开始支持。

setCaptureVolume

设置采集音量

```
public func setCaptureVolume(volume: Int) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
volume	Int	必填	采集音量，取值范围 0, 100。

setOutputVolume

设置最大输出音量

```
public func setOutputVolume(_ volume: Int) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
volume	Int	必填	最大音量，取值范围 0, 100。

音频路由

setAudioRoute

设置音频路由

```
public func setAudioRoute(_ route: AudioRoute) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
route	AudioRoute	必填	路由位置。

摄像头操作

startCameraTest

开始摄像头测试，如果摄像头打开成功，会将画面渲染到设置的 CameraView 上

```
public func startCameraTest(
    cameraView: UIView,
    completion: CompletionClosure?
) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
cameraView	UIView	必填	摄像头采集画面的渲染视图。
completion	CompletionClosure?	必填	操作是否成功。

stopCameraTest

停止摄像头测试

```
public func stopCameraTest() {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

openLocalCamera

打开本地摄像头

```
public func openLocalCamera(
    isFront: Bool,
    completion: CompletionClosure?
) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
isFront	Bool	必填	是否前置摄像头。
completion	CompletionClosure?	必填	操作是否成功。

closeLocalCamera

关闭本地摄像头

```
public func closeLocalCamera() {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

switchCamera

切换摄像头

```
public func switchCamera(isFront: Bool) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
isFront	Bool	必填	是否前置摄像头。

switchMirror

切换镜像状态

```
public func switchMirror(mirrorType: MirrorType) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
mirrorType	MirrorType	必填	镜像状态。

updateVideoQuality

更新视频质量

```
public func updateVideoQuality(_ quality: VideoQuality) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
quality	VideoQuality	必填	视频质量。

屏幕分享

startScreenShare

开启屏幕分享

```
public func startScreenShare(appGroup: String) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
appGroup	String	必填	App Group ID。

stopScreenShare

关闭屏幕采集

```
public func stopScreenShare() {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

重置

reset

重置为默认状态

```
public func reset() {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

数据结构

DeviceType

设备类型。

枚举值	值	说明
microphone	0	麦克风类型。
camera	1	摄像头类型。
screenShare	2	屏幕分享类型。

DeviceError

设备相关错误码。

枚举值	值	说明
noError	0	操作成功。
noDeviceDetected	1	未检测到设备。
noSystemPermission	2	没有系统权限。
notSupportCapture	3	不支持采集。
occupiedError	4	设备已占用。
unknownError	5	未知错误。

DeviceStatus

设备开启状态。

枚举值	值	说明
off	0	关闭。
on	1	开启。

AudioRoute

音频路由。

枚举值	值	说明

speakerphone	0	扬声器，使用扬声器播放（即"免提"），扬声器位于手机底部，声音偏大，适合外放音乐。
earpiece	1	听筒，使用听筒播放，听筒位于手机顶部，声音偏小，适合需要保护隐私的通话场景。

VideoQuality

视频质量。

枚举值	值	说明
quality360P	1	360P。
quality540P	2	540P。
quality720P	3	720P。
quality1080P	4	1080P。

NetworkQuality

网络质量。

枚举值	值	说明
unknown	0	未知网络。
excellent	1	极佳。
good	2	良好。
poor	3	较差。
bad	4	差。
veryBad	5	极差。
down	6	中断。

MirrorType

摄像头镜像状态。

枚举值	值	说明

auto	0	自动，前置摄像头镜像，后置摄像头不镜像。
enable	1	前后摄像头均镜像。
disable	2	前后摄像头均不镜像。

NetworkType

网络类型。

枚举值	值	说明
unknown	0	未知或未识别的网络类型。
wifi	1	WiFi 网络连接。
cellular	2	蜂窝/移动数据网络连接。

DeviceFocusOwner

设备焦点。

枚举值	说明
call	语音通话场景。
live	直播场景。
room	房间场景。
none	未设置。

NetworkInfo

网络信息。

属性	类型	说明
userID	String	用户唯一 ID。
quality	NetworkQuality	网络质量。
upLoss	UInt32	上行丢包率，取值范围 0, 100。
downLoss	UInt32	下行丢包率，取值范围 0, 100。

delay	UInt32	延迟（单位：毫秒）。

DeviceState

设备状态。

属性	类型	说明
microphoneStatus	DeviceStatus	麦克风状态。
microphoneLastError	DeviceError	麦克风错误，用于出现报错时提取错误信息。
captureVolume	Int	采集音量，取值范围 0, 100。
currentMicVolume	Int	当前用户实际输出音量。
outputVolume	Int	最大输出音量，取值范围 0, 100。
cameraStatus	DeviceStatus	摄像头状态。
cameraLastError	DeviceError	摄像头错误，用于出现报错时提取错误信息。
isFrontCamera	Bool	是否为前置摄像头。
localMirrorType	MirrorType	镜像状态。
localVideoQuality	VideoQuality	本地视频质量。
currentAudioRoute	AudioRoute	当前音频路由位置。
screenStatus	DeviceStatus	屏幕分享状态。
networkInfo	NetworkInfo	网络信息。
networkType	NetworkType	当前网络类型。

使用示例

```
// 获取单例实例
let store = DeviceStore.shared
// 订阅状态变化
store.state.subscribe { state in
    print("麦克风状态：\(state.microphoneStatus)")
    print("摄像头状态：\(state.cameraStatus)")
```

```
    print("网络质量: \(state.networkInfo.quality)")
}
// 打开麦克风
store.openLocalMicrophone { code, message in
    if code == 0 {
        print("麦克风打开成功")
    }
}
// 打开前置摄像头
store.openLocalCamera(isFront: true) { code, message in
    if code == 0 {
        print("摄像头打开成功")
    }
}
```

GiftStore

最近更新时间：2026-04-20 17:34:06

简介

`GiftStore` 提供了一套完整的礼物管理 API，包括发送礼物、刷新礼物列表、设置语言和监听礼物事件。 通过该类，可以在直播间内实现礼物互动功能。

> **重要：**
> 使用 **create(liveID:)** 工厂方法创建 `GiftStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 礼物状态更新通过 **state** 发布者传递。订阅它以接收房间内礼物数据的实时更新。

功能特性

- **礼物发送**：支持向当前房间发送指定礼物。
- **礼物列表**：获取和刷新当前房间可用的礼物列表。
- **语言设置**：设置礼物信息的展示语言。
- **事件监听**：监听礼物接收事件。

可订阅数据

GiftState 的字段描述如下：

属性名	类型	描述
usableGifts	[GiftCategory]	当前房间可用的所有礼物分类及礼物列表。

API 列表

函数名	描述
create	创建礼物管理实例。
giftEventPublisher	礼物事件发布者。
sendGift	发送礼物。

refreshUsableGifts	刷新可用礼物列表。
setLanguage	设置展示语言。

创建实例

create

创建礼物管理实例。

```
public static func create(liveID: String) -> GiftStore {
    let store: GiftStoreImpl = StoreFactory.shared.getStore(liveId:
liveID)
    return store
}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播间 ID。

观察状态和事件

giftEventPublisher

礼物事件发布者

礼物操作

sendGift

向当前房间发送指定礼物

```
public func sendGift(giftID: String,
                     count: UInt,
                     completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
giftID	String	必填	要发送的礼物唯一标识 ID。
count	UInt	必填	单次发送的礼物数量。
completion	CompletionClosure?	必填	完成回调（成功/失败状态）。

refreshUsableGifts

手动刷新当前房间的可用礼物列表。

```
public func refreshUsableGifts(completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
completion	CompletionClosure?	必填	完成回调（成功时可通过 state 获取最新礼物列表，失败时返回错误信息）。

setLanguage

设置礼物信息的展示语言。

```
public func setLanguage(_ language: String) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
language	String	必填	语言代码（"zh-CN" 表示中文，"en" 表示英文），设置完成展示界面刷新后礼物名称、描述等会同步更新为对应语言。

数据结构

Gift

礼物数据模型，包含单个礼物的完整属性信息。

属性	类型	说明
giftID	String	礼物 ID。
name	String	礼物名称。
desc	String	礼物描述。
iconURL	String	礼物图标图片的网络 URL，用于加载礼物缩略图。
resourceURL	String	礼物动效资源文件的网络 URL，用于加载礼物展示动效。
level	UInt	礼物等级，用于区分礼物稀有度或价值层级。
coins	UInt	礼物价格（金币）。
extensionInfo	[String: String]	礼物扩展信息，可自定义字段（如特效类型、赠送限制等）。

GiftCategory

礼物分类。

属性	类型	说明
categoryID	String	分类唯一标识 ID，用于区分不同礼物分类。
name	String	分类展示名称，用于 UI 分类显示（如 "热门礼物"，"高级礼物"）。
desc	String	分类描述信息，用于说明该分类的特点。

extensionInfo	[String: String]	分类扩展信息，包含自定义字段（如排序权重、显示样式等）。
giftList	[Gift]	当前分类下的所有礼物列表。

GiftState

礼物状态，管理当前房间的礼物数据状态，支持实时更新并可被订阅监听。

属性	类型	说明
usableGifts	[GiftCategory]	当前房间可用的所有礼物分类及礼物列表。

GiftListener

礼物事件，用于接收直播间/语音聊天房内的礼物动态。

方法

onReceiveGift: 收到新礼物消息的事件回调。当直播间/语音聊天房内有其他观众发送礼物时，会触发该事件并返回相关信息。

```
case onReceiveGift(liveID: String, gift: Gift, count: UInt, sender:
LiveUserInfo)
```

参数名	类型	说明
liveID	String	直播间 ID。
gift	Gift	礼物信息。
count	UInt8	礼物数量。
sender	LiveUserInfo	礼物发送者信息。

LikeStore

最近更新时间：2026-04-20 17:34:06

简介

`LikeStore` 提供了一套完整的点赞管理 API，包括发送点赞、监听点赞事件和获取点赞状态。通过该类，可以在直播间内实现点赞互动功能。

> **重要：**
> 使用 **create(liveID:)** 工厂方法创建 `LikeStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 点赞状态更新通过 **state** 发布者传递。订阅它以接收房间内点赞数据的实时更新。

功能特性

- **点赞发送**：支持向当前房间发送点赞。
- **点赞状态**：获取当前房间的累计点赞数。
- **事件监听**：监听点赞接收事件。

数据结构

LikeState

点赞状态，用于展示和订阅直播间/语音聊天房的点赞信息。

属性	类型	说明
totalLikeCount	UInt	当前直播间/语音聊天房的累计总点赞数，支持实时更新并可被订阅监听。

LikeListener

点赞事件，用于接收直播间/语音聊天房内的点赞动态。
此监听器用于接收直播间/语音聊天房内的点赞动态。
方法
onReceiveLikesMessage: 收到新点赞消息的事件回调。当直播间/语音聊天房内有其他观众发送点赞时，会触发该事件并返回相关信息。

```
case onReceiveLikesMessage(liveID: String, totalLikesReceived: UInt,
sender: LiveUserInfo)
```

参数名	类型	说明
liveID	String	直播间 ID。
totalLikesReceived	UInt	本次收到的新点赞数。
sender	LiveUserInfo	点赞发送者信息。

LiveAudienceStore

最近更新时间：2026-04-20 17:34:06

简介

`LiveAudienceStore` 提供了一套完整的观众管理 API，包括获取观众列表、设置管理员、踢出用户、禁言等功能。 通过该类，可以在直播间内实现观众管理功能。

> **重要：**
> 使用 **create(liveID:)** 工厂方法创建 `LiveAudienceStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 观众状态更新通过 **state** 发布者传递。订阅它以接收房间内观众数据的实时更新。

功能特性

- **观众列表**：获取和管理当前房间的观众列表。
- **权限管理**：设置和撤销管理员权限。
- **用户管理**：踢出用户、禁言等操作。
- **事件监听**：监听房主、管理员、观众加入/离开等事件。

可订阅数据

LiveAudienceState 的字段描述如下：

属性名	类型	描述
audienceList	[LiveUserInfo]	观众列表。
audienceCount	UInt	观众数量。
messageBannedUserList	[LiveUserInfo]	消息被禁言的用户列表。

API 列表

函数名	描述
create	创建观众管理实例。

liveAudienceEventPublisher	观众事件发布者。
fetchAudienceList	获取观众列表。
setAdministrator	设置管理员。
revokeAdministrator	撤销管理员。
kickUserOutOfRoom	踢出用户。
disableSendMessage	禁言/解禁用户。

创建实例

create

创建观众管理实例

```
public static func create(liveID: String) -> LiveAudienceStore {
    let store: LiveAudienceStoreImpl =
StoreFactory.shared.getStore(liveId: liveID)
    return store
}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播间 ID。

观察状态和事件

liveAudienceEventPublisher

观众事件发布者

观众管理

fetchAudienceList

获取观众列表

```
public func fetchAudienceList(
    completion: CompletionClosure?
) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
completion	Completion Closure?	必填	完成回调。

setAdministrator

设置管理员

```
public func setAdministrator(userID: String,
                             completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	要设置为管理员的用户 ID。
completion	Completion Closure?	必填	完成回调。

revokeAdministrator

撤销管理员

```
public func revokeAdministrator(userID: String,
```

```
                                completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	要撤销管理员权限的用户 ID。
completion	Completion Closure?	必填	完成回调。

kickUserOutOfRoom

将用户踢出房间

```
public func kickUserOutOfRoom(userID: String,
                              completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	要踢出的用户 ID。
completion	Completion Closure?	必填	完成回调。

disableSendMessage

禁用/解禁用户发送消息

```
public func disableSendMessage(userID: String,
                               isDisable: Bool,
                               completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
```

```
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	目标用户 ID。
isDisable	Bool	必填	true 表示禁用发送消息，false 表示解禁。
completion	Completion Closure?	必填	完成回调。

数据结构

Role

用户角色。

枚举值	说明
owner	房主。
admin	管理员。
generalUser	普通用户。

LiveUserInfo

直播用户信息。

属性	类型	说明
userID	String	用户唯一标识 ID。
userName	String	用户名称。
avatarURL	String	用户头像 URL。

LiveAudienceState

直播观众状态。

属性	类型	说明
audienceList	[LiveUserInfo]	观众列表。
audienceCount	UInt	观众数量。
messageBannedUserList	[LiveUserInfo]	消息被禁言的用户列表。

LiveAudienceListener

直播观众事件。
此监听器用于接收直播间内全角色（房主、管理员、观众）的动态事件。
方法
onOwnerJoined: 房主加入事件。

```
case onOwnerJoined(owner: LiveUserInfo)
```

参数名	类型	说明
owner	LiveUserInfo	加入的房主信息。

onOwnerLeft: 房主离开事件。

```
case onOwnerLeft(owner: LiveUserInfo)
```

参数名	类型	说明
owner	LiveUserInfo	离开的房主信息。

onAdminJoined: 管理员加入事件。

```
case onAdminJoined(admin: LiveUserInfo)
```

参数名	类型	说明
admin	LiveUserInfo	加入的管理员信息。

onAdminLeft: 管理员离开事件。

```
case onAdminLeft(admin: LiveUserInfo)
```

参数名	类型	说明
admin	LiveUserInfo	离开的管理员信息。

onAudienceJoined: 观众加入事件。

```
case onAudienceJoined(audience: LiveUserInfo)
```

参数名	类型	说明
audience	LiveUserInfo	加入的观众信息。

onAudienceLeft: 观众离开事件。

```
case onAudienceLeft(audience: LiveUserInfo)
```

参数名	类型	说明
audience	LiveUserInfo	离开的观众信息。

onAudienceMessageDisabled: 观众被禁止发言事件。

```
case onAudienceMessageDisabled(audience: LiveUserInfo, isDisable: Bool)
```

参数名	类型	说明
audience	LiveUserInfo	观众信息。
isDisable	Bool	是否被禁止发言。

LiveCoreView

最近更新时间：2026-04-20 17:34:06

简介

`LiveCoreView` 提供了直播推流和播放的视图容器，支持多人连麦、PK 等功能。 通过该组件，可以实现直播间的视频渲染和交互。

> **重要：**
> 使用前需要先调用 **setLiveID(_:)** 设置直播间 ID。

功能特性

- **视频渲染**：提供直播推流和播放的视图容器。
- **连麦支持**：支持多人连麦功能。
- **PK 支持**：支持主播 PK 功能。
- **房间外预览**：支持在进入房间前预览直播流。

数据结构

CoreViewType

核心视图类型。

枚举值	说明
playView	播放视图。
pushView	推流视图。

ViewLayer

视图层级。

枚举值	说明
foreground	前景层。
background	背景层。

VideoViewAdapter

视频视图适配器协议。

方法

createCoGuestView: 创建连麦视图。

```
func createCoGuestView(seatInfo: SeatInfo, viewLayer: ViewLayer) ->
UIView?
```

参数名	类型	说明
seatInfo	SeatInfo	连麦用户的麦位信息。
viewLayer	ViewLayer	视图层级，前景层或背景层。

createCoHostView: 创建跨房连麦视图。

```
func createCoHostView(seatInfo: SeatInfo, viewLayer: ViewLayer) ->
UIView?
```

参数名	类型	说明
seatInfo	SeatInfo	跨房连麦用户的麦位信息。
viewLayer	ViewLayer	视图层级，前景层或背景层。

createBattleView: 创建 PK 视图。

```
func createBattleView(seatInfo: SeatInfo) -> UIView?
```

参数名	类型	说明
seatInfo	SeatInfo	PK 用户的麦位信息。

createBattleContainerView: 创建 PK 容器视图。

```
func createBattleContainerView() -> UIView?
```

LiveListStore

最近更新时间：2026-04-20 17:34:06

简介

`LiveListStore` 提供了一套完整的直播间管理 API，包括开播、加入直播、离开直播、结束直播等功能。 通过该类，可以实现直播间的生命周期管理。主播调用 startLive/endLive 来开播和解散房间，观众调用 joinLive/leaveLive 来加入和退出直播间。

> **重要：**
> 使用 **shared** 单例对象获取 `LiveListStore` 实例。

> **说明：**
> 直播状态更新通过 **state** 发布者传递。订阅它以接收直播数据的实时更新。

功能特性

- **直播列表**：获取和管理直播间列表。
- **开播**：主播开播。如果是新的直播间 ID，将创建直播间并开播；如果是服务端已创建的直播间 ID，将加入该直播间并开播。
- **直播加入**：加入已存在的直播间。
- **直播管理**：更新直播信息、结束直播等操作。
- **事件监听**：监听直播结束、被踢出等事件。

可订阅数据

LiveListState 的字段描述如下：

属性名	类型	描述
liveList	[LiveInfo]	直播列表。
liveListCursor	String	直播列表游标。
currentLive	LiveInfo	当前直播信息。

API 列表

函数名	描述

shared	单例对象。
liveListEventPublisher	直播列表事件发布者。
fetchLiveList	获取直播列表。
fetchLiveInfo	获取直播信息。
startLive	主播开播。如果是新的直播间 ID，将创建直播间并开播；如果是服务端已创建的直播间 ID，将加入该直播间并开播。
joinLive	加入直播（观众调用）。
leaveLive	离开直播（观众调用）。
endLive	结束直播（主播调用）。
updateLiveInfo	更新直播信息。
queryMetaData	查询元数据。
updateLiveMetaData	更新元数据。

获取实例

shared

单例对象。

观察状态和事件

liveListEventPublisher

直播列表事件发布者

直播列表

fetchLiveList

获取直播列表

```
public func fetchLiveList(cursor: String,
                          count: Int,
                          completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
```

```
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
cursor	String	必填	游标。
count	Int	必填	数量。
completion	Completion Closure?	必填	完成回调。

fetchLiveInfo

获取直播信息

```
public func fetchLiveInfo(liveID: String,
                          completion: LiveInfoCompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播间 ID。
completion	LiveInfoCompletionClosure?	必填	完成回调。

直播操作

startLive

开播

```
public func startLive(_ liveInfo: LiveInfo,
```

```
                        completion: LiveInfoCompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

主播开播。如果是新的直播间 ID，将创建直播间并开播；如果是服务端已创建的直播间 ID，将加入该直播间并开播。

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveInfo	LiveInfo	必填	直播信息。
completion	LiveInfoCompletionClosure?	必填	完成回调。

joinLive

加入直播

```
public func joinLive(liveID: String,
                     completion: LiveInfoCompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

观众调用此接口加入已存在的直播间。

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播 ID。
completion	LiveInfoCompletionClosure?	必填	完成回调。

leaveLive

离开直播

```
public func leaveLive(completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

观众调用此接口离开当前直播间。
若主播仅需退出房间但不解散直播间，也可调用此接口。
版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
completion	CompletionClosure?	必填	完成回调。

endLive

结束直播（主播调用）

```
public func endLive(completion: StopLiveCompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

主播调用此接口结束当前直播并解散房间。
版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
completion	StopLiveCompletionClosure?	必填	完成回调。

updateLiveInfo

更新直播信息

```
public func updateLiveInfo(_ liveInfo: LiveInfo,
                           modifyFlag: LiveInfo.ModifyFlag,
                           completion: CompletionClosure?) {
```

```
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveInfo	LiveInfo	必填	直播信息。
modifyFlag	LiveInfo.ModifyFlag	必填	修改标志。
completion	CompletionClosure?	必填	完成回调。

元数据操作

queryMetaData

查询元数据

```
public func queryMetaData(keys: [String], completion:
MetaDataCompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
keys	[String]	必填	键列表。
completion	MetaDataCompletionClosure?	必填	完成回调。

updateLiveMetaData

更新直播元数据

```
public func updateLiveMetaData(_ metaData: [String: String],
                               completion: CompletionClosure?) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
metaData	[String: String]	必填	元数据。
completion	CompletionClosure?	必填	完成回调。

数据结构

TakeSeatMode

上麦模式。

枚举值	说明
free	自由上麦。
apply	申请上麦。

SeatLayoutTemplate

麦位布局模板，用于简化创建直播间时的麦位配置。

枚举值	说明
videoDynamicGrid9Seats	竖屏动态九宫格，适用于视频直播场景。
videoDynamicFloat7Seats	竖屏动态1v6浮动布局，适用于视频直播场景。
videoLeftFocus9Seats	竖屏左侧焦点九宫格布局，适用于视频直播场景。

videoUniformGrid9Seats	竖屏均匀九宫格布局，适用于视频直播场景。
videoFixedGrid9Seats	竖屏静态九宫格，适用于视频直播场景。
videoFixedFloat7Seats	竖屏静态1v6浮动布局，适用于视频直播场景。
videoLandscape4Seats	横屏4人麦位布局，适用于视频直播场景。
karaoke	语音 KTV 布局，适用于 K 歌场景，可指定麦位数量。
audioSalon	语音沙龙布局，适用于语聊场景，可指定麦位数量。

LiveEndedReason

直播结束原因。

枚举值	说明
endedByHost	主播主动结束。
endedByServer	服务器结束。

LiveKickedOutReason

被踢出直播间原因。

枚举值	说明
byAdmin	被管理员踢出。
byLoggedOnOtherDevice	在其他设备登录。
byServer	被服务器踢出。
forNetworkDisconnected	网络断开连接。
forJoinRoomStatusInvalidDuringOffline	离线期间房间状态无效。

forCountOfJoinedRoomsExceedLimit	加入房间数量超过限制。

LiveInfo

直播信息。

属性	类型	说明
liveID	String	直播 ID。
liveName	String	直播名称。
notice	String	直播公告。
isMessageDisable	Bool	是否禁用消息。
isPublicVisible	Bool	是否公开可见。
isSeatEnabled	Bool	是否启用麦位。
keepOwnerOnSeat	Bool	房主是否保持在麦位上。
maxSeatCount	Int	最大麦位数量。
seatMode	TakeSeatMode	上麦模式。
seatTemplate	SeatLayoutTemplate	麦位布局模板，用于简化麦位配置。
seatLayoutTemplateID	UInt	麦位布局模板 ID。
coverURL	String	封面 URL。
backgroundURL	String	背景 URL。
categoryList	[NSNumber]	分类列表。
activityStatus	Int	活动状态。
liveOwner	LiveUserInfo	直播房主信息。
createTime	Int	创建时间。
totalViewerCount	Int	总观看人数。

isGiftEnabled	Bool	是否启用礼物。
metaData	[String: String]	元数据。

LiveListState

直播列表状态。

属性	类型	说明
liveList	[LiveInfo]	直播列表。
liveListCursor	String	直播列表游标。
currentLive	LiveInfo	当前直播信息。

LiveListListener

直播列表事件。
方法
onLiveEnded: 直播结束事件。

```
case onLiveEnded(liveID: String, reason: LiveEndedReason, message:
String)
```

参数名	类型	说明
liveID	String	直播 ID。
reason	LiveEndedReason	结束原因。
message	String	消息。

onKickedOutOfLive: 被踢出直播间事件。

```
case onKickedOutOfLive(liveID: String, reason: LiveKickedOutReason,
message: String)
```

参数名	类型	说明
liveID	String	直播 ID。
reason	LiveKickedOutReason	被踢出原因。

message	String	消息。

LiveSeatStore

最近更新时间：2026-04-20 17:34:06

简介

`LiveSeatStore` 提供了一套完整的麦位管理 API，包括上麦、下麦、锁麦、解锁麦位、踢用户下麦、远程控制设备等功能。 通过该类，可以在直播间内实现麦位管理功能。

> **重要：**
> 使用 **create(liveID:)** 工厂方法创建 `LiveSeatStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 麦位状态更新通过 **state** 发布者传递。订阅它以接收房间内麦位数据的实时更新。

功能特性

- **麦位管理**：上麦、下麦、锁麦、解锁麦位等操作。
- **用户管理**：踢用户下麦、移动用户到指定麦位。
- **设备控制**：远程控制用户的摄像头和麦克风。
- **事件监听**：监听麦位相关事件。

可订阅数据

LiveSeatState 的字段描述如下：

属性名	类型	描述
seatList	[SeatInfo]	麦位列表。
canvas	LiveCanvas	画布信息。
speakingUsers	[String: Int]	正在说话的用户。
avStatistics	[AVStatistics]	音视频相关统计信息。

API 列表

函数名	描述
create	创建麦位管理实例。

liveSeatEventPublisher	麦位事件发布者。
takeSeat	上麦。
leaveSeat	下麦。
lockSeat	锁麦。
unlockSeat	解锁麦位。
kickUserOutOfSeat	踢用户下麦。
moveUserToSeat	移动用户。
openRemoteCamera	开启远程摄像头。
closeRemoteCamera	关闭远程摄像头。
openRemoteMicrophone	开启远程麦克风。
closeRemoteMicrophone	关闭远程麦克风。

创建实例

create

创建 LiveSeatStore 实例

```
public static func create(liveID: String) -> LiveSeatStore {
    let store: LiveSeatStoreImpl = StoreFactory.shared.getStore(liveId:
liveID)
    return store
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

liveID	String	必填	直播间 ID。

观察状态和事件

liveSeatEventPublisher

麦位事件发布者

麦位操作

takeSeat

上麦

```
public func takeSeat(seatIndex: Int,
                     completion: CompletionClosure?)
{
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
seatIndex	Int	必填	麦位索引。
completion	CompletionClosure?	必填	完成回调。

leaveSeat

下麦

```
public func leaveSeat(completion: CompletionClosure? = nil) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
completion	Completion Closure?	必填	完成回调。

lockSeat

锁定麦位

```
public func lockSeat(
    seatIndex: Int,
    completion: CompletionClosure?
) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
seatIndex	Int	必填	麦位索引。
completion	Completion Closure?	必填	完成回调。

unlockSeat

解锁麦位

```
public func unlockSeat(
    seatIndex: Int,
    completion: CompletionClosure?
) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
seatIndex	Int	必填	麦位索引。
completion	Completion Closure?	必填	完成回调。

用户管理

kickUserOutOfSeat

踢用户下麦

```
public func kickUserOutOfSeat(userID: String,
                              completion: CompletionClosure?)
{
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。
completion	Completion Closure?	必填	完成回调。

moveUserToSeat

移动用户到麦位

```
public func moveUserToSeat(
    userID: String,
    targetIndex: Int,
    policy: MoveSeatPolicy?,
    completion: CompletionClosure?
) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。
targetIndex	Int	必填	目标麦位索引。
policy	MoveSeatPolicy?	必填	移动策略。
completion	CompletionClosure?	必填	完成回调。

远程设备控制

openRemoteCamera

开启远程摄像头

```
public func openRemoteCamera(
    userID: String,
    policy: DeviceControlPolicy,
    completion: CompletionClosure?
) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。
policy	DeviceControlPolicy	必填	设备控制策略。
completion	CompletionClosure?	必填	完成回调。

closeRemoteCamera

关闭远程摄像头

```
public func closeRemoteCamera(
    userID: String,
    completion: CompletionClosure?
) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。
completion	CompletionClosure?	必填	完成回调。

openRemoteMicrophone

开启远程麦克风

```
public func openRemoteMicrophone(
    userID: String,
    policy: DeviceControlPolicy,
    completion: CompletionClosure?
) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。

policy	DeviceControlPolicy	必填	设备控制策略。
completion	CompletionClosure?	必填	完成回调。

closeRemoteMicrophone

关闭远程麦克风

```
public func closeRemoteMicrophone(
    userID: String,
    completion: CompletionClosure?
) {
    fatalError("\(#function) must be overridden by subclass")
}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。
completion	CompletionClosure?	必填	完成回调。

数据结构

MoveSeatPolicy

移动麦位策略。

枚举值	说明
abortWhenOccupied	被占用时中止。
forceReplace	强制替换。
swapPosition	交换位置。

DeviceControlPolicy

设备控制策略。

枚举值	说明
unlockOnly	仅解锁。

SuspendStatus

用户的挂起状态。

枚举值	说明
none	未挂起。
inBackground	用户进入后台挂起。
inCalling	用户正在接听电话。

LiveSeatEvent

麦位相关的回调事件。

枚举值	说明
onLocalCameraOpenedByAdmin	当本地摄像头被管理员开启时触发此回调。
onLocalCameraClosedByAdmin	当本地摄像头被管理员关闭时触发此回调。
onLocalMicrophoneOpenedByAdmin	当本地麦克风被管理员开启时触发此回调。
onLocalMicrophoneClosedByAdmin	当本地麦克风被管理员关闭时触发此回调。

SeatUserInfo

麦位用户信息。

属性	类型	说明
userID	String	用户 ID。
userName	String	用户名。
avatarURL	String	头像 URL。

role	Role	用户角色。
liveID	String	直播间 ID。
microphoneStatus	DeviceStatus	麦克风状态。
allowOpenMicrophone	Bool	是否允许开启麦克风。
cameraStatus	DeviceStatus	摄像头状态。
allowOpenCamera	Bool	是否允许开启摄像头。
userSuspendStatus	SuspendStatus	用户挂起状态。

RegionInfo

麦位视图坐标信息。

属性	类型	说明
x	CGFloat	X 坐标。
y	CGFloat	Y 坐标。
w	CGFloat	宽度。
h	CGFloat	高度。
zorder	Int	层级顺序。

AVStatistics

音视频相关统计信息。

属性	类型	说明
userID	String	用户 ID。
videoBitrate	UInt	本地视频的码率。
videoWidth	CGFloat	本地视频的宽度。
videoHeight	CGFloat	本地视频的高度。
frameRate	UInt	本地视频的帧率。

audioSampleRate	UInt	音频的采样率。
audioBitrate	UInt	音频码率。

SeatInfo

麦位信息。

属性	类型	说明
index	Int	麦位索引。
isLocked	Bool	是否锁定。
userInfo	SeatUserInfo	用户信息。
region	RegionInfo	区域信息。

LiveCanvas

直播画布。

属性	类型	说明
w	CGFloat	宽度。
h	CGFloat	高度。
templateID	UInt	模板 ID。

LiveSeatState

LiveSeatStore 对外提供的麦位状态数据。

属性	类型	说明
seatList	[SeatInfo]	麦位列表。
canvas	LiveCanvas	画布信息。
speakingUsers	[String: Int]	正在说话的用户。
avStatistics	[AVStatistics]	音视频相关统计信息。

LoginStore

最近更新时间：2026-04-20 17:34:06

简介

`LoginStore` 提供了一套完整的登录管理 API，包括用户登录、登出、设置个人信息等功能。 通过该类，可以管理用户的登录状态和用户资料。

> **重要：**
> 使用 **shared** 单例对象访问 `LoginStore` 实例。

> **说明：**
> 登录状态更新通过 **state** 发布者传递。订阅它以接收登录状态的实时更新。

功能特性

- **用户登录**：支持使用 SDK 应用 ID、用户 ID 和用户签名进行登录。
- **用户登出**：支持用户登出操作。
- **个人信息设置**：支持设置用户昵称、头像、性别等个人资料。

可订阅数据

LoginState 的字段描述如下：

属性名	类型	描述
loginStatus	LoginStatus	登录状态。
loginUserInfo	UserProfile?	登录用户信息。

API 列表

函数名	描述
shared	单例对象。
loginEventPublisher	登录事件发布者。
login	登录。
logout	登出。

setSelfInfo	设置个人信息。

获取实例

shared

单例对象

```
public static let shared: LoginStore = LoginStoreImpl()
```

版本信息
从3.5版本开始支持。

观察事件

loginEventPublisher

登录事件发布者

登录操作

login

登录

```
public func login(sdkAppID: Int32, userID: String, userSig: String,
completion: CompletionClosure?) {}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
sdkAppID	Int32	必填	SDK 应用 ID。
userID	String	必填	用户 ID。
userSig	String	必填	用户签名。
completion	CompletionClosure?	必填	完成回调。

logout

登出

```
public func logout(completion: CompletionClosure?) {}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
completion	CompletionClosure?	必填	完成回调。

setSelfInfo

设置个人信息

```
public func setSelfInfo(userProfile: UserProfile, completion:
CompletionClosure?) {}
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userProfile	UserProfile	必填	用户资料。
completion	CompletionClosure?	必填	完成回调。

数据结构

LoginStatus

登录状态。

枚举值	说明
unlogin	未登录。
logined	已登录。

AllowType

好友验证方式。

枚举值	值	说明
allowAny	0	允许任何人。
needConfirm	1	需要验证。
denyAny	2	拒绝任何人。

Gender

性别。

枚举值	值	说明
unknown	0	未知。
male	1	男性。
female	2	女性。

LoginEvent

登录事件。

枚举值	说明
kickedOffline	当前用户被踢下线。
loginExpired	登录票据过期。

UserProfile

用户资料。

属性	类型	说明
userID	String	用户 ID。
nickname	String?	昵称。
avatarURL	String?	头像 URL。
selfSignature	String?	个性签名。

gender	Gender?	性别。
role	UInt32?	角色。
level	UInt32?	等级。
birthday	UInt32?	生日。
allowType	AllowType?	好友验证方式。
customInfo	[String: Data]?	自定义信息。

LoginState

登录状态。

属性	类型	说明
loginStatus	LoginStatus	登录状态。
loginUserInfo	UserProfile?	登录用户信息。

Flutter
API 概述

最近更新时间：2026-04-20 17:34:06

AtomicXCore API 概览 (Flutter)

AtomicXCore SDK 是最新推出的面向视频直播、语聊房等场景的全新一代基于响应式的 API。您可以非常快速的基于这组 API 构建自己的 UI 页面。它支持房间管理、屏幕分享、成员管理、麦位控制、基础美颜等丰富功能。本页面包含 AtomicXCore SDK 的所有的功能模块，并分类展示如下。

用户登录

类名	说明
LoginStore	登录相关接口，管理用户登录、登出、用户信息设置等操作。

设备管理

类名	说明
AudioEffectStore	音效设置相关接口，管理主播的变声、混响和耳返等音效功能。
BaseBeautyStore	基础美颜相关接口，管理磨皮、美白、红润等基础美颜效果的调节和状态同步。
DeviceStore	设备相关接口，操作麦克风、摄像头等。

直播互动

类名	说明
BattleStore	直播 PK 管理相关接口，管理 PK 的创建、加入、离开等操作。
CoGuestStore	直播连麦管理相关接口，管理主播与观众之间的连麦申请、邀请、接受、拒绝等操作。
CoHostStore	直播主播连线管理相关接口，管理主播间相互连线的创建、加入、离开等操作。
LikeStore	点赞相关接口，管理直播间/语音聊天房内的点赞发送、点赞状态同步及点赞事件监听等操作。
LiveAudienceStore	直播观众相关接口，管理观众列表、权限设置等操作。

LiveListStore	直播列表相关接口，管理直播房间的创建、加入、离开等操作。
LiveSeatStore	直播麦位管理相关接口，管理麦位的上下麦、锁麦、释放麦位等操作。

弹幕消息

类名	说明
BarrageStore	弹幕相关接口，管理直播间/语音聊天房内的弹幕发送、弹幕状态同步及弹幕事件监听等操作。

礼物消息

类名	说明
GiftStore	礼物相关接口，管理直播间/语音聊天房内的礼物发送、礼物状态同步及礼物事件监听等操作。

视图组件

类名	说明
LiveCoreView	直播核心视图组件，提供直播推流和播放的视图容器，支持多人连麦、PK 等功能。

Define

最近更新时间：2026-04-20 17:34:07

简介

此模块定义了跨平台通用的基础类型，为 API 调用提供统一的错误处理和回调机制。

功能特性

- **回调机制**：提供统一的完成回调闭包类型。

数据结构

CompletionHandler

完成回调接口。
用于异步操作结果回调的接口，成功时调用 onSuccess，失败时调用 onFailure。
属性

属性	类型	说明
errorCode	int	错误码。
errorMessage	String?	错误消息描述。
isSuccess	bool	是否成功。

AudioEffectStore

最近更新时间：2026-04-20 17:34:07

简介

`AudioEffectStore` 提供了一套完整的音效管理 API，包括变声效果、混响效果和耳返功能。 通过该类，主播可以在直播过程中实时调整自己的声音效果，提升直播体验。

> **重要：**
> 使用 **shared** 单例获取 `AudioEffectStore` 实例。设置的音效在退出房间后会自动失效，下次进房需要重新设置。

> **说明：**
> 音效状态更新通过 **audioEffectState** 发布者传递。订阅它以接收有关变声、混响和耳返状态的实时更新。

> **警告：**
> 由于蓝牙耳机的硬件延迟非常高，在主播佩戴蓝牙耳机时无法开启耳返功能。请在用户界面上提示主播佩戴有线耳机。

功能特性

- **变声效果**：支持多种变声特效，如熊孩子、小女孩、大叔等。
- **混响效果**：支持多种混响特效，如 KTV、小房间、大会堂等。
- **耳返功能**：主播可在耳机中听到自己的声音，适用于唱歌场景。
- **音量控制**：支持耳返音量的精细调节。

可订阅数据

AudioEffectState 的字段描述如下：

属性名	类型	描述
audioChangerType	ValueListenable<AudioChangerType>	变声状态。
audioReverbType	ValueListenable<AudioReverbType>	混响状态。

isEarMonitorOpened	ValueListenable<bool>	耳返开启。
earMonitorVolume	ValueListenable<int>	耳返音量，取值范围0 - 100。 如果将 volume 设置成100之后感觉音量还是太小，可以将 volume 最大设置成150，但超过100的 volume 会有爆音的风险，请谨慎操作。

API 列表

函数名	描述
shared	获取单例实例。
setAudioChangerType	设置变声效果。
setAudioReverbType	设置混响效果。
setVoiceEarMonitorEnable	开启/关闭耳返。
setVoiceEarMonitorVolume	设置耳返音量。
reset	重置为默认状态。

获取实例

shared

获取单例实例。

变声设置

setAudioChangerType

设置变声效果

```
void setAudioChangerType(AudioChangerType type);
```

通过该接口您可以设置人声的变声特效。

变声特效可以作用于人声之上，通过声学算法对人声进行二次处理，以获得与原始声音所不同的音色。

版本信息

从3.5版本开始支持。

调用时机

进入房间后，且需要使用变声效果时调用。

注意事项

> **说明：**
> 设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

参数说明

参数名	类型	是否必填	描述
type	AudioChangerType	必填	变声效果类型。

混响设置

setAudioReverbType

设置混响效果

```
void setAudioReverbType(AudioReverbType type);
```

通过该接口您可以设置人声的混响效果。

混响特效可以作用于人声之上，通过声学算法对声音进行叠加处理，模拟出各种不同环境下的临场感受。

版本信息

从3.5版本开始支持。

调用时机

进入房间后，且需要使用混响效果时调用。

注意事项

> **说明：**
> 设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

参数说明

参数名	类型	是否必填	描述

type	AudioReverbType	必填	混响效果类型。

耳返设置

setVoiceEarMonitorEnable

开启/关闭耳返

```
void setVoiceEarMonitorEnable(bool enable);
```

主播开启耳返后，可以在耳机里听到麦克风采集到的自己发出的声音，该特效适用于主播唱歌的应用场景中。

版本信息

从3.5版本开始支持。

适用场景

适用于主播唱歌场景，让主播能够实时听到自己的声音以便调整演唱效果。

调用时机

进入房间后，且主播佩戴有线耳机时调用。

注意事项

> **警告：**
> 由于蓝牙耳机的硬件延迟非常高，所以在主播佩戴蓝牙耳机时无法开启此特效，请尽量在用户界面上提示主播佩戴有线耳机。

参数说明

参数名	类型	是否必填	描述
enable	bool	必填	是否开启耳返。

setVoiceEarMonitorVolume

设置耳返音量

```
void setVoiceEarMonitorVolume(int volume);
```

通过该接口您可以设置耳返特效中声音的音量大小。

版本信息

从3.5版本开始支持。

调用时机

开启耳返功能后调用。

注意事项

> **说明：**
> 如果将 volume 设置成100之后感觉音量还是太小，可以将 volume 最大设置成150，但超过100的 volume 会有爆音的风险，请谨慎操作。

参数说明

参数名	类型	是否必填	描述
volume	int	必填	耳返音量。（取值范围：0 – 100（超过100可能导致爆音））（默认值：100）。

重置

reset

重置为默认状态

```
void reset();
```

将所有音效设置重置为默认值，包括关闭变声效果、关闭混响效果、关闭耳返并重置耳返音量。
版本信息
从3.5版本开始支持。
调用时机
需要恢复默认音效设置时调用，例如退出直播间前。

数据结构

AudioChangerType

变声效果类型。

枚举值	值	说明
none	0	关闭特效。
child	1	熊孩子。
littleGirl	2	小女孩。
man	3	大叔。

heavyMetal	4	重金属。
cold	5	感冒。
foreigner	6	外语腔。
trappedBeast	7	困兽。
fatso	8	肥宅。
strongCurrent	9	强电流。
heavyMachinery	10	重机械。
ethereal	11	空灵。

AudioReverbType

混响效果类型。

枚举值	值	说明
none	0	关闭特效。
ktv	1	KTV。
smallRoom	2	小房间。
auditorium	3	大会堂。
deep	4	低沉。
loud	5	洪亮。
metallic	6	金属声。
magnetic	7	磁性。

AudioEffectState

AudioEffectStore 对外提供的音效相关状态数据。

属性	类型	说明
audioChangerType	ValueListenable<AudioChangerType>	变声状态。

audioReverbType	ValueListenable<AudioReverbType>	混响状态。
isEarMonitorOpened	ValueListenable<bool>	耳返开启。
earMonitorVolume	ValueListenable<int>	耳返音量，取值范围0 – 100。 如果将 volume 设置成100之后感觉音量还是太小，可以将 volume 最大设置成150，但超过100的 volume 会有爆音的风险，请谨慎操作。

使用示例

```
// 获取单例实例
final store = AudioEffectStore.shared;
// 订阅状态变化
store.audioEffectState.audioChangerType.addListener(() {
    print("当前变声效果：
${store.audioEffectState.audioChangerType.value}");
});
store.audioEffectState.audioReverbType.addListener(() {
    print("当前混响效果：
${store.audioEffectState.audioReverbType.value}");
});
// 设置变声效果
store.setAudioChangerType(AudioChangerType.littleGirl);
// 设置混响效果
store.setAudioReverbType(AudioReverbType.ktv);
// 开启耳返
store.setVoiceEarMonitorEnable(true);
store.setVoiceEarMonitorVolume(80);
```

BarrageStore

最近更新时间：2026-04-20 17:34:07

简介

`BarrageStore` 提供了一套完整的弹幕管理 API，包括发送文本弹幕、发送自定义弹幕和添加本地提示消息。 通过该类，可以在直播间内实现弹幕互动功能。

> **重要：**
> 使用 **create** 工厂方法创建 `BarrageStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 弹幕状态更新通过 **barrageState** 发布者传递。订阅它以接收房间内弹幕数据的实时更新。

功能特性

- **文本弹幕**：支持发送纯文本弹幕消息。
- **自定义弹幕**：支持发送自定义格式的弹幕（如带特效的弹幕）。
- **本地提示**：支持添加仅本地可见的提示消息。

可订阅数据

BarrageState 的字段描述如下：

属性名	类型	描述
messageList	ValueListenable<List<Barrage>>	当前房间的弹幕消息列表，支持实时更新并可被订阅监听。

API 列表

函数名	描述
create	创建弹幕管理实例。
addBarrageListener	添加弹幕事件监听器以接收自定义消息。
removeBarrageListener	移除弹幕事件监听器。

sendTextMessage	发送文本弹幕。
sendCustomMessage	发送自定义弹幕。
appendLocalTip	添加本地提示消息。

创建实例

create

弹幕管理核心类，用于处理直播间/语音聊天房内的弹幕相关业务逻辑。

```
static BarrageStore create(String liveID) {
  return StoreFactory.shared.getStore<BarrageStore>(liveID: liveID);
}
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播间 ID。

观察事件

addBarrageListener

添加弹幕事件监听器。

```
void addBarrageListener(BarrageListener listener);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	BarrageListener	必填	弹幕事件监听器实例。

removeBarrageListener

移除弹幕事件监听器。

```
void removeBarrageListener(BarrageListener listener);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	BarrageListener	必填	要移除的弹幕事件监听器实例。

发送弹幕

sendTextMessage

发送文本类型弹幕。

```
Future<CompletionHandler> sendTextMessage({
  required String text,
  Map<String, String>? extensionInfo,
});
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
text	String	必填	文本弹幕内容。
extensionInfo	Map<String, String>?	必填	扩展信息，可包含自定义字段（如指定弹幕颜色、字体大小等）。

sendCustomMessage

发送自定义类型弹幕。

```
Future<CompletionHandler> sendCustomMessage({
  required String businessID,
  required String data,
```

```
});
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
businessID	String	必填	业务标识 ID，用于区分不同业务场景的自定义弹幕。
data	String	必填	自定义数据内容，通常为 JSON 格式字符串，用于传递业务自定义的数据。

本地消息

appendLocalTip

添加本地提示消息（在本地添加提示或操作反馈消息，仅当前客户端可见）。

```
void appendLocalTip(Barrage message);
```

版本信息

从3.5版本开始支持。

注意事项

> **说明：**
> 该消息仅在本地显示，不会通过网络发送给其他用户。

参数说明

参数名	类型	是否必填	描述
message	Barrage	必填	本地弹幕消息（如系统提示、操作反馈等，仅当前用户可见）。

数据结构

BarrageType

弹幕类型枚举，用于区分不同的弹幕消息种类。

枚举值	值	说明

text	0	文本类型弹幕，包含纯文字内容。
custom	1	自定义类型弹幕，支持业务自定义数据格式（如带特效的弹幕、互动消息等）。

Barrage

弹幕数据模型，包含单条弹幕的完整属性信息。

属性	类型	说明
liveID	String	弹幕所属直播间/语音聊天房的唯一标识 ID。
sender	LiveUserInfo	弹幕发送者的用户信息（如用户 ID、昵称、头像等）。
sequence	int	弹幕消息的唯一序列 ID，用于消息排序和去重。
timestampInSecond	int	弹幕发送时间戳（单位：秒），用于展示发送时间顺序。
messageType	BarrageType	弹幕消息类型（文本或自定义）。
textContent	String	文本类型弹幕的消息内容，即弹幕的文本内容。
extensionInfo	Map<String, String>	弹幕扩展信息，可自定义字段（如显示样式、优先级等）。当 messageType 为 TEXT 时有效。
businessID	String	自定义类型弹幕的业务标识 ID，用于区分不同业务场景的自定义弹幕。
data	String	自定义类型弹幕的具体数据内容（通常为 JSON 格式字符串），当 messageType 为 CUSTOM 时有效。

BarrageState

弹幕状态，管理当前房间的弹幕数据状态。

属性	类型	说明

messageList	ValueListenable<List<Barrage>>	当前房间的弹幕消息列表，支持实时更新并可被订阅监听。

BarrageListener

弹幕事件监听器。

属性	类型	说明
onCustomMessageReceived	void Function(Barrage)?	收到自定义弹幕消息时触发的回调。自定义消息不会添加到 **BarrageState.messageList**，而是通过此回调进行分发。

BaseBeautyStore

最近更新时间：2026-04-20 17:34:07

简介

基础美颜功能通过简单易用的 API 实现实时美颜效果调节。`BaseBeautyStore` 提供了一套完整的接口来管理美颜效果的设置和状态订阅。

> **说明：**
> 美颜状态更新通过 **baseBeautyState** 发布者传递。订阅它以接收有关美颜效果级别的实时更新。

功能特性

- **磨皮效果**：支持0 – 9级别的磨皮效果调节。
- **美白效果**：支持0 – 9级别的美白效果调节。
- **红润效果**：支持0 – 9级别的红润效果调节。
- **状态订阅**：实时订阅美颜状态变化，同步 UI 显示与实际效果。

可订阅数据

BaseBeautyState 的字段描述如下：

属性名	类型	描述
smoothLevel	ValueListenable<double>	磨皮级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。
whitenessLevel	ValueListenable<double>	美白级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。
ruddyLevel	ValueListenable<double>	红润级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。

API 列表

函数名	描述
shared	获取单例实例。
setSmoothLevel	设置磨皮级别。
setWhitenessLeve	设置美白级别。

l	
setRuddyLevel	设置红润级别。
reset	重置为默认状态。

获取实例

shared

获取单例实例。

美颜调节

setSmoothLevel

设置磨皮级别

```
void setSmoothLevel(double smoothLevel);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
smoothLevel	double	必填	磨皮级别，取值范围 0, 9；0表示关闭，9表示效果最明显。

setWhitenessLevel

设置美白级别

```
void setWhitenessLevel(double whitenessLevel);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
whitenessLevel	double	必填	美白级别，取值范围 0, 9；0表示关闭，9表示效果最明显。

setRuddyLevel

设置红润级别

```
void setRuddyLevel(double ruddyLevel);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
ruddyLevel	double	必填	红润级别，取值范围 0, 9；0表示关闭，9表示效果最明显。

reset

将所有美颜参数（磨皮、美白、红润）重置为默认关闭状态（值为0）。

```
void reset();
```

版本信息
从3.5版本开始支持。

数据结构

BaseBeautyState

基础美颜状态，管理磨皮、美白、红润等美颜效果的级别数据。支持订阅以同步 UI 显示与实际效果。

属性	类型	说明
smoothLevel	ValueListenable<double>	磨皮级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。
whitenessLevel	ValueListenable<double>	美白级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。
ruddyLevel	ValueListenable<double>	红润级别，取值范围 0 – 9；0表示关闭，9表示效果最明显。

使用示例

```
// 获取单例实例
final store = BaseBeautyStore.shared;
// 定义监听器
late final VoidCallback smoothLevelListener = _onSmoothLevelChanged;
late final VoidCallback whitenessLevelListener =
_onWhitenessLevelChanged;
late final VoidCallback ruddyLevelListener = _onRuddyLevelChanged;
void _onSmoothLevelChanged() {
    print('磨皮级别: ${store.baseBeautyState.smoothLevel.value}');
}
void _onWhitenessLevelChanged() {
    print('美白级别: ${store.baseBeautyState.whitenessLevel.value}');
}
void _onRuddyLevelChanged() {
    print('红润级别: ${store.baseBeautyState.ruddyLevel.value}');
}
// 订阅状态变化
store.baseBeautyState.smoothLevel.addListener(smoothLevelListener);
store.baseBeautyState.whitenessLevel.addListener(whitenessLevelListener)
;
store.baseBeautyState.ruddyLevel.addListener(ruddyLevelListener);
// 设置美颜效果
store.setSmoothLevel(5.0);
store.setWhitenessLevel(3.0);
store.setRuddyLevel(2.0);
// 重置所有美颜效果
store.reset();
// 取消订阅
store.baseBeautyState.smoothLevel.removeListener(smoothLevelListener);
store.baseBeautyState.whitenessLevel.removeListener(whitenessLevelListen
er);
store.baseBeautyState.ruddyLevel.removeListener(ruddyLevelListener);
```

BattleStore

最近更新时间：2026-04-20 17:34:07

简介

PK 功能实现主播之间的实时互动对战。`BattleStore` 提供了一套全面的 API 来管理整个 PK 生命周期。

> **重要：**
> 请始终使用工厂方法 **create(liveID:)** 并提供有效的直播间 ID 来创建 `BattleStore` 实例。不要尝试直接初始化。

> **说明：**
> PK 状态更新通过 **battleState** 发布者传递。订阅它以接收有关 PK 信息、参与用户和分数的实时更新。

> **警告：**
> 如果 PK 请求在指定的超时时间内未收到响应，将触发超时事件。请始终在 UI 中处理超时场景。

功能特性

- **PK 请求管理**：主播可以发起 PK 请求，被邀请方可以接受或拒绝。
- **状态管理**：实时跟踪 PK 信息、参与用户和分数。
- **事件驱动架构**：提供完整的 PK 事件回调。
- **超时处理**：为 PK 请求提供内置超时机制。

可订阅数据

BattleState 的字段描述如下：

属性名	类型	描述
currentBattleInfo	ValueListenable<BattleInfo?>	当前 PK 信息。
battleUsers	ValueListenable<List<SeatUserInfo>>	PK 用户列表。
battleScore	ValueListenable<Map<String, int>>	PK 分数的映射。

API 列表

函数名	描述
BattleStore.create	创建 BattleStore 实例。
addBattleListener	新增 PK 事件回调。
removeBattleListener	移除 PK 事件回调。
requestBattle	发起 PK 请求。
cancelBattleRequest	取消 PK 请求。
acceptBattle	接受 PK 请求。
rejectBattle	拒绝 PK 请求。
exitBattle	退出 PK。

创建实例

BattleStore.create

创建 BattleStore 实例。

观察状态和事件

addBattleListener

新增 PK 事件回调

```
void addBattleListener(BattleListener listener);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	BattleListener	必填	监听器。

removeBattleListener

移除 PK 事件回调

```
void removeBattleListener(BattleListener listener);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	BattleListener	必填	监听器。

PK 操作

requestBattle

发起 PK 请求

```
Future<BattleRequestCompletionHandler> requestBattle({
    required BattleConfig config,
    required List<String> userIDList,
    required int timeout,
});
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
config	BattleConfig	必填	PK 配置。
userIDList	List<String>	必填	需要进入 PK 的用户 ID 列表。
timeout	int	必填	请求超时时间。

cancelBattleRequest

取消 PK 请求

```
Future<CompletionHandler> cancelBattleRequest({
```

```
    required String battleID,
    required List<String> userIDList,
});
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
battleID	String	必填	PK ID。
userIDList	List<String>	必填	用户 ID 列表。

acceptBattle

接受 PK 请求

```
Future<CompletionHandler> acceptBattle(String battleID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
battleID	String	必填	PK ID。

rejectBattle

拒绝 PK 请求

```
Future<CompletionHandler> rejectBattle(String battleID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
battleID	String	必填	PK ID。

exitBattle

退出 PK

```
Future<CompletionHandler> exitBattle(String battleID);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
battleID	String	必填	PK ID。

数据结构

BattleEndedReason

正在 PK 中用户收到 PK 结束的原因。

枚举值	值	说明
timeOver	0	PK 倒计时结束。
allMemberExit	1	所有 PK 成员退出。

BattleListener

收到 PK 相关的回调事件。
方法

方法名	说明
onBattleStarted	当 PK 正式开始时触发此回调，通知所有参与者 PK 已经开始。
onBattleEnded	当 PK 结束时触发此回调。
onUserJoinBattle	当有用户加入 PK 时触发此回调。
onUserExitBattle	当有用户退出 PK 时触发此回调。
onBattleRequestReceived	当收到 PK 请求时触发此回调。
onBattleRequestC	当 PK 请求被取消时触发此回调。

ancelled	
onBattleRequestTimeout	当 PK 请求超时时触发此回调。
onBattleRequestAccept	当 PK 请求被接受时触发此回调。
onBattleRequestReject	当 PK 请求被拒绝时触发此回调。

BattleConfig

发送 PK 请求时，设置的 PK 配置信息。

属性	类型	说明
duration	int	PK 持续时间（单位：秒）。
needResponse	bool	被邀请用户是否需要回复同意/拒绝。
extensionInfo	String	扩展信息。

BattleInfo

PK 信息。

属性	类型	说明
battleID	String	PK ID。
config	BattleConfig	发送 PK 请求时，设置的 PK 配置信息。
startTime	int	PK 开始标记时间戳（单位：秒）。
endTime	int	PK 结束标记时间戳（单位：秒）。

BattleState

BattleStore 对外提供的 PK 相关状态数据。

属性	类型	说明
currentBattleInfo	ValueListenable<BattleInfo?>	当前 PK 信息。

battleUsers	ValueListenable<List<SeatUserInfo>>	PK 用户列表。
battleScore	ValueListenable<Map<String, int>>	PK 分数的映射。

BattleRequestCompletionHandler

PK 请求完成处理器。

Dart 中 PK 请求的完成处理器，包含 PK 请求的结果。

属性

属性	类型	说明
battleInfo	BattleInfo?	成功时返回的 PK 信息。
resultMap	Map<String, int>?	PK 请求的响应结果映射。

使用示例

```
// 创建 store 实例
final store = BattleStore.create('live_room_123');
// 定义监听器
late final VoidCallback battleInfoListener = _onBattleInfoChanged;
late final VoidCallback battleUsersListener = _onBattleUsersChanged;
void _onBattleInfoChanged() {
    final battleInfo = store.battleState.currentBattleInfo.value;
    if (battleInfo != null) {
        print('当前 PK ID: ${battleInfo.battleID}');
    }
}
void _onBattleUsersChanged() {
    print('PK 用户数: ${store.battleState.battleUsers.value.length}');
}
// 订阅状态变化
store.battleState.currentBattleInfo.addListener(battleInfoListener);
store.battleState.battleUsers.addListener(battleUsersListener);
// 添加 PK 事件监听器
final battleListener = BattleListener(
    onBattleStarted: (battleInfo, inviter, invitees) {
        print('PK 开始，发起者: ${inviter.userName}');
```

```
    },
    onBattleEnded: (battleInfo, reason) {
        print('PK 结束，原因: $reason');
    },
);
store.addBattleListener(battleListener);
// 发起 PK 请求
final config = BattleConfig(duration: 300, needResponse: true);
final result = await store.requestBattle(
    config: config,
    userIDList: ['user_456'],
    timeout: 30,
);
if (result.code == 0) {
    print('PK 请求成功: ${result.battleInfo?.battleID}');
} else {
    print('PK 请求失败: ${result.message}');
}
// 取消订阅
store.battleState.currentBattleInfo.removeListener(battleInfoListener);
store.battleState.battleUsers.removeListener(battleUsersListener);
store.removeBattleListener(battleListener);
```

CoGuestStore

最近更新时间：2026-04-20 17:34:07

简介

连麦功能通过基于麦位的系统实现主播和观众成员之间的实时互动。`CoGuestStore` 提供了一套全面的 API 来管理整个连麦生命周期。

> **重要：**
> 请始终使用工厂方法 **CoGuestStore.create** 并提供有效的直播间 ID 来创建 `CoGuestStore` 实例。不要尝试直接初始化。

> **说明：**
> 连麦状态更新通过 **coGuestState** 发布者传递。订阅它以接收有关已连接用户、邀请和申请的实时更新。

> **警告：**
> 如果连麦请求在指定的超时时间内未收到响应，将触发带有 `NoResponseReason.timeout` 的事件。请始终在 UI 中处理超时场景。

功能特性

- **双向邀请**：主播可以邀请观众成员，观众成员也可以申请加入。
- **状态管理**：实时跟踪已连接用户、邀请和申请。
- **事件驱动架构**：为主播和观众角色提供独立的事件流。
- **超时处理**：为邀请和申请提供内置超时机制。

可订阅数据

CoGuestState 的字段描述如下：

属性名	类型	描述
connected	ValueListenable<List<SeatUserInfo>>	已经在麦位上的用户列表。
invitees	ValueListenable<List<LiveUserInfo>>	主播发出邀请的用户列表。
applicants	ValueListenable<List<LiveUserInfo>>	主播收到申请连麦的用户列表。

candidates	ValueListenable<List<LiveUserInfo>>	候选连麦的用户列表。

API 列表

函数名	描述
CoGuestStore.create	创建对象实例。
addHostListener	主播端事件回调。
removeHostListener	主播端事件回调。
addGuestListener	观众端事件回调。
removeGuestListener	观众端事件回调。
applyForSeat	观众申请连麦。
cancelApplication	观众取消申请。
acceptApplication	主播接受申请。
rejectApplication	主播拒绝申请。
inviteToSeat	主播邀请观众连麦。
cancelInvitation	主播取消邀请。
acceptInvitation	观众接受邀请。
rejectInvitation	观众拒绝邀请。
disconnect	结束连麦会话。

创建实例

CoGuestStore.create

创建对象实例。

观察状态和事件

addHostListener

添加主播侧事件回调监听器

```
void addHostListener(HostListener listener);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	HostListener	必填	监听器。

removeHostListener

移除主播侧事件回调监听器

```
void removeHostListener(HostListener listener);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	HostListener	必填	监听器。

addGuestListener

添加观众侧事件回调监听器

```
void addGuestListener(GuestListener listener);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	GuestListener	必填	监听器。

removeGuestListener

移除观众侧事件回调监听器

```
void removeGuestListener(GuestListener listener);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	GuestListener	必填	监听器。

观众操作

applyForSeat

申请上麦

```
Future<CompletionHandler> applyForSeat({
    required int seatIndex,
    required int timeout,
    String? extraInfo,
});
```

以观众身份请求加入连麦会话。

调用此方法后，会向直播间内的所有主播发送连麦请求。请求将保持活动状态，直到：

- 主播通过 acceptApplication 接受
- 主播通过 rejectApplication 拒绝
- 超时时间到期
- 您通过 cancelApplication 取消

版本信息

从3.5版本开始支持。

注意事项

> **说明：**
> 如果在超时时间内没有主播响应，将触发带有 NoResponseReason.timeout 的 {ref2} 事件。

参数说明

参数名	类型	是否必填	描述
seatIndex	int	必填	麦位索引，-1表示自动分配麦位。
timeout	int	必填	超时时间（单位：秒）。
extraInfo	String?	必填	额外信息。

cancelApplication

取消上麦申请

```
Future<CompletionHandler> cancelApplication();
```

取消之前发送的连麦申请。调用此方法后，所有主播将收到申请取消的通知。

版本信息

从3.5版本开始支持。

注意事项

> **说明：**
> 如果申请已被主播处理，取消操作可能无效。

acceptApplication

接受上麦申请

```
Future<CompletionHandler> acceptApplication(String userID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。

rejectApplication

拒绝上麦申请

```
Future<CompletionHandler> rejectApplication(String userID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。

主播操作

inviteToSeat

邀请观众上麦

```
Future<CompletionHandler> inviteToSeat({
    required String inviteeID,
    required int seatIndex,
    required int timeout,
    String? extraInfo,
});
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
inviteeID	String	必填	被邀请用户 ID。
seatIndex	int	必填	麦位索引，-1表示自动分配麦位。
timeout	int	必填	超时时间（单位：秒）。
extraInfo	String?	必填	额外信息。

cancelInvitation

取消上麦邀请

```
Future<CompletionHandler> cancelInvitation(String inviteeID);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
inviteeID	String	必填	被邀请用户 ID。

acceptInvitation

接受上麦邀请

```
Future<CompletionHandler> acceptInvitation(String inviterID);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
inviterID	String	必填	邀请者用户 ID。

rejectInvitation

拒绝上麦邀请

```
Future<CompletionHandler> rejectInvitation(String inviterID);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
inviterID	String	必填	邀请者用户 ID。

连接控制

disconnect

结束连麦会话。

数据结构

NoResponseReason

主播发出的连麦邀请或者观众发起连麦请求的无响应原因。

枚举值	值	说明
timeout	0	请求超时。
alreadySeated	1	用户已在麦位上。

HostListener

主播侧收到的回调事件。

方法

方法名	说明
onGuestApplicationReceived	当观众申请连麦时触发此回调。
onGuestApplicationCancelled	当观众取消连麦申请时触发此回调。
onGuestApplicationProcessedByOtherHost	当观众的连麦申请被其他主播处理时触发此回调。
onHostInvitationResponded	当主播发出的连麦邀请收到观众响应时触发此回调。
onHostInvitationNoResponse	当主播发出的连麦邀请无响应时触发此回调。

GuestListener

观众侧收到的回调事件。

方法

方法名	说明
onHostInvitationReceived	当收到主播的连麦邀请时触发此回调。
onHostInvitationCancelled	当主播取消连麦邀请时触发此回调。

onGuestApplicationResponded	当观众的连麦申请收到主播响应时触发此回调。
onGuestApplicationNoResponse	当观众的连麦申请无响应时触发此回调。
onKickedOffSeat	当观众被主播踢下麦位时触发此回调。

CoGuestState

CoGuestStore 对外提供的连麦相关状态数据。

属性	类型	说明
connected	ValueListenable<List<SeatUserInfo>>	已经在麦位上的用户列表。
invitees	ValueListenable<List<LiveUserInfo>>	主播发出邀请的用户列表。
applicants	ValueListenable<List<LiveUserInfo>>	主播收到申请连麦的用户列表。
candidates	ValueListenable<List<LiveUserInfo>>	候选连麦的用户列表。

使用示例

```
// 创建 store 实例
final store = CoGuestStore.create('live_room_123');
// 定义监听器
late final VoidCallback connectedListener = _onConnectedChanged;
late final VoidCallback applicantsListener = _onApplicantsChanged;
void _onConnectedChanged() {
    print('已连接用户: ${store.coGuestState.connected.value.length}');
}
void _onApplicantsChanged() {
    print('待处理申请: ${store.coGuestState.applicants.value.length}');
}
// 订阅状态变化
store.coGuestState.connected.addListener(connectedListener);
store.coGuestState.applicants.addListener(applicantsListener);
// 添加主播事件监听器（主播使用）
```

```
final hostListener = HostListener(
    onGuestApplicationReceived: (guestUser) {
        print('收到来自 ${guestUser.userName} 的申请');
        // 显示接受/拒绝的 UI
    },
    onHostInvitationResponded: (isAccept, guestUser) {
        print('观众 ${guestUser.userName} ${isAccept ? "接受了" : "拒绝
了"}');
    },
);
store.addHostListener(hostListener);
// 主播：接受申请
final result = await store.acceptApplication('user_456');
if (result.code == 0) {
    print('申请接受成功');
}
// 完成后取消订阅
store.coGuestState.connected.removeListener(connectedListener);
store.coGuestState.applicants.removeListener(applicantsListener);
store.removeHostListener(hostListener);
```

CoHostStore

最近更新时间：2026-04-20 17:34:07

简介

跨房连线功能允许不同直播间的主播进行实时互动。`CoHostStore` 提供了一套全面的 API 来管理整个跨房连线生命周期。

> **重要：**
> 请始终使用工厂方法 **CoHostStore.create** 并提供有效的直播间 ID 来创建 `CoHostStore` 实例。不要尝试直接初始化。

> **说明：**
> 连线状态更新通过 **coHostState** 发布者传递。订阅它以接收有关连线状态、已连接主播、邀请和申请的实时更新。

> **警告：**
> 如果连线请求在指定的超时时间内未收到响应，将触发超时事件。请始终在 UI 中处理超时场景。

功能特性

- **双向连线**：主播可以向其他主播发起连线请求，也可以接收其他主播的连线请求。
- **状态管理**：实时跟踪连线状态、已连接主播、邀请列表和申请者。
- **事件驱动架构**：提供连线事件流用于监听各种连线状态变化。
- **布局模板**：支持多种连线布局模板，如动态网格布局和1对6布局。

可订阅数据

CoHostState 的字段描述如下：

属性名	类型	描述
coHostStatus	ValueListenable<CoHostStatus>	跨房连线的实时状态。
connected	ValueListenable<List<SeatUserInfo>>	正在和当前直播间连线的主播列表。
invitees	ValueListenable<List<SeatUserInfo>>	向其他直播间发出请求的主播列表。

applicant	ValueListenable<SeatUserInfo?>	向当前直播间发起连线请求的主播。
candidatesCursor	ValueListenable<String>	推荐用户列表游标。
candidates	ValueListenable<List<SeatUserInfo>>	推荐用户列表。

API 列表

函数名	描述
CoHostStore.create	创建对象实例。
addCoHostListener	连线事件回调。
removeCoHostListener	连线事件回调。
requestHostConnection	发起连线请求。
cancelHostConnection	取消连线请求。
acceptHostConnection	接受连线请求。
rejectHostConnection	拒绝连线请求。
exitHostConnection	退出连线。
muteRemoteHostAudio	禁音/取消禁音远端主播的音频。
getCoHostCandidates	获取推荐主播列表。

创建实例

CoHostStore.create

创建对象实例。

观察状态和事件

addCoHostListener

添加连线回调监听器

```
void addCoHostListener(CoHostListener listener);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	CoHostListener	必填	监听器。

removeCoHostListener

移除连线回调监听器

```
void removeCoHostListener(CoHostListener listener);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	CoHostListener	必填	监听器。

连线操作

requestHostConnection

发起主播连线请求

```
Future<CompletionHandler> requestHostConnection({
    required String targetHostLiveID,
    required CoHostLayoutTemplate layoutTemplate,
    required int timeout,
    String extraInfo = '',
```

```
});
```

向目标主播发起跨房连线请求。

调用此方法后，会向目标主播发送连线请求。请求将保持活动状态，直到：
- 目标主播通过 acceptHostConnection(fromHostLiveID:completion:) 接受
- 目标主播通过 rejectHostConnection(fromHostLiveID:completion:) 拒绝
- 超时时间到期
- 您通过 cancelHostConnection(toHostLiveID:completion:) 取消

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
targetHostLiveID	String	必填	目标主播的直播间 ID。
layoutTemplate	CoHostLayoutTemplate	必填	连线布局模板。
timeout	int	必填	请求超时时间（单位：秒）。
extraInfo	String?	必填	扩展信息。

cancelHostConnection

取消主播连线请求

```
Future<CompletionHandler> cancelHostConnection(String toHostLiveID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
toHostLiveID	String	必填	目标主播的直播间 ID。

acceptHostConnection

接受主播连线请求

```
Future<CompletionHandler> acceptHostConnection(String fromHostLiveID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
fromHostLiveID	String	必填	发起连线请求的主播直播间 ID。

rejectHostConnection

拒绝主播连线请求

```
Future<CompletionHandler> rejectHostConnection(String fromHostLiveID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
fromHostLiveID	String	必填	发起连线请求的主播直播间 ID。

exitHostConnection

退出主播连线

```
Future<CompletionHandler> exitHostConnection();
```

版本信息

从3.5版本开始支持。

muteRemoteHostAudio

禁音/取消禁音远端主播的音频

```
Future<CompletionHandler> muteRemoteHostAudio({
    required String liveID,
    required bool isMuted,
});
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
liveID	String	必填	远端主播的直播间 ID。
isMuted	bool	必填	是否禁音远端主播的音频。`true` 表示禁音，`false` 表示取消禁音。

getCoHostCandidates

获取可以与当前主播连线的推荐主播列表

```
Future<CompletionHandler> getCoHostCandidates(String cursor);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
cursor	String	必填	游标。

数据结构

CoHostStatus

当前用户的跨房连线状态。

枚举值	值	说明
connected	0	和其他主播正在连线中。
disconnected	1	没有和其他主播连线。

CoHostLayoutTemplate

连线布局模板。

枚举值	值	说明
hostVoiceConnecti	2	语聊房连线布局。

on		
hostDynamicGrid	600	主播动态网格布局。
hostDynamic1v6	601	主播动态1对6布局。
hostVideoLeftFocus9Seats	602	主播视频左焦点9席位布局。
hostVideoUniformGrid9Seats	603	主播视频均匀网格9席位布局。

CoHostListener

连线请求的回调事件。

方法

方法名	说明
onCoHostRequestReceived	当收到连线请求时触发此回调。
onCoHostRequestCancelled	当连线请求被取消时触发此回调。
onCoHostRequestAccepted	当连线请求被接受时触发此回调。
onCoHostRequestRejected	当连线请求被拒绝时触发此回调。
onCoHostRequestTimeout	当连线请求超时时触发此回调。
onCoHostUserJoined	当用户加入连线时触发此回调。
onCoHostUserLeft	当用户离开连线时触发此回调。

CoHostState

CoHostStore 对外提供的跨房连线相关状态数据。

属性	类型	说明
coHostStatus	ValueListenable<CoHostStatus>	跨房连线的实时状态。

connected	ValueListenable<List<SeatUserInfo>>	正在和当前直播间连线的主播列表。
invitees	ValueListenable<List<SeatUserInfo>>	向其他直播间发出请求的主播列表。
applicant	ValueListenable<SeatUserInfo?>	向当前直播间发起连线请求的主播。
candidatesCursor	ValueListenable<String>	推荐用户列表游标。
candidates	ValueListenable<List<SeatUserInfo>>	推荐用户列表。

使用示例

```
// 创建 store 实例
final store = CoHostStore.create('live_room_123');
// 定义监听器
late final VoidCallback statusListener = _onStatusChanged;
late final VoidCallback connectedListener = _onConnectedChanged;
void _onStatusChanged() {
    print('连线状态: ${store.coHostState.coHostStatus.value}');
}
void _onConnectedChanged() {
    print('已连接主播: ${store.coHostState.connected.value.length}');
}
// 订阅状态变化
store.coHostState.coHostStatus.addListener(statusListener);
store.coHostState.connected.addListener(connectedListener);
// 添加连线事件监听器
final coHostListener = CoHostListener(
    onCoHostRequestReceived: (inviter, extensionInfo) {
        print('收到来自 ${inviter.userName} 的连线请求');
        // 显示接受/拒绝的 UI
    },
    onCoHostRequestAccepted: (invitee) {
        print('连线请求被 ${invitee.userName} 接受');
    },
    onCoHostUserJoined: (userInfo) {
        print('主播 ${userInfo.userName} 加入连线');
```

```
    },
);
store.addCoHostListener(coHostListener);
// 发起连线请求
final result = await store.requestHostConnection(
    targetHostLiveID: 'target_live_id',
    layoutTemplate: CoHostLayoutTemplate.hostDynamicGrid,
    timeout: 30,
    extraInfo: '',
);
if (result.code == 0) {
    print('连线请求发送成功');
}
// 完成后取消订阅
store.coHostState.coHostStatus.removeListener(statusListener);
store.coHostState.connected.removeListener(connectedListener);
store.removeCoHostListener(coHostListener);
```

DeviceStore

最近更新时间：2026-04-20 17:34:07

简介

`DeviceStore` 提供了一套全面的 API 来管理音视频设备，包括麦克风、摄像头和屏幕分享等功能。

> **重要：**
> 请使用 **shared** 单例获取 `DeviceStore` 实例。不要尝试直接初始化。

> **说明：**
> 设备状态更新通过 **state** 发布者传递。订阅它以接收有关麦克风、摄像头、网络等状态的实时更新。

功能特性

- **麦克风管理**：打开/关闭麦克风，设置采集音量和输出音量。
- **摄像头管理**：打开/关闭摄像头，切换前后摄像头，设置镜像和视频质量。
- **音频路由**：切换扬声器和听筒。
- **屏幕分享**：开启和关闭屏幕分享功能。
- **网络状态**：实时监控网络质量信息。

可订阅数据

DeviceState 的字段描述如下：

属性名	类型	描述
microphoneStatus	ValueListenable<DeviceStatus>	麦克风状态。
microphoneLastError	ValueListenable<DeviceError>	麦克风错误，用于出现报错时提取错误信息。
captureVolume	ValueListenable<int>	采集音量，取值范围 0, 100。
currentMicVolume	ValueListenable<int>	当前用户实际输出音量。
outputVolume	ValueListenable<int>	最大输出音量，取值范围 0, 100。
cameraStatus	ValueListenable<DeviceStatus>	摄像头状态。

cameraLastError	ValueListenable<DeviceError>	摄像头错误，用于出现报错时提取错误信息。
isFrontCamera	ValueListenable<bool>	是否为前置摄像头。
localMirrorType	ValueListenable<MirrorType>	镜像状态。
localVideoQuality	ValueListenable<VideoQuality>	本地视频质量。
currentAudioRoute	ValueListenable<AudioRoute>	当前音频路由位置。
screenStatus	ValueListenable<DeviceStatus>	屏幕分享状态。
networkInfo	ValueListenable<NetworkInfo>	网络信息。
networkType	NetworkType	当前网络类型。

API 列表

函数名	描述
shared	单例对象。
openLocalMicrophone	打开本地麦克风。
closeLocalMicrophone	关闭本地麦克风。
setCaptureVolume	设置采集音量。
setOutputVolume	设置输出音量。
setAudioRoute	设置音频路由。
openLocalCamera	打开本地摄像头。
closeLocalCamera	关闭本地摄像头。
switchCamera	切换摄像头。

switchMirror	切换镜像状态。
updateVideoQuality	更新视频质量。
startScreenShare	开启屏幕分享。
stopScreenShare	关闭屏幕分享。
reset	重置为默认状态。

获取实例

shared

单例对象。

麦克风操作

openLocalMicrophone

打开本地麦克风

```
Future<CompletionHandler> openLocalMicrophone();
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
completion	Completion Closure?	必填	操作是否成功。

closeLocalMicrophone

关闭本地麦克风

```
void closeLocalMicrophone();
```

版本信息
从3.5版本开始支持。

setCaptureVolume

设置采集音量

```
void setCaptureVolume(int volume);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
volume	int	必填	采集音量，取值范围 0, 100。

setOutputVolume

设置最大输出音量

```
void setOutputVolume(int volume);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
volume	int	必填	最大音量，取值范围 0, 100。

音频路由

setAudioRoute

设置音频路由

```
void setAudioRoute(AudioRoute route);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
route	AudioRoute	必填	路由位置。

摄像头操作

openLocalCamera

打开本地摄像头

```
Future<CompletionHandler> openLocalCamera(bool isFront);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
isFront	bool	必填	是否前置摄像头。
completion	Completion Closure?	必填	操作是否成功。

closeLocalCamera

关闭本地摄像头

```
void closeLocalCamera();
```

版本信息

从3.5版本开始支持。

switchCamera

切换摄像头

```
void switchCamera(bool isFront);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
isFront	bool	必填	是否前置摄像头。

switchMirror

切换镜像状态

```
void switchMirror(MirrorType mirrorType);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
mirrorType	MirrorType	必填	镜像状态。

updateVideoQuality

更新视频质量

```
void updateVideoQuality(VideoQuality quality);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
quality	VideoQuality	必填	视频质量。

屏幕分享

startScreenShare

开启屏幕分享

```
void startScreenShare();
```

版本信息

从3.5版本开始支持。

stopScreenShare

关闭屏幕采集

```
void stopScreenShare();
```

版本信息

从3.5版本开始支持。

重置

reset

重置为默认状态

```
void reset();
```

版本信息

从3.5版本开始支持。

数据结构

DeviceType

设备类型。

枚举值	值	说明
microphone	0	麦克风类型。
camera	1	摄像头类型。
screenShare	2	屏幕分享类型。

DeviceError

设备相关错误码。

枚举值	值	说明
noError	0	操作成功。
noDeviceDetected	1	未检测到设备。
noSystemPermission	2	没有系统权限。
notSupportCapture	3	不支持采集。

occupiedError	4	设备已占用。
unknownError	5	未知错误。

DeviceStatus

设备开启状态。

枚举值	值	说明
off	0	关闭。
on	1	开启。

AudioRoute

音频路由。

枚举值	值	说明
speakerphone	0	扬声器，使用扬声器播放（即"免提"），扬声器位于手机底部，声音偏大，适合外放音乐。
earpiece	1	听筒，使用听筒播放，听筒位于手机顶部，声音偏小，适合需要保护隐私的通话场景。

VideoQuality

视频质量。

枚举值	值	说明
quality360P	1	360P。
quality540P	2	540P。
quality720P	3	720P。
quality1080P	4	1080P。

NetworkQuality

网络质量。

枚举值	值	说明
unknown	0	未知网络。
excellent	1	极佳。
good	2	良好。
poor	3	较差。
bad	4	差。
veryBad	5	极差。
down	6	中断。

MirrorType

摄像头镜像状态。

枚举值	值	说明
auto	0	自动，前置摄像头镜像，后置摄像头不镜像。
enable	1	前后摄像头均镜像。
disable	2	前后摄像头均不镜像。

DeviceFocusOwner

设备焦点。

枚举值	说明
call	语音通话场景。
live	直播场景。
room	房间场景。
none	未设置。

NetworkInfo

网络信息。

属性	类型	说明
userID	String	用户唯一 ID。
quality	NetworkQuality	网络质量。
upLoss	int	上行丢包率，取值范围 0, 100。
downLoss	int	下行丢包率，取值范围 0, 100。
delay	int	延迟（单位：毫秒）。

DeviceState

设备状态。

属性	类型	说明
microphoneStatus	ValueListenable<DeviceStatus>	麦克风状态。
microphoneLastError	ValueListenable<DeviceError>	麦克风错误，用于出现报错时提取错误信息。
captureVolume	ValueListenable<int>	采集音量，取值范围 0, 100。
currentMicVolume	ValueListenable<int>	当前用户实际输出音量。
outputVolume	ValueListenable<int>	最大输出音量，取值范围 0, 100。
cameraStatus	ValueListenable<DeviceStatus>	摄像头状态。
cameraLastError	ValueListenable<DeviceError>	摄像头错误，用于出现报错时提取错误信息。
isFrontCamera	ValueListenable<bool>	是否为前置摄像头。
localMirrorType	ValueListenable<MirrorType>	镜像状态。
localVideoQuality	ValueListenable<VideoQuality>	本地视频质量。
currentAudioRoute	ValueListenable<AudioRoute>	当前音频路由位置。

screenStatus	ValueListenable<DeviceStatus>	屏幕分享状态。
networkInfo	ValueListenable<NetworkInfo>	网络信息。
networkType	NetworkType	当前网络类型。

使用示例

```
// 获取单例实例
let store = DeviceStore.shared
// 订阅状态变化
store.state.subscribe { state in
    print("麦克风状态: \(state.microphoneStatus)")
    print("摄像头状态: \(state.cameraStatus)")
    print("网络质量: \(state.networkInfo.quality)")
}
// 打开麦克风
store.openLocalMicrophone { code, message in
    if code == 0 {
        print("麦克风打开成功")
    }
}
// 打开前置摄像头
store.openLocalCamera(isFront: true) { code, message in
    if code == 0 {
        print("摄像头打开成功")
    }
}
```

GiftStore

最近更新时间：2026-04-20 17:34:07

简介

`GiftStore` 提供了一套完整的礼物管理 API，包括发送礼物、刷新礼物列表、设置语言和监听礼物事件。 通过该类，可以在直播间内实现礼物互动功能。

> **重要：**
> 使用 GiftStore.create 工厂方法创建 `GiftStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 礼物状态更新通过 giftState 发布者传递。订阅它以接收房间内礼物数据的实时更新。

功能特性

- **礼物发送**：支持向当前房间发送指定礼物。
- **礼物列表**：获取和刷新当前房间可用的礼物列表。
- **语言设置**：设置礼物信息的展示语言。
- **事件监听**：监听礼物接收事件。

可订阅数据

GiftState 的字段描述如下：

属性名	类型	描述
usableGifts	ValueListenable<List<GiftCategory>>	当前房间可用的所有礼物分类及礼物列表。

API 列表

函数名	描述
GiftStore.create	创建礼物管理实例。
addGiftListener	礼物事件回调。
removeGiftListener	礼物事件回调。

sendGift	发送礼物。
refreshUsableGifts	刷新可用礼物列表。
setLanguage	设置展示语言。

创建实例

GiftStore.create

创建礼物管理实例。

观察状态和事件

addGiftListener

礼物事件回调

removeGiftListener

礼物事件回调

礼物操作

sendGift

向当前房间发送指定礼物

```
Future<CompletionHandler> sendGift({
  required String giftID,
  required int count,
});
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
giftID	String	必填	要发送的礼物唯一标识 ID。
count	int	必填	单次发送的礼物数量。

refreshUsableGifts

手动刷新当前房间的可用礼物列表。

```
Future<CompletionHandler> refreshUsableGifts();
```

版本信息

从3.5版本开始支持。

setLanguage

设置礼物信息的展示语言。

```
void setLanguage(String language);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
language	String	必填	语言代码（"zh-CN" 表示中文，"en" 表示英文），设置完成展示界面刷新后礼物名称、描述等会同步更新为对应语言。

数据结构

Gift

礼物数据模型，包含单个礼物的完整属性信息。

属性	类型	说明
giftID	String	礼物 ID。
name	String	礼物名称。
desc	String	礼物描述。
iconURL	String	礼物图标图片的网络 URL，用于加载礼物缩略图。
resourceURL	String	礼物动效资源文件的网络 URL，用于加载礼物展示动效。
level	int	礼物等级，用于区分礼物稀有度或价值层

		级。
coins	int	礼物价格（金币）。
extensionInfo	Map<String, String>	礼物扩展信息，可自定义字段（如特效类型、赠送限制等）。

GiftCategory

礼物分类。

属性	类型	说明
categoryID	String	分类唯一标识 ID，用于区分不同礼物分类。
name	String	分类展示名称，用于 UI 分类显示（如 "热门礼物"，"高级礼物"）。
desc	String	分类描述信息，用于说明该分类的特点。
extensionInfo	Map<String, String>	分类扩展信息，包含自定义字段（如排序权重、显示样式等）。
giftList	List< Gift >	当前分类下的所有礼物列表。

GiftState

礼物状态，管理当前房间的礼物数据状态，支持实时更新并可被订阅监听。

属性	类型	说明
usableGifts	ValueListenable<List< GiftCategory >>	当前房间可用的所有礼物分类及礼物列表。

GiftListener

礼物事件，用于接收直播间/语音聊天房内的礼物动态。

方法

onReceiveGift: 收到新礼物消息的事件回调。当直播间/语音聊天房内有其他观众发送礼物时，会触发该事件并返回相关信息。

```
void Function(String liveID, Gift gift, int count, LiveUserInfo sender)?
onReceiveGift;
GiftListener({this.onReceiveGift});
```

参数名	类型	说明
liveID	String	直播间 ID。
gift	Gift	礼物信息。
count	int	礼物数量。
sender	LiveUserInfo	礼物发送者信息。

LikeStore

最近更新时间：2026-04-20 17:34:07

简介

`LikeStore` 提供了一套完整的点赞管理 API，包括发送点赞、监听点赞事件和获取点赞状态。通过该类，可以在直播间内实现点赞互动功能。

> **重要：**
> 使用 **create** 工厂方法创建 `LikeStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 点赞状态更新通过 **likeState** 发布者传递。订阅它以接收房间内点赞数据的实时更新。

功能特性

- **点赞发送**：支持向当前房间发送点赞。
- **点赞状态**：获取当前房间的累计点赞数。
- **事件监听**：监听点赞接收事件。

数据结构

LikeState

点赞状态，用于展示和订阅直播间/语音聊天房的点赞信息。

属性	类型	说明
totalLikeCount	ValueListenable<int>	当前直播间/语音聊天房的累计总点赞数，支持实时更新并可被订阅监听。

LikeListener

点赞事件，用于接收直播间/语音聊天房内的点赞动态。
此监听器用于接收直播间/语音聊天房内的点赞动态。
方法
onReceiveLikesMessage: 收到新点赞消息的事件回调。当直播间/语音聊天房内有其他观众发送点赞时，会触发该事件并返回相关信息。

```
void Function(String liveID, int totalLikesReceived, LiveUserInfo
sender)? onReceiveLikesMessage;
LikeListener({this.onReceiveLikesMessage});
```

参数名	类型	说明
liveID	String	直播间 ID。
totalLikesReceived	int	本次收到的新点赞数。
sender	LiveUserInfo	点赞发送者信息。

LiveAudienceStore

最近更新时间：2026-04-20 17:34:07

简介

`LiveAudienceStore` 提供了一套完整的观众管理 API，包括获取观众列表、设置管理员、踢出用户、禁言等功能。 通过该类，可以在直播间内实现观众管理功能。

> **重要：**
> 使用 LiveAudienceStore.create 工厂方法创建 `LiveAudienceStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 观众状态更新通过 liveAudienceState 发布者传递。订阅它以接收房间内观众数据的实时更新。

功能特性

- **观众列表**：获取和管理当前房间的观众列表。
- **权限管理**：设置和撤销管理员权限。
- **用户管理**：踢出用户、禁言等操作。
- **事件监听**：监听房主、管理员、观众加入/离开等事件。

可订阅数据

LiveAudienceState 的字段描述如下：

属性名	类型	描述
audienceList	ValueListenable<List<LiveUserInfo>>	观众列表。
audienceCount	ValueListenable<int>	观众数量。
messageBannedUserList	ValueListenable<List<LiveUserInfo>>	消息被禁言的用户列表。

API 列表

函数名	描述

LiveAudienceStore.create	创建观众管理实例。
addLiveAudienceListener	观众事件回调。
removeLiveAudienceListener	观众事件回调。
fetchAudienceList	获取观众列表。
setAdministrator	设置管理员。
revokeAdministrator	撤销管理员。
kickUserOutOfRoom	踢出用户。
disableSendMessage	禁言/解禁用户。

创建实例

LiveAudienceStore.create

创建观众管理实例。

观察状态和事件

addLiveAudienceListener

添加观众事件监听器

```
void addLiveAudienceListener(LiveAudienceListener listener);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	LiveAudienceListener	必填	监听器。

removeLiveAudienceListener

移除观众事件监听器

```
void removeLiveAudienceListener(LiveAudienceListener listener);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	LiveAudienceListener	必填	监听器。

观众管理

fetchAudienceList

获取观众列表

```
Future<CompletionHandler> fetchAudienceList();
```

版本信息

从3.5版本开始支持。

setAdministrator

设置管理员

```
Future<CompletionHandler> setAdministrator(String userID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	要设置为管理员的用户 ID。

revokeAdministrator

撤销管理员

```
Future<CompletionHandler> revokeAdministrator(String userID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	要撤销管理员权限的用户 ID。

kickUserOutOfRoom

将用户踢出房间

```
Future<CompletionHandler> kickUserOutOfRoom(String userID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	要踢出的用户 ID。

disableSendMessage

禁用/解禁用户发送消息

```
Future<CompletionHandler> disableSendMessage({
  required String userID,
  required bool isDisable,
});
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	目标用户 ID。

isDisable	bool	必填	true 表示禁用发送消息，false 表示解禁。

数据结构

Role

用户角色。

枚举值	说明
owner	房主。
admin	管理员。
generalUser	普通用户。

LiveUserInfo

直播用户信息。

属性	类型	说明
userID	String	用户唯一标识 ID。
userName	String	用户名称。
avatarURL	String	用户头像 URL。

LiveAudienceState

直播观众状态。

属性	类型	说明
audienceList	ValueListenable<List<LiveUserInfo>>	观众列表。
audienceCount	ValueListenable<int>	观众数量。
messageBannedUserList	ValueListenable<List<LiveUserInfo>>	消息被禁言的用户列表。

LiveAudienceListener

直播观众事件。

此监听器用于接收直播间内全角色（房主、管理员、观众）的动态事件。

方法

onOwnerJoined: 房主加入事件。

```
void Function(LiveUserInfo owner)? onOwnerJoined;
```

参数名	类型	说明
owner	LiveUserInfo	加入的房主信息。

onOwnerLeft: 房主离开事件。

```
void Function(LiveUserInfo owner)? onOwnerLeft;
```

参数名	类型	说明
owner	LiveUserInfo	离开的房主信息。

onAdminJoined: 管理员加入事件。

```
void Function(LiveUserInfo admin)? onAdminJoined;
```

参数名	类型	说明
admin	LiveUserInfo	加入的管理员信息。

onAdminLeft: 管理员离开事件。

```
void Function(LiveUserInfo admin)? onAdminLeft;
```

参数名	类型	说明
admin	LiveUserInfo	离开的管理员信息。

onAudienceJoined: 观众加入事件。

```
void Function(LiveUserInfo audience)? onAudienceJoined;
```

参数名	类型	说明

audience	LiveUserInfo	加入的观众信息。

onAudienceLeft: 观众离开事件。

```
void Function(LiveUserInfo audience)? onAudienceLeft;
```

参数名	类型	说明
audience	LiveUserInfo	离开的观众信息。

onAudienceMessageDisabled: 观众被禁止发言事件。

```
void Function(LiveUserInfo audience, bool isDisable)?
onAudienceMessageDisabled;
LiveAudienceListener({this.onOwnerJoined, this.onOwnerLeft,
this.onAdminJoined, this.onAdminLeft, this.onAudienceJoined,
this.onAudienceLeft, this.onAudienceMessageDisabled});
```

参数名	类型	说明
audience	LiveUserInfo	观众信息。
isDisable	bool	是否被禁止发言。

LiveCoreView

最近更新时间：2026-04-20 17:34:07

简介

`LiveCoreView` 提供了直播推流和播放的视图容器，支持多人连麦、PK 等功能。 通过该组件，可以实现直播间的视频渲染和交互。

> **重要：**
> 使用前需要先调用 LiveCoreController.setLiveID 设置直播间 ID。

功能特性

- **视频渲染**：提供直播推流和播放的视图容器。
- **连麦支持**：支持多人连麦功能。
- **PK 支持**：支持主播 PK 功能。
- **房间外预览**：支持在进入房间前预览直播流。

数据结构

CoreViewType

核心视图类型。

枚举值	说明
playView	播放视图。
pushView	推流视图。

ViewLayer

视图层级。

枚举值	说明
foreground	前景层。
background	背景层。

LiveCoreController

Live core widget controller protocol

方法

create: 创建 LiveCoreController。

```
static LiveCoreController create(CoreViewType type) {
  return LiveCoreControllerImpl(type);
}
```

参数名	类型	说明
type	CoreViewType	核心视图类型。

setLiveID: 设置直播 ID。应该在使用其他接口之前先设置直播 ID。

```
void setLiveID(String liveID);
```

参数名	类型	说明
liveID	String	直播 ID。

startPreviewLiveStream: 房间外预览。

```
void startPreviewLiveStream(String roomID, bool isMuteAudio,
TUIPlayCallback? playCallback);
```

参数名	类型	说明
roomID	String	直播 ID。
isMuteAudio	bool	是否静音。
playCallback	TUIPlayCallback?	播放回调。

stopPreviewLiveStream: 停止房间外预览。

```
void stopPreviewLiveStream(String roomID);
```

参数名	类型	说明
roomID	String	直播 ID。

callExperimentalAPI: 调用实验性 API。

```
static void callExperimentalAPI(String jsonStr) {
  LiveCoreControllerImpl.callExperimentalAPI(jsonStr);
}
```

VideoViewAdapter

视频视图适配器协议。

方法

createCoGuestView: 创建连麦视图。

```
CoGuestWidgetBuilder coGuestWidgetBuilder = (BuildContext context,
SeatInfo seatInfo, ViewLayer viewPlayer) {
  return Container();
};
```

参数名	类型	说明
seatInfo	SeatInfo	连麦用户的麦位信息。
viewLayer	ViewLayer	视图层级，前景层或背景层。

createCoHostView: 创建跨房连麦视图。

```
CoHostWidgetBuilder coHostWidgetBuilder = (BuildContext context,
SeatInfo seatInfo, ViewLayer viewPlayer) {
  return Container();
};
```

参数名	类型	说明
seatInfo	SeatInfo	跨房连麦用户的麦位信息。
viewLayer	ViewLayer	视图层级，前景层或背景层。

createBattleView: 创建 PK 视图。

```
BattleWidgetBuilder battleWidgetBuilder = (BuildContext context,
SeatInfo seatInfo) {
```

```
  return Container();
};
```

参数名	类型	说明
seatInfo	SeatInfo	PK 用户的麦位信息。

createBattleContainerView: 创建 PK 容器视图。

```
BattleContainerWidgetBuilder battleContainerWidgetBuilder =
(BuildContext context) {
  return Container();
};
VideoWidgetBuilder({
  CoGuestWidgetBuilder? coGuestWidgetBuilder,
  CoHostWidgetBuilder? coHostWidgetBuilder,
  BattleWidgetBuilder? battleWidgetBuilder,
  BattleContainerWidgetBuilder? battleContainerWidgetBuilder,
}) {
  if (coGuestWidgetBuilder != null) this.coGuestWidgetBuilder =
coGuestWidgetBuilder;
  if (coHostWidgetBuilder != null) this.coHostWidgetBuilder =
coHostWidgetBuilder;
  if (battleWidgetBuilder != null) this.battleWidgetBuilder =
battleWidgetBuilder;
  if (battleContainerWidgetBuilder != null)
this.battleContainerWidgetBuilder = battleContainerWidgetBuilder;
}
```

LiveListStore

最近更新时间：2026-04-20 17:34:07

简介

`LiveListStore` 提供了一套完整的直播间管理 API，包括开播、加入直播、离开直播、结束直播等功能。 通过该类，可以实现直播间的生命周期管理。主播调用 startLive/endLive 来开播和解散房间，观众调用 joinLive/leaveLive 来加入和退出直播间。

> **重要：**
> 使用 **LiveListStore.shared** 单例对象获取 `LiveListStore` 实例。

> **说明：**
> 直播状态更新通过 **liveState** 发布者传递。订阅它以接收直播数据的实时更新。

功能特性

- **直播列表**：获取和管理直播间列表。
- **开播**：主播开播。如果是新的直播间 ID，将创建直播间并开播；如果是服务端已创建的直播间 ID，将加入该直播间并开播。
- **直播加入**：加入已存在的直播间。
- **直播管理**：更新直播信息、结束直播等操作。
- **事件监听**：监听直播结束、被踢出等事件。

可订阅数据

LiveListState 的字段描述如下：

属性名	类型	描述
liveList	ValueListenable<List<LiveInfo>>	直播列表。
liveListCursor	ValueListenable<String>	直播列表游标。
currentLive	ValueListenable<LiveInfo>	当前直播信息。

API 列表

函数名	描述

LiveListStore.shared	单例对象。
addLiveListListener	直播列表事件回调。
removeLiveListListener	直播列表事件回调。
fetchLiveList	获取直播列表。
fetchLiveInfo	获取直播信息。
startLive	主播开播。如果是新的直播间 ID，将创建直播间并开播；如果是服务端已创建的直播间 ID，将加入该直播间并开播。
joinLive	加入直播（观众调用）。
leaveLive	离开直播（观众调用）。
endLive	结束直播（主播调用）。
updateLiveInfo	更新直播信息。
queryMetaData	查询元数据。
updateLiveMetaData	更新元数据。

获取实例

LiveListStore.shared

单例对象。

观察状态和事件

addLiveListListener

添加直播列表事件监听器

```
void addLiveListListener(LiveListListener listener);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	LiveListListener	必填	监听器。

removeLiveListListener

移除直播列表事件监听器

```
void removeLiveListListener(LiveListListener listener);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
listener	LiveListListener	必填	监听器。

直播列表

fetchLiveList

获取直播列表

```
Future<CompletionHandler> fetchLiveList({
  required String cursor,
  required int count,
});
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
cursor	String	必填	游标。
count	int	必填	数量。

fetchLiveInfo

获取直播信息

```
Future<LiveInfoCompletionHandler> fetchLiveInfo(String liveID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播间 ID。
completion	LiveInfoCompletionHandler	必填	完成回调。

直播操作

startLive

开播

```
Future<LiveInfoCompletionHandler> startLive(LiveInfo liveInfo);
```

主播开播。如果是新的直播间 ID，将创建直播间并开播；如果是服务端已创建的直播间 ID，将加入该直播间并开播。

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveInfo	LiveInfo	必填	直播信息。

joinLive

加入直播

```
Future<LiveInfoCompletionHandler> joinLive(String liveID);
```

观众调用此接口加入已存在的直播间。

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
liveID	String	必填	直播 ID。

leaveLive

离开直播

```
Future<CompletionHandler> leaveLive();
```

观众调用此接口离开当前直播间。

若主播仅需退出房间但不解散直播间，也可调用此接口。

版本信息

从3.5版本开始支持。

endLive

结束直播（主播调用）

```
Future<StopLiveCompletionHandler> endLive();
```

主播调用此接口结束当前直播并解散房间。

版本信息

从3.5版本开始支持。

updateLiveInfo

更新直播信息

```
Future<CompletionHandler> updateLiveInfo({
  required LiveInfo liveInfo,
  required List<ModifyFlag> modifyFlagList,
});
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

liveInfo	LiveInfo	必填	直播信息。
modifyFlag	List<ModifyFlag>	必填	修改标志。

元数据操作

queryMetaData

查询元数据

```
Future<MetaDataCompletionHandler> queryMetaData(List<String> keys);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
keys	List<String>	必填	键列表。

updateLiveMetaData

更新直播元数据

```
Future<CompletionHandler> updateLiveMetaData(Map<String, String>
metaData);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
metaData	Map<String, String>	必填	元数据。

数据结构

TakeSeatMode

上麦模式。

枚举值	说明
free	自由上麦。
apply	申请上麦。

SeatLayoutTemplate

麦位布局模板，用于简化创建直播间时的麦位配置。

枚举值	说明
VideoDynamicGrid9Seats	竖屏动态九宫格，适用于视频直播场景。
VideoDynamicFloat7Seats	竖屏动态1v6浮动布局，适用于视频直播场景。
VideoLeftFocus9Seats	竖屏左侧焦点九宫格布局，适用于视频直播场景。
VideoUniformGrid9Seats	竖屏均匀九宫格布局，适用于视频直播场景。
VideoFixedGrid9Seats	竖屏静态九宫格，适用于视频直播场景。
VideoFixedFloat7Seats	竖屏静态1v6浮动布局，适用于视频直播场景。
VideoLandscape4Seats	横屏4人麦位布局，适用于视频直播场景。
Karaoke	语音 KTV 布局，适用于 K 歌场景，可指定麦位数量。
AudioSalon	语音沙龙布局，适用于语聊场景，可指定麦位数量。

LiveEndedReason

直播结束原因。

枚举值	说明
endedByHost	主播主动结束。
endedByServer	服务器结束。

LiveKickedOutReason

被踢出直播间原因。

枚举值	说明
byAdmin	被管理员踢出。
byLoggedOnOtherDevice	在其他设备登录。
byServer	被服务器踢出。
forNetworkDisconnected	网络断开连接。
forJoinRoomStatusInvalidDuringOffline	离线期间房间状态无效。
forCountOfJoinedRoomsExceedLimit	加入房间数量超过限制。

LiveInfo

直播信息。

属性	类型	说明
liveID	String	直播 ID。
liveName	String	直播名称。
notice	String	直播公告。
isMessageDisable	bool	是否禁用消息。
isPublicVisible	bool	是否公开可见。
isSeatEnabled	bool	是否启用麦位。
keepOwnerOnSeat	bool	房主是否保持在麦位上。
maxSeatCount	int	最大麦位数量。
seatMode	TakeSeatMode	上麦模式。

seatTemplate	SeatLayoutTemplate	麦位布局模板，用于简化麦位配置。
seatLayoutTemplateID	int	麦位布局模板 ID。
coverURL	String	封面 URL。
backgroundURL	String	背景 URL。
categoryList	List<int>	分类列表。
activityStatus	int	活动状态。
liveOwner	LiveUserInfo	直播房主信息。
createTime	int	创建时间。
totalViewerCount	int	总观看人数。
isGiftEnabled	bool	是否启用礼物。
metaData	Map<String, String>	元数据。

LiveListState

直播列表状态。

属性	类型	说明
liveList	ValueListenable<List<LiveInfo>>	直播列表。
liveListCursor	ValueListenable<String>	直播列表游标。
currentLive	ValueListenable<LiveInfo>	当前直播信息。

LiveListListener

直播列表事件。

方法

onLiveEnded: 直播结束事件。

```
void Function(String liveID, LiveEndedReason reason, String message)?
onLiveEnded;
```

参数名	类型	说明
liveID	String	直播 ID。
reason	LiveEndedReason	结束原因。
message	String	消息。

onKickedOutOfLive: 被踢出直播间事件。

```
void Function(String liveID, LiveKickedOutReason reason, String
message)? onKickedOutOfLive;
LiveListListener({this.onLiveEnded, this.onKickedOutOfLive});
```

参数名	类型	说明
liveID	String	直播 ID。
reason	LiveKickedOutReason	被踢出原因。
message	String	消息。

LiveInfoCompletionHandler

Dart 直播信息完成回调。
Dart 直播信息操作的完成回调，包含返回的直播信息。
属性

属性	类型	说明
liveInfo	LiveInfo	成功时返回的直播信息。

StopLiveCompletionHandler

Dart 停止直播完成回调。
Dart 停止直播操作的完成回调，包含直播统计数据。
属性

属性	类型	说明
statisticsData	TUILiveStatisticsData	成功时返回的直播统计数据。

MetaDataCompletionHandler

Dart 元数据完成回调。

Dart 元数据操作的完成回调，包含元数据结果。

属性

属性	类型	说明
metaData	Map<String, String>	成功时返回的元数据。

LiveSeatStore

最近更新时间：2026-04-20 17:34:07

简介

`LiveSeatStore` 提供了一套完整的麦位管理 API，包括上麦、下麦、锁麦、解锁麦位、踢用户下麦、远程控制设备等功能。 通过该类，可以在直播间内实现麦位管理功能。

> **重要：**
> 使用 **LiveSeatStore.create** 工厂方法创建 `LiveSeatStore` 实例，需要传入有效的直播间 ID。

> **说明：**
> 麦位状态更新通过 **liveSeatState** 发布者传递。订阅它以接收房间内麦位数据的实时更新。

功能特性

- **麦位管理**：上麦、下麦、锁麦、解锁麦位等操作。
- **用户管理**：踢用户下麦、移动用户到指定麦位。
- **设备控制**：远程控制用户的摄像头和麦克风。
- **事件监听**：监听麦位相关事件。

可订阅数据

LiveSeatState 的字段描述如下：

属性名	类型	描述
seatList	ValueListenable<List<SeatInfo>>	麦位列表。
canvas	ValueListenable<LiveCanvas>	画布信息。
speakingUsers	ValueListenable<Map<String, int>>	正在说话的用户。
avStatistics	ValueListenable<List<AVStatistics>>	音视频相关统计信息。

API 列表

函数名	描述
LiveSeatStore.create	创建麦位管理实例。
addLiveSeatEventListener	麦位事件回调。
removeLiveSeatEventListener	麦位事件回调。
takeSeat	上麦。
leaveSeat	下麦。
lockSeat	锁麦。
unlockSeat	解锁麦位。
kickUserOutOfSeat	踢用户下麦。
moveUserToSeat	移动用户。
openRemoteCamera	开启远程摄像头。
closeRemoteCamera	关闭远程摄像头。
openRemoteMicrophone	开启远程麦克风。
closeRemoteMicrophone	关闭远程麦克风。

创建实例

LiveSeatStore.create

创建麦位管理实例。

观察状态和事件

addLiveSeatEventListener

添加麦位事件监听器

```
void addLiveSeatEventListener(LiveSeatListener listener);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	LiveSeatListener	必填	监听器。

removeLiveSeatEventListener

移除麦位事件监听器

```
void removeLiveSeatEventListener(LiveSeatListener listener);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
listener	LiveSeatListener	必填	监听器。

麦位操作

takeSeat

上麦

```
Future<CompletionHandler> takeSeat(int seatIndex);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
seatIndex	int	必填	麦位索引。

leaveSeat

下麦

```
Future<CompletionHandler> leaveSeat();
```

版本信息
从3.5版本开始支持。

lockSeat

锁定麦位

```
Future<CompletionHandler> lockSeat(int seatIndex);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
seatIndex	int	必填	麦位索引。

unlockSeat

解锁麦位

```
Future<CompletionHandler> unlockSeat(int seatIndex);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
seatIndex	int	必填	麦位索引。

用户管理

kickUserOutOfSeat

踢用户下麦

```
Future<CompletionHandler> kickUserOutOfSeat(String userID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。

moveUserToSeat

移动用户到麦位

```
Future<CompletionHandler> moveUserToSeat({
  required String userID,
  required int targetIndex,
  MoveSeatPolicy policy = MoveSeatPolicy.abortWhenOccupied,
});
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。
targetIndex	int	必填	目标麦位索引。
policy	MoveSeatPolicy	必填	移动策略。

远程设备控制

openRemoteCamera

开启远程摄像头

```
Future<CompletionHandler> openRemoteCamera({
  required String userID,
  required DeviceControlPolicy policy,
```

```
});
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。
policy	DeviceControlPolicy	必填	设备控制策略。

closeRemoteCamera

关闭远程摄像头

```
Future<CompletionHandler> closeRemoteCamera(String userID);
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。

openRemoteMicrophone

开启远程麦克风

```
Future<CompletionHandler> openRemoteMicrophone({
  required String userID,
  required DeviceControlPolicy policy,
});
```

版本信息

从3.5版本开始支持。

参数说明

参数名	类型	是否必填	描述

userID	String	必填	用户 ID。
policy	DeviceControlPolicy	必填	设备控制策略。

closeRemoteMicrophone

关闭远程麦克风

```
Future<CompletionHandler> closeRemoteMicrophone(String userID);
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
userID	String	必填	用户 ID。

数据结构

MoveSeatPolicy

移动麦位策略。

枚举值	说明
abortWhenOccupied	被占用时中止。
forceReplace	强制替换。
swapPosition	交换位置。

DeviceControlPolicy

设备控制策略。

枚举值	说明
unlockOnly	仅解锁。

SuspendStatus

用户的挂起状态。

枚举值	说明
none	未挂起。
inBackground	用户进入后台挂起。
inCalling	用户正在接听电话。

LiveSeatListener

麦位相关的回调事件。

方法

方法名	说明
onLocalCameraOpenedByAdmin	当本地摄像头被管理员开启时触发此回调。
onLocalCameraClosedByAdmin	当本地摄像头被管理员关闭时触发此回调。
onLocalMicrophoneOpenedByAdmin	当本地麦克风被管理员开启时触发此回调。
onLocalMicrophoneClosedByAdmin	当本地麦克风被管理员关闭时触发此回调。

SeatUserInfo

麦位用户信息。

属性	类型	说明
userID	String	用户 ID。
userName	String	用户名。
avatarURL	String	头像 URL。
role	Role	用户角色。
liveID	String	直播间 ID。
microphoneStatus	DeviceStatus	麦克风状态。
allowOpenMicrophone	bool	是否允许开启麦克风。

cameraStatus	DeviceStatus	摄像头状态。
allowOpenCamera	bool	是否允许开启摄像头。
userSuspendStatus	SuspendStatus	用户挂起状态。

RegionInfo

麦位视图坐标信息。

属性	类型	说明
x	int	X 坐标。
y	int	Y 坐标。
w	int	宽度。
h	int	高度。
zorder	int	层级顺序。

AVStatistics

音视频相关统计信息。

属性	类型	说明
userID	String	用户 ID。
videoBitrate	int	本地视频的码率。
videoWidth	int	本地视频的宽度。
videoHeight	int	本地视频的高度。
frameRate	int	本地视频的帧率。
audioSampleRate	int	音频的采样率。
audioBitrate	int	音频码率。

SeatInfo

麦位信息。

属性	类型	说明
index	int	麦位索引。
isLocked	bool	是否锁定。
userInfo	SeatUserInfo	用户信息。
region	RegionInfo	区域信息。

LiveCanvas

直播画布。

属性	类型	说明
w	int	宽度。
h	int	高度。
templateID	int	模板 ID。

LiveSeatState

LiveSeatStore 对外提供的麦位状态数据。

属性	类型	说明
seatList	ValueListenable<List<SeatInfo>>	麦位列表。
canvas	ValueListenable<LiveCanvas>	画布信息。
speakingUsers	ValueListenable<Map<String, int>>	正在说话的用户。
avStatistics	ValueListenable<List<AVStatistics>>	音视频相关统计信息。

LoginStore

最近更新时间：2026-04-20 17:34:07

简介

`LoginStore` 提供了一套完整的登录管理 API，包括用户登录、登出、设置个人信息等功能。 通过该类，可以管理用户的登录状态和用户资料。

> **重要：**
> 使用 **shared** 单例对象访问 `LoginStore` 实例。

> **说明：**
> 登录状态更新通过 **loginState** 发布者传递。订阅它以接收登录状态的实时更新。

功能特性

- **用户登录**：支持使用 SDK 应用 ID、用户 ID 和用户签名进行登录。
- **用户登出**：支持用户登出操作。
- **个人信息设置**：支持设置用户昵称、头像、性别等个人资料。

可订阅数据

LoginState 的字段描述如下：

属性名	类型	描述
loginStatus	LoginStatus	登录状态。
loginUserInfo	UserProfile?	登录用户信息。

API 列表

函数名	描述
shared	单例对象。
login	登录。
logout	登出。
setSelfInfo	设置个人信息。

获取实例

shared

单例对象

```
static LoginStore get shared => LoginStoreImpl.instance;
```

版本信息
从3.5版本开始支持。

登录操作

login

登录

```
Future<CompletionHandler> login({
  required int sdkAppID,
  required String userID,
  required String userSig,
});
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
sdkAppID	int	必填	SDK 应用 ID。
userID	String	必填	用户 ID。
userSig	String	必填	用户签名。

logout

登出

```
Future<CompletionHandler> logout();
```

版本信息
从3.5版本开始支持。

setSelfInfo

设置个人信息

```
Future<CompletionHandler> setSelfInfo({required UserProfile userInfo});
```

版本信息
从3.5版本开始支持。
参数说明

参数名	类型	是否必填	描述
userProfile	UserProfile	必填	用户资料。

数据结构

LoginStatus

登录状态。

枚举值	说明
unlogin	未登录。
logined	已登录。

AllowType

好友验证方式。

枚举值	值	说明
allowAny	0	允许任何人。
needConfirm	1	需要验证。
denyAny	2	拒绝任何人。

Gender

性别。

枚举值	值	说明
unknown	0	未知。

male	1	男性。
female	2	女性。

LoginEvent

登录事件。

枚举值	说明
kickedOffline	当前用户被踢下线。
loginExpired	登录票据过期。

UserProfile

用户资料。

属性	类型	说明
userID	String	用户 ID。
nickname	String?	昵称。
avatarURL	String?	头像 URL。
selfSignature	String?	个性签名。
gender	Gender?	性别。
role	int?	角色。
level	int?	等级。
birthday	int?	生日。
allowType	AllowType?	好友验证方式。
customInfo	Map<String, String>?	自定义信息。

LoginState

登录状态。

属性	类型	说明
loginStatus	LoginStatus	登录状态。

loginUserInfo	UserProfile?	登录用户信息。

uni-app

最近更新时间：2026-04-20 17:34:07

AtomicXCore SDK 是腾讯云最新推出的面向即时通信、音视频通话、视频直播、语聊房等场景的全新一代基于响应式的 API，您可以非常快速的基于这组 API 构建自己的 UI 页面，它支持房间管理、屏幕分享、成员管理、麦位控制、基础美颜等丰富功能，同时基于 TRTC SDK，能够提供超低延时、高品质的音视频体验，本页面包含 AtomicXCore SDK 的所有 API 接口，按功能模块分类展示。

LoginState

用户身份认证与登录管理模块
核心功能：负责用户身份验证、登录状态管理、用户信息维护等基础认证服务。
响应式数据

数据列表	描述
loginUserInfo	当前登录用户信息。
loginStatus	当前登录状态。

接口函数

函数列表	描述
login	登录方法。
logout	登出方法。
setSelfInfo	设置用户信息。

LiveListState

直播列表管理模块

- 核心功能：管理直播间的完整生命周期，包括创建、加入、离开、结束等核心业务流程。
- 技术特点：支持分页加载、实时状态同步、直播信息动态更新，采用响应式数据管理，确保 UI 与数据状态实时同步。
- 业务价值：为直播平台提供核心的直播间管理能力，支持大规模并发直播场景，是直播业务的基础设施。
- 应用场景：直播列表展示、直播间创建、直播状态管理、直播数据统计等核心业务场景。

响应式数据

数据列表	描述

liveList	直播列表数据。
liveListCursor	直播列表游标，用于分页加载。
currentLive	当前直播信息。

接口函数

函数列表	描述
fetchLiveList	获取直播列表。
createLive	创建直播间。
joinLive	加入直播间。
leaveLive	离开直播间。
endLive	结束直播。
updateLiveInfo	更新直播信息。
addLiveListListener	添加直播列表事件监听。
removeLiveListListener	移除直播列表事件监听。

LiveSeatState

直播间座位管理模块

- 核心功能：实现多人连麦场景下的座位控制，支持复杂的座位状态管理和音视频设备控制。
- 技术特点：基于 WebRTC 技术，支持多路音视频流管理，提供座位锁定、设备控制、权限管理等高级功能。
- 业务价值：为多人互动直播提供核心技术支撑，支持PK、连麦、多人游戏等丰富的互动场景。
- 应用场景：多人连麦、主播 PK、互动游戏、在线教育、会议直播等需要多人音视频互动的场景。

响应式数据

数据列表	描述
seatList	座位列表。
canvas	画布信息。
speakingUsers	正在说话的用户列表。

接口函数

函数列表	描述
takeSeat	用户上麦。
leaveSeat	用户下麦。
muteMicrophone	静音麦克风。
unmuteMicrophone	取消静音麦克风。
kickUserOutOfSeat	将用户踢出麦位。
moveUserToSeat	将用户移动到麦位。
lockSeat	锁定麦位。
unlockSeat	解锁麦位。
openRemoteCamera	开启远端用户摄像头。
closeRemoteCamera	关闭远端用户摄像头。
openRemoteMicrophone	开启远端用户麦克风。
closeRemoteMicrophone	关闭远端用户麦克风。
addLiveSeatEventListener	添加麦位事件监听。
removeLiveSeatEventListener	移除麦位事件监听。

LiveAudienceState

直播间观众管理模块

- 核心功能：管理直播间观众列表，提供观众权限控制、管理员设置等直播间秩序维护功能。
- 技术特点：支持实时观众列表更新、权限分级管理、批量操作等高级功能，确保直播间秩序和用户体验。
- 业务价值：为直播平台提供完整的观众管理解决方案，支持大规模观众场景下的秩序维护。
- 应用场景：观众管理、权限控制、直播间秩序维护、观众互动管理等核心业务场景。

响应式数据

数据列表	描述
audienceList	直播间观众列表。
audienceCount	直播间观众数量。

接口函数

函数列表	描述
fetchAudienceList	获取观众列表。
setAdministrator	设置管理员。
revokeAdministrator	撤销管理员。
kickUserOutOfRoom	将用户踢出房间。
disableSendMessage	禁用发送消息。
addAudienceListener	添加观众监听。
removeAudienceListener	移除观众监听。

DeviceState

设备状态管理模块

- 核心功能：管理摄像头、麦克风等音视频设备的控制，提供设备状态监控、权限检查等基础设备服务。
- 技术特点：支持多设备管理、设备状态实时监控、权限动态检查、设备故障自动恢复等高级功能。
- 业务价值：为直播系统提供稳定的设备基础，确保音视频采集的可靠性和用户体验。
- 应用场景：设备管理、权限控制、音视频采集、设备故障处理等基础技术场景。

响应式数据

数据列表	描述
microphoneStatus	麦克风开启状态。
microphoneLastError	麦克风最后一次错误状态。
hasPublishAudioPermission	是否有音频发布权限。
captureVolume	采集音量大小（0-100）。
currentMicVolume	当前麦克风音量（0-100）。
outputVolume	输出音量大小（0-100）。
cameraStatus	摄像头开启状态。
cameraLastError	摄像头最后一次错误状态。

isFrontCamera	是否为前置摄像头。
localMirrorType	本地镜像类型。
localVideoQuality	本地视频质量设置。
currentAudioRoute	当前音频输出路由（扬声器/耳机）。
screenStatus	屏幕共享状态。
networkInfo	网络信息状态。

接口函数

函数列表	描述
openLocalMicrophone	开启本地麦克风。
closeLocalMicrophone	关闭本地麦克风。
setCaptureVolume	设置采集音量。
setOutputVolume	设置输出音量。
openLocalCamera	开启本地摄像头。

CoGuestState

连麦嘉宾管理模块

- 核心功能：处理观众与主播之间的连麦互动，管理连麦申请、邀请、接受、拒绝等完整的连麦流程。
- 技术特点：基于实时音视频技术，支持连麦状态实时同步、音视频质量自适应、网络状况监控等高级功能。
- 业务价值：为直播平台提供观众参与互动的核心能力，增强用户粘性和直播趣味性。
- 应用场景：观众连麦、互动问答、在线K歌、游戏直播等需要观众参与的互动场景。

响应式数据

数据列表	描述
connected	已连接的连麦嘉宾列表。
invitees	被邀请上麦的用户列表。
applicants	申请上麦的用户列表。
candidates	可邀请上麦的候选用户列表。

接口函数

函数列表	描述
applyForSeat	申请连麦座位。
cancelApplication	取消申请。
acceptApplication	接受申请。
rejectApplication	拒绝申请。
inviteToSeat	邀请上麦。
cancelInvitation	取消邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
disconnect	断开连麦连接。
addCoGuestGuestListener	添加连麦嘉宾侧事件监听。
removeCoGuestGuestListener	移除连麦嘉宾侧事件监听。
addCoGuestHostListener	添加连麦主播侧事件监听。
removeCoGuestHostListener	移除连麦主播侧事件监听。

CoHostState

连线主播管理模块

- **核心功能：实现主播间的连线功能，支持主播邀请、连线申请、连线状态管理等主播间互动功能。**
- **技术特点：支持多主播音视频同步、画中画显示、音视频质量优化等高级技术，确保连麦体验的流畅性。**
- **业务价值：为直播平台提供主播间协作的核心能力，支持 PK、合作直播等高级业务场景。**
- **应用场景：主播 PK、合作直播、跨平台连麦、主播互动等高级直播场景。**

响应式数据

数据列表	描述
connected	已连接的连麦主播列表。
invitees	被邀请连线的主播列表。
applicant	当前申请连线的主播信息。

candidates	可邀请连线的候选主播列表。
coHostStatus	当前连线状态。

接口函数

函数列表	描述
requestHostConnection	请求连线。
cancelHostConnection	取消连线请求。
acceptHostConnection	接受连线请求。
rejectHostConnection	拒绝连线请求。
exitHostConnection	退出连线。
addCoHostListener	添加连线主播事件监听。
removeCoHostListener	移除连线主播事件监听。

AudioEffectState

音效处理模块

- 核心功能：提供变声、混响、耳返等高级音效功能，支持多种音效效果和实时音效调节。
- 技术特点：基于腾讯天籁实验室的音频处理算法，支持实时音效处理、低延迟音频传输、音质优化等高级技术。
- 业务价值：为直播平台提供差异化的音效体验，增强用户参与度和直播趣味性。
- 应用场景：变声直播、K 歌直播、音效娱乐、专业音效等需要音频处理的场景。

响应式数据

数据列表	描述
isEarMonitorOpened	耳返开关状态。
earMonitorVolume	耳返音量大小。
audioChangerType	变声状态。
audioReverbType	混响状态。

接口函数

函数列表	描述

setAudioChangerType	设置变声效果。
setAudioReverbType	设置混响效果。
setVoiceEarMonitorEnable	设置耳返开关状态。
setVoiceEarMonitorVolume	设置耳返音量大小。

BarrageState

弹幕消息管理模块

- 核心功能：处理直播间内的文本消息、自定义消息等弹幕功能，支持弹幕发送、消息状态同步等完整弹幕系统。
- 技术特点：支持高并发消息处理、实时消息同步、消息过滤、表情包支持等高级功能。
- 业务价值：为直播平台提供核心的互动能力，增强用户参与度和直播氛围。
- 应用场景：弹幕互动、消息管理、表情包、聊天室等社交互动场景。

响应式数据

数据列表	描述
messageList	当前房间的弹幕消息列表。
allowSendMessage	是否允许发送消息。

接口函数

函数列表	描述
sendTextMessage	发送文本类型弹幕。
sendCustomMessage	发送自定义类型弹幕。

BaseBeautyState

基础美颜模块

- 核心功能：提供磨皮、美白、红润等基础美颜效果调节，支持实时美颜参数调整。
- 技术特点：基于 AI 美颜算法，支持实时美颜处理、参数平滑调节、性能优化等高级技术。
- 业务价值：为直播平台提供基础的美颜能力，提升用户形象和直播质量。
- 应用场景：美颜直播、形象优化、美颜调节、直播美化等需要美颜功能的场景。

响应式数据

数据列表	描述

smoothLevel	磨皮级别 取值范围[0,9]: 0 表示关闭，9 表示效果最明显。
whitenessLevel	美白级别 取值范围[0,9]: 0 表示关闭，9 表示效果最明显。
ruddyLevel	红润级别 取值范围[0,9]: 0 表示关闭，9 表示效果最明显。

接口函数

函数列表	描述
setSmoothLevel	设置磨皮级别。
setWhitenessLevel	设置美白级别。
setRuddyLevel	设置红润级别。

GiftState

礼物系统管理模块

- 核心功能：处理礼物的发送、接收、礼物列表管理等功能，支持礼物分类、礼物动画、礼物统计等完整礼物经济系统。
- 技术特点：支持礼物动画渲染、礼物特效处理、礼物统计、礼物排行榜等高级功能。
- 业务价值：为直播平台提供核心的变现能力，支持礼物经济、虚拟货币等商业模式。 应用场景：礼物打赏、虚拟货币、礼物特效、礼物统计等商业化场景。

响应式数据

数据列表	描述
usableGifts	可用礼物列表。

接口函数

函数列表	描述
refreshUsableGifts	刷新可用礼物列表。
sendGift	发送礼物。
addGiftListener	添加礼物事件监听器。
removeGiftListener	移除礼物事件监听器。

LikeState

点赞互动模块

- 核心功能：处理直播间的点赞功能，支持点赞发送、点赞统计、点赞事件监听等互动功能。
- 技术特点：支持高并发点赞处理、实时点赞统计、点赞动画效果、点赞排行榜等高级功能。
- 业务价值：为直播平台提供基础的互动能力，增强用户参与度和直播氛围。
- 应用场景：点赞互动、人气统计、互动效果、用户参与等基础互动场景。

响应式数据

数据列表	描述
totalLikeCount	总点赞数量。

接口函数

函数列表	描述
sendLike	发送点赞。
addLikeListener	添加点赞事件监听。
removeLikeListener	移除点赞事件监听。

Web

最近更新时间：2026-04-20 17:34:07

AtomicXCore SDK 是腾讯云最新推出的面向即时通信、音视频通话、视频直播、语聊房等场景的全新一代基于响应式的 API，您可以非常快速的基于这组 API 构建自己的 UI 页面，它支持房间管理、屏幕分享、成员管理、麦位控制、基础美颜等丰富功能，同时基于 TRTC SDK，能够提供超低延时、高品质的音视频体验，本页面包含 AtomicXCore SDK 的所有 API 接口，按功能模块分类展示。

LoginState

用户身份认证与登录管理模块

- 核心功能：提供用户登录、登出、个人信息管理等基础身份认证功能，是整个系统的用户身份基础。
- 技术特点：支持多种登录方式、用户信息缓存、登录状态持久化等功能，确保用户身份的安全性和可靠性。
- 业务价值：为所有业务模块提供统一的用户身份认证服务，是系统安全和用户体验的基础保障。
- 应用场景：用户登录、身份验证、个人信息管理、权限控制等核心身份认证场景。

响应式数据

数据列表	描述
loginUserInfo	当前登录用户信息。

接口函数

函数列表	描述
login	用户登录。
setSelfInfo	设置用户个人信息。
logout	用户登出函数。

DeviceState

设备状态管理模块

- 核心功能：管理摄像头、麦克风等音视频设备的控制，提供设备状态监控、权限检查等基础设备服务。
- 技术特点：支持多设备管理、设备状态实时监控、权限动态检查、设备故障自动恢复等高级功能。
- 业务价值：为直播系统提供稳定的设备基础，确保音视频采集的可靠性和用户体验。
- 应用场景：设备管理、权限控制、音视频采集、设备故障处理等基础技术场景。

响应式数据

数据列表	描述

microphoneStatus	麦克风状态。
microphoneList	麦克风设备列表。
currentMicrophone	当前选中的麦克风设备。
microphoneLastError	麦克风最后一次错误信息。
captureVolume	麦克风采集音量。
currentMicVolume	当前麦克风音量。
isMicrophoneTesting	麦克风测试状态。
testingMicVolume	测试时的麦克风音量。
cameraStatus	摄像头状态。
cameraList	摄像头设备列表。
currentCamera	当前选中的摄像头设备。
cameraLastError	摄像头最后一次错误信息。
isCameraTesting	摄像头测试状态。
isCameraTestLoading	摄像头测试加载状态。
isFrontCamera	是否为前置摄像头。
localMirrorType	本地视频镜像类型。
localVideoQuality	本地视频质量。
speakerList	扬声器设备列表。
currentSpeaker	当前选中的扬声器设备。
outputVolume	扬声器输出音量。
currentAudioRoute	当前音频路由。
isSpeakerTesting	扬声器测试状态。
screenStatus	屏幕分享状态。
networkInfo	网络信息。

接口函数

函数列表	描述
openLocalMicrophone	开启本地麦克风。
closeLocalMicrophone	关闭本地麦克风。
muteLocalAudio	静音本地音频。
unmuteLocalAudio	取消静音本地音频。
getMicrophoneList	获取麦克风设备列表。
setCurrentMicrophone	设置当前麦克风设备。
startMicrophoneTest	开始麦克风测试。
setCaptureVolume	设置音频采集音量。
setOutputVolume	设置音频播放音量。
stopMicrophoneTest	停止麦克风测试。
getSpeakerList	获取扬声器设备列表。
setCurrentSpeaker	设置当前扬声器设备。
setAudioRoute	设置音频路由。
startSpeakerTest	开始扬声器测试。
stopSpeakerTest	停止扬声器测试。
openLocalCamera	开启本地摄像头。
closeLocalCamera	关闭本地摄像头。
getCameraList	获取摄像头设备列表。
setCurrentCamera	设置当前摄像头设备。
switchCamera	切换前后摄像头。
switchMirror	切换视频镜像模式。
updateVideoQuality	更新视频质量。
startCameraDeviceTest	开始摄像头设备测试。
startScreenShare	开始屏幕分享。

stopScreenShare	停止屏幕分享。
stopCameraDeviceTest	停止摄像头设备测试。
screenCaptureStopped	屏幕分享停止回调。

LiveListState

直播列表管理模块

- 核心功能：管理直播间的完整生命周期，包括创建、加入、离开、结束等核心业务流程，支持直播列表的分页获取和实时更新。
- 技术特点：支持分页加载、实时状态同步、直播信息动态更新，采用响应式数据管理，确保 UI 与数据状态实时同步。新增 liveList 和 liveListCursor 响应式数据，提供 fetchLiveList 接口函数。
- 业务价值：为直播平台提供核心的直播间管理能力，支持大规模并发直播场景，是直播业务的基础设施。
- 应用场景：直播列表展示、直播间创建、直播状态管理、直播数据统计等核心业务场景。

响应式数据

数据列表	描述
currentLive	转换TUILiveInfo为LiveInfo格式。
liveList	直播列表数据，包含所有直播间的信息。
liveListCursor	直播列表分页游标，用于获取下一页数据。

接口函数

函数列表	描述
createLive	创建直播间。
joinLive	加入直播间。
leaveLive	离开直播间。
endLive	结束直播。
updateLiveInfo	更新直播间信息。
queryMetaData	查询元数据。
updateLiveMetaData	更新直播间元数据。
fetchLiveList	获取直播列表。

LiveSeatState

直播间座位管理模块

- 核心功能：实现多人连麦场景下的座位控制，支持复杂的座位状态管理和音视频设备控制，包括上麦、下麦、座位锁定等完整功能。
- 技术特点：基于 WebRTC 技术，支持多路音视频流管理，提供座位锁定、设备控制、权限管理等高级功能。新增 seatList、canvas、speakingUsers、networkQualities 响应式数据，以及完整的座位操作接口。
- 业务价值：为多人互动直播提供核心技术支撑，支持 PK、连麦、多人游戏等丰富的互动场景。
- 应用场景：多人连麦、主播 PK、互动游戏、在线教育、会议直播等需要多人音视频互动的场景。

响应式数据

数据列表	描述
seatList	麦位列表，包含所有麦位的状态和用户信息。
canvas	画布配置，用于视频渲染和布局管理。
speakingUsers	正在发言的用户列表。
networkQualities	网络质量信息，包含各用户的网络状态。

接口函数

函数列表	描述
takeSeat	用户上麦。
leaveSeat	用户下麦。
lockSeat	锁定麦位。
unLockSeat	解锁麦位。
kickUserOutOfSeat	踢用户下麦。
moveUserToSeat	移动用户到指定麦位。
openRemoteCamera	开启远端摄像头。
closeRemoteCamera	关闭远端摄像头。
openRemoteMicrophone	开启远端麦克风。
closeRemoteMicrophone	关闭远端麦克风。
muteMicrophone	静音麦克风。

unmuteMicrophone	取消静音麦克风。
startPlayStream	开始播放流。
stopPlayStream	停止播放流。

LiveAudienceState

直播间观众管理模块

- 核心功能：管理直播间观众列表，提供观众权限控制、管理员设置等直播间秩序维护功能，支持实时观众统计。
- 技术特点：支持实时观众列表更新、权限分级管理、批量操作等高级功能，确保直播间秩序和用户体验。新增 audienceList 和 audienceCount 响应式数据。
- 业务价值：为直播平台提供完整的观众管理解决方案，支持大规模观众场景下的秩序维护。
- 应用场景：观众管理、权限控制、直播间秩序维护、观众互动管理等核心业务场景。

响应式数据

数据列表	描述
audienceList	观众列表，包含直播间所有观众的信息。
audienceCount	观众总数统计。

接口函数

函数列表	描述
fetchAudienceList	获取观众列表。
setAdministrator	设置管理员。
revokeAdministrator	撤销管理员。
kickUserOutOfRoom	踢出房间。
disableSendMessage	禁言用户。

LiveMonitorState

直播监控管理模块

- 核心功能：提供直播间实时监控功能，包括直播状态监控、数据统计、异常检测等核心监控能力，支持多直播间监控。
- 技术特点：支持实时数据采集、多维度监控指标、智能告警机制，确保直播服务的稳定性和可靠性。新增 monitorLiveInfoList 响应式数据，优化监控接口。
- 业务价值：为直播平台提供全方位的监控保障，及时发现和处理异常情况，提升服务质量。

- 应用场景：直播质量监控、性能分析、异常告警、数据统计等运营管理场景。

响应式数据

数据列表	描述
monitorLiveInfoList	监控的直播间信息列表。

接口函数

函数列表	描述
init	初始化监控。
getLiveList	获取直播列表。
closeRoom	关闭房间。
sendMessage	发送消息。
startPlay	开始播放。
stopPlay	停止播放。
muteLiveAudio	静音直播音频。

CoGuestState

连麦嘉宾管理模块

- 核心功能：处理观众与主播之间的连麦互动，管理连麦申请、邀请、接受、拒绝等完整的连麦流程，支持连麦状态管理。
- 技术特点：基于实时音视频技术，支持连麦状态实时同步、音视频质量自适应、网络状况监控等高级功能。新增 connected、invitees、applicants、candidates 响应式数据和 applyForSeat 接口。
- 业务价值：为直播平台提供观众参与互动的核心能力，增强用户粘性和直播趣味性。
- 应用场景：观众连麦、互动问答、在线K歌、游戏直播等需要观众参与的互动场景。

响应式数据

数据列表	描述
candidates	取消订阅连麦事件。
connected	连麦连接状态。
invitees	被邀请的用户列表。
applicants	申请连麦的用户列表。

接口函数

函数列表	描述
cancelApplication	取消申请。
acceptApplication	接受申请。
rejectApplication	拒绝申请。
cancelInvitation	取消邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
disConnect	断开连接。
applyForSeat	申请上麦。

CoHostState

连麦主播管理模块

- 核心功能：实现主播间的连麦功能，支持主播邀请、连麦申请、连麦状态管理等主播间互动功能，提供完整的主播连麦流程。
- 技术特点：支持多主播音视频同步、画中画显示、音视频质量优化等高级技术，确保连麦体验的流畅性。新增 coHostStatus、connected、applicant、invitees、candidates 响应式数据和完整的连麦控制接口。
- 业务价值：为直播平台提供主播间协作的核心能力，支持 PK、合作直播等高级业务场景。
- 应用场景：主播 PK、合作直播、跨平台连麦、主播互动等高级直播场景。

响应式数据

数据列表	描述
coHostStatus	连麦主播状态。
connected	连麦连接状态。
applicant	申请连麦的主播信息。
invitees	被邀请的用户列表。
candidates	候选用户列表。

接口函数

函数列表	描述
requestHostConnection	请求主播连麦。
cancelHostConnection	取消主播连麦。
acceptHostConnection	接受主播连麦。
rejectHostConnection	拒绝主播连麦。
exitHostConnection	退出主播连麦。

BattleState

PK 对战管理模块

- 核心功能：管理主播间的 PK 对战功能，包括对战邀请、接受、拒绝、结束等完整的 PK 流程，支持实时比分统计。
- 技术特点：支持实时对战状态同步、比分计算、对战结果统计等功能。新增 battleScore 响应式数据，提供完整的 PK 对战体验。
- 业务价值：为直播平台提供竞技互动功能，增强直播趣味性和用户参与度。
- 应用场景：主播 PK、才艺比拼、游戏对战、互动竞技等娱乐场景。

响应式数据

数据列表	描述
currentBattleInfo	当前PK信息。
battleUsers	PK参与用户列表。
battleScore	PK对战的实时比分。

接口函数

函数列表	描述
requestBattle	请求 PK。
cancelBattleRequest	取消 PK 请求。
acceptBattle	接受 PK。
rejectBattle	拒绝 PK。
exitBattle	退出 PK。

BarrageState

弹幕消息管理模块

- 核心功能：处理直播间内的文本消息、自定义消息等弹幕功能，支持弹幕发送、消息状态同步等完整弹幕系统，提供本地提示功能。
- 技术特点：支持高并发消息处理、实时消息同步、消息过滤、表情包支持等高级功能。新增 sendTextMessage、sendCustomMessage、appendLocalTip 接口函数。
- 业务价值：为直播平台提供核心的互动能力，增强用户参与度和直播氛围。
- 应用场景：弹幕互动、消息管理、表情包、聊天室等社交互动场景。

响应式数据

数据列表	描述
messageList	弹幕消息列表。

接口函数

函数列表	描述
sendTextMessage	发送文本消息。
sendCustomMessage	发送自定义消息。
appendLocalTip	添加本地提示。

MessageListState

消息列表管理模块

- 核心功能：管理聊天消息列表，支持消息加载、滚动控制、已读回执、消息高亮等完整的消息展示功能。
- 技术特点：支持虚拟滚动、消息优化、实时更新等高性能消息处理。新增 activeConversationID、messageList、hasMoreOlderMessage 等响应式数据。
- 业务价值：为即时通信提供核心的消息展示能力，确保良好的聊天体验。
- 应用场景：即时通信、群聊、私聊、消息管理等通信场景。

响应式数据

数据列表	描述
activeConversationID	当前活跃的会话 ID。
messageList	消息列表数据。
hasMoreOlderMessage	是否有更多旧消息。

hasMoreNewerMessage	是否有更多新消息。
enableReadReceipt	是否启用已读回执。
isDisableScroll	是否禁用滚动。
recalledMessageIDSet	已撤回消息的 ID 集合。
highlightMessageIDSet	高亮消息的 ID 集合。

接口函数

函数列表	描述
setEnableReadReceipt	设置已读回执。
setIsDisableScroll	设置滚动禁用。
highlightMessage	高亮消息。

MessageInputState

消息输入管理模块

- 核心功能：管理消息输入框的状态和行为，支持文本输入、表情包、@功能、输入状态提示等完整的输入体验。
- 技术特点：支持富文本编辑、输入状态同步、草稿保存等功能。新增 inputRawValue、isPeerTyping 响应式数据。
- 业务价值：为用户提供便捷的消息输入体验，提升沟通效率。
- 应用场景：消息编辑、表情输入、文件发送、语音输入等输入场景。

响应式数据

数据列表	描述
inputRawValue	输入框的原始文本内容。
isPeerTyping	对方是否正在输入。

接口函数

函数列表	描述
updateRawValue	更新原始值。
setEditorInstance	设置编辑器实例。
setContent	设置内容。

insertContent	插入内容。
focusEditor	聚焦编辑器。
blurEditor	失焦编辑器。
sendMessage	发送消息。

MessageActionState

消息操作管理模块

- 核心功能：管理消息的各种操作，包括转发、引用、复制、删除、撤回等完整的消息操作功能。
- 技术特点：支持批量操作、操作状态管理、权限控制等功能。新增 forwardMessageIDList、isForwardMessageSelectionDone 等响应式数据。
- 业务价值：为用户提供丰富的消息操作能力，提升使用体验。
- 应用场景：消息转发、消息引用、消息管理、批量操作等场景。

响应式数据

数据列表	描述
forwardMessageIDList	要转发的消息 ID 列表。
isForwardMessageSelectionDone	消息转发选择是否完成。
forwardConversationIDList	转发目标会话 ID 列表。
quotedMessage	被引用的消息信息。

接口函数

函数列表	描述
forwardMessage	转发消息。
setForwardMessageIDList	设置转发消息列表。
setIsForwardMessageSelectionDone	设置转发选择完成。
setForwardConversationIDList	设置转发会话列表。
quoteMessage	引用消息。
clearQuotedMessage	清除引用消息。
copyTextMessage	复制文本消息。

deleteMessage	删除消息。
recallMessage	撤回消息。
resetMessageActionState	重置消息操作状态。

ConversationListState

会话列表管理模块

- 核心功能：管理用户的会话列表，支持会话排序、未读统计、会话操作等完整的会话管理功能。
- 技术特点：支持实时会话更新、智能排序、网络状态监控等功能。新增 conversationList、activeConversation、totalUnRead、netStatus 响应式数据。
- 业务价值：为用户提供清晰的会话管理界面，提升沟通效率。
- 应用场景：会话管理、联系人列表、群组管理、消息中心等场景。

响应式数据

数据列表	描述
conversationList	会话列表数据。
activeConversation	当前活跃的会话。
totalUnRead	未读消息总数。
netStatus	网络连接状态。

接口函数

函数列表	描述
markConversationUnread	标记会话未读。
setActiveConversation	设置活跃会话。
pinConversation	置顶会话。
deleteConversation	删除会话。
muteConversation	静音会话。
setConversationDraft	设置会话草稿。
createC2CConversation	创建单聊会话。
createGroupConversation	创建群聊会话。

ContactListState

联系人管理模块

- 核心功能：管理用户的联系人列表，包括好友管理、群组管理、黑名单管理等完整的联系人功能。
- 技术特点：支持联系人分组、好友申请处理、群组申请管理等功能。新增 friendList、groupList、blackList 等响应式数据和完整的联系人操作接口。
- 业务价值：为用户提供完整的社交关系管理能力，构建社交网络。
- 应用场景：好友管理、群组管理、联系人搜索、社交网络等场景。

响应式数据

数据列表	描述
friendList	好友列表。
groupList	群组列表。
blackList	黑名单列表。
friendApplicationUnreadCount	好友申请未读数。
friendGroupList	好友分组列表。
friendApplicationList	好友申请列表。
groupApplicationList	群申请列表。

接口函数

函数列表	描述
setGroupApplicationList	设置群组申请列表。
setFriendList	设置好友列表。
setGroupList	设置群组列表。
setBlackList	设置黑名单列表。
setFriendApplicationUnreadCount	设置好友申请未读数。
setFriendGroupList	设置好友分组列表。
setFriendApplicationList	设置好友申请列表。
initContactListWatcher	初始化联系人监听器。

addFriend	添加好友。
markFriendApplicationAsRead	标记好友申请为已读。
acceptFriendApplication	接受好友申请。
refuseFriendApplication	拒绝好友申请。
addToBlacklist	添加到黑名单。
removeFromBlacklist	从黑名单移除。
deleteFriend	删除好友。
setFriendRemark	设置好友备注。
createFriendGroup	创建好友分组。
deleteFriendGroup	删除好友分组。
addToFriendGroup	添加到好友分组。
removeFromFriendGroup	从好友分组移除。
renameFriendGroup	重命名好友分组。
joinGroup	加入群组。
acceptGroupApplication	接受群组申请。
refuseGroupApplication	拒绝群组申请。

C2CSettingState

单聊设置管理模块

- 核心功能：管理单聊会话的各种设置，包括用户信息、聊天设置、权限控制等功能。
- 技术特点：支持实时设置同步、权限管理、个性化配置等功能。更新响应式数据为 userID、avatar、signature 等标准化字段。
- 业务价值：为用户提供个性化的单聊体验，提升沟通质量。
- 应用场景：单聊设置、用户信息管理、聊天权限控制等场景。

响应式数据

数据列表	描述
currentConversationRef	当前会话引用。

userIDRef	用户 ID。
nickRef	用户昵称。
avatarRef	用户头像。
signatureRef	用户个性签名。
remarkRef	好友备注名。
isMutedRef	会话静音状态。
isPinnedRef	会话置顶状态。
isContactRef	好友关系状态。
userID	用户 ID。
avatar	用户头像 URL。
signature	用户个性签名。
remark	用户备注名。
isMuted	是否已静音。
isPinned	是否已置顶。
isContact	是否为联系人。

接口函数

函数列表	描述
setChatPinned	设置聊天置顶。
setChatMuted	设置聊天免打扰。
setUserRemark	设置用户备注。

GroupSettingState

群聊设置管理模块

- 核心功能：管理群聊的各种设置和操作，包括群信息管理、成员管理、权限控制等完整的群管理功能。
- 技术特点：支持群权限管理、成员操作、群设置同步等功能。新增 groupID、groupType、groupName 等完整的群信息响应式数据和管理接口。
- 业务价值：为群主和管理员提供完整的群管理能力，维护群秩序。

- 应用场景：群管理、成员管理、权限控制、群设置等场景。

响应式数据

数据列表	描述
groupID	群组 ID。
groupType	群组类型。
groupName	群组名称。
avatar	用户头像 URL。
introduction	群组介绍。
notification	群组公告。
isMuted	是否已静音。
isPinned	是否已置顶。
groupOwner	群主信息。
adminMembers	管理员列表。
allMembers	所有成员列表。
memberCount	成员总数。
maxMemberCount	最大成员数。
currentUserID	当前用户 ID。
currentUserRole	当前用户角色。
nameCard	用户名片。
isMuteAllMembers	是否禁言所有成员。
isInGroup	是否在群组中。
inviteOption	邀请选项。

接口函数

函数列表	描述
getGroupMemberList	获取群成员列表。

updateGroupProfile	更新群资料。
addGroupMember	添加群成员。
deleteGroupMember	删除群成员。
changeGroupOwner	转让群主。
setGroupMemberRole	设置群成员角色。
setGroupMemberNameCard	设置群成员名片。
setChatPinned	设置聊天置顶。
setChatMuted	设置聊天免打扰。
setGroupMemberMuteTime	设置群成员禁言。
setMuteAllMember	设置全员禁言。
dismissGroup	解散群组。
quitGroup	退出群组。
hasPermission	检查权限。
canOperateOnMember	检查成员操作权限。
getAvailablePermissions	获取可用权限。
initWatcher	初始化监听器。

VideoMixerState

视频混流管理模块

- 核心功能：管理视频混流功能，支持多路视频合成、布局管理、媒体源控制等高级视频处理功能。
- 技术特点：支持实时视频混流、动态布局调整、媒体源管理等功能。新增 isVideoMixerEnabled、mediaSourceList、activeMediaSource 响应式数据和完整的混流控制接口。
- 业务价值：为直播平台提供专业的视频制作能力，提升直播质量。
- 应用场景：多人直播、画中画、视频合成、专业制作等场景。

响应式数据

数据列表	描述
publishVideoQuality	根据分辨率获取尺寸。

isVideoMixerEnabled	视频混流是否启用。
mediaSourceList	媒体源列表。
activeMediaSource	当前活跃的媒体源。

接口函数

函数列表	描述
getVideoDataByQuality	根据质量获取视频数据。
getSizeByQuality	根据质量获取尺寸。
getSizeByResolution	根据分辨率获取尺寸。
transformTUIVideoQualityToTRTCVideoResolution	转换视频质量到 TRTC 分辨率。
transformTRTCVideoResolutionToTUIVideoQuality	转换 TRTC 分辨率到视频质量。
transformTRTCVideoResModeToTUIVideoResMode	转换 TRTC 分辨率模式。
changeActiveMediaSource	切换活跃媒体源。
updateVideoQuality	更新视频质量。
getDefaultLayoutByMediaSource	根据媒体源获取默认布局。
addMediaSource	添加媒体源。
updateMediaSource	更新媒体源。
removeMediaSource	移除媒体源。
enableLocalVideoMixer	启用本地视频混流。
clearMediaSource	清空所有媒体源。
initMediaSourceManager	初始化媒体源管理器。
initVideoMixerState	初始化视频混流状态。

VirtualBackgroundState

虚拟背景管理模块

- 核心功能：管理虚拟背景功能，支持背景替换、背景模糊、自定义背景等视频美化功能。
- 技术特点：基于 AI 技术实现实时背景分割和替换。新增 virtualBackgroundConfig 响应式数据和 isSupported、setVirtualBackground 接口函数。

- 业务价值：为用户提供隐私保护和视频美化功能，提升视频通话体验。
- 应用场景：视频通话、在线会议、直播美化、隐私保护等场景。

响应式数据

数据列表	描述
virtualBackgroundConfig	虚拟背景配置。

接口函数

函数列表	描述
initVirtualBackground	初始化虚拟背景。
saveVirtualBackground	保存虚拟背景。
isSupported	检查是否支持。
setVirtualBackground	设置虚拟背景。

ASRState

语音识别管理模块

- 核心功能：管理语音识别功能，支持实时语音转文字、转录历史管理、转录导出等功能。
- 技术特点：基于先进的 ASR 技术，支持多语言识别、实时转录、历史记录等功能。新增 recentTranscripts、transcriptHistory 响应式数据和 exportTranscripts 接口。
- 业务价值：为用户提供语音转文字服务，提升沟通效率和可访问性。
- 应用场景：会议记录、语音转录、无障碍通信、内容记录等场景。

响应式数据

数据列表	描述
recentTranscripts	最近的语音转录记录。
transcriptHistory	语音转录历史记录。

接口函数

函数列表	描述
setRecentTranscriptsDuration	设置最近转录时长。
clearHistory	清空转录历史记录。

exportTranscripts	导出转录内容。

SearchState

搜索功能管理模块

- 核心功能：管理全局搜索功能，支持消息搜索、用户搜索、群组搜索等多维度搜索能力。
- 技术特点：支持高级搜索、搜索历史、搜索建议等功能。重构响应式数据为 keyword、results、isLoading 等标准化字段，提供完整的搜索接口。
- 业务价值：为用户提供快速查找信息的能力，提升使用效率。
- 应用场景：消息搜索、联系人搜索、内容查找、历史记录等场景。

响应式数据

数据列表	描述
keyword	搜索关键词。
results	搜索结果。
isLoading	是否正在加载。
error	错误信息。
searchAdvancedParams	高级搜索参数。
selectedSearchType	选中的搜索类型。

接口函数

函数列表	描述
search	搜索。
setKeyword	设置关键词。
setSelectedType	设置选中类型。
setSearchMessageAdvancedParams	设置消息搜索高级参数。
setSearchUserAdvancedParams	设置用户搜索高级参数。
setSearchGroupAdvancedParams	设置群组搜索高级参数。
loadMore	加载更多。

SeatStore

座位存储管理模块

- 核心功能：提供座位状态的底层存储和管理，支持座位信息缓存、用户信息映射、设备请求处理等核心功能。
- 技术特点：采用响应式存储设计，支持实时状态同步、事件驱动更新等功能。新增 liveOwnerUserId、localUserId、seatList 等响应式数据和 getUserInfo、convertUserInfoToAudienceInfo 接口。
- 业务价值：为座位管理提供可靠的数据基础，确保座位状态的一致性。
- 应用场景：座位状态管理、用户信息存储、设备状态跟踪等底层场景。

响应式数据

数据列表	描述
liveOwnerUserId	直播间主播用户 ID。
localUserId	本地用户 ID。
seatList	麦位列表，包含所有麦位的状态和用户信息。
coHostUserList	连麦主播列表。
sentDeviceRequestMap	已发送的设备请求映射。
receivedDeviceRequestMap	已接收的设备请求映射。
userInfoMap	用户信息映射表。

接口函数

函数列表	描述
getUserInfo	获取用户信息。
convertUserInfoToAudienceInfo	转换用户信息为观众信息。

React Native

最近更新时间：2026-04-20 17:34:07

AtomicXCore SDK 是腾讯云最新推出的面向视频直播、语聊房等场景的全新一代基于响应式的 API，您可以非常快速地基于这组 API 构建自己的 UI 页面，它支持房间管理、屏幕分享、成员管理、麦位控制、基础美颜等丰富功能，同时基于 TRTC SDK，能够提供超低延时、高品质的音视频体验，本页面包含 AtomicXCore SDK 的所有 API 接口，按功能模块分类展示。

LoginState

用户身份认证与登录管理模块
核心功能：负责用户身份验证、登录状态管理、用户信息维护等基础认证服务。
响应式数据

数据列表	描述
loginUserInfo	当前登录用户信息。
loginStatus	当前登录状态。

接口函数

函数列表	描述
login	登录方法。
logout	登出方法。
setSelfInfo	设置用户信息。

LiveListState

直播列表管理模块

- 核心功能：管理直播间的完整生命周期，包括创建、加入、离开、结束等核心业务流程。
- 技术特点：支持分页加载、实时状态同步、直播信息动态更新，采用响应式数据管理，确保 UI 与数据状态实时同步。
- 业务价值：为直播平台提供核心的直播间管理能力，支持大规模并发直播场景，是直播业务的基础设施。
- 应用场景：直播列表展示、直播间创建、直播状态管理、直播数据统计等核心业务场景。

响应式数据

数据列表	描述

liveList	直播列表数据。
liveListCursor	直播列表游标，用于分页加载。
currentLive	当前直播信息。

接口函数

函数列表	描述
fetchLiveList	获取直播列表。
createLive	创建直播间。
joinLive	加入直播间。
leaveLive	离开直播间。
endLive	结束直播。
updateLiveInfo	更新直播信息。
addLiveListListener	添加直播列表事件监听。
removeLiveListListener	移除直播列表事件监听。

LiveSeatState

直播间座位管理模块

- 核心功能：实现多人连麦场景下的座位控制，支持复杂的座位状态管理和音视频设备控制。
- 技术特点：基于 WebRTC 技术，支持多路音视频流管理，提供座位锁定、设备控制、权限管理等高级功能。
- 业务价值：为多人互动直播提供核心技术支撑，支持PK、连麦、多人游戏等丰富的互动场景。
- 应用场景：多人连麦、主播 PK、互动游戏、在线教育、会议直播等需要多人音视频互动的场景。

响应式数据

数据列表	描述
seatList	座位列表。
canvas	画布信息。
speakingUsers	正在说话的用户列表。

接口函数

函数列表	描述
takeSeat	用户上麦。
leaveSeat	用户下麦。
muteMicrophone	静音麦克风。
unmuteMicrophone	取消静音麦克风。
kickUserOutOfSeat	将用户踢出麦位。
moveUserToSeat	将用户移动到麦位。
lockSeat	锁定麦位。
unlockSeat	解锁麦位。
openRemoteCamera	开启远端用户摄像头。
closeRemoteCamera	关闭远端用户摄像头。
openRemoteMicrophone	开启远端用户麦克风。
closeRemoteMicrophone	关闭远端用户麦克风。
addLiveSeatEventListener	添加麦位事件监听。
removeLiveSeatEventListener	移除麦位事件监听。

LiveAudienceState

直播间观众管理模块

- 核心功能：管理直播间观众列表，提供观众权限控制、管理员设置等直播间秩序维护功能。
- 技术特点：支持实时观众列表更新、权限分级管理、批量操作等高级功能，确保直播间秩序和用户体验。
- 业务价值：为直播平台提供完整的观众管理解决方案，支持大规模观众场景下的秩序维护。
- 应用场景：观众管理、权限控制、直播间秩序维护、观众互动管理等核心业务场景。

响应式数据

数据列表	描述
audienceList	直播间观众列表。
audienceCount	直播间观众数量。

接口函数

函数列表	描述
fetchAudienceList	获取观众列表。
setAdministrator	设置管理员。
revokeAdministrator	撤销管理员。
kickUserOutOfRoom	将用户踢出房间。
disableSendMessage	禁用发送消息。
addAudienceListener	添加观众监听。
removeAudienceListener	移除观众监听。

DeviceState

设备状态管理模块

- 核心功能：管理摄像头、麦克风等音视频设备的控制，提供设备状态监控、权限检查等基础设备服务。
- 技术特点：支持多设备管理、设备状态实时监控、权限动态检查、设备故障自动恢复等高级功能。
- 业务价值：为直播系统提供稳定的设备基础，确保音视频采集的可靠性和用户体验。
- 应用场景：设备管理、权限控制、音视频采集、设备故障处理等基础技术场景。

响应式数据

数据列表	描述
microphoneStatus	麦克风开启状态。
microphoneLastError	麦克风最后一次错误状态。
hasPublishAudioPermission	是否有音频发布权限。
captureVolume	采集音量大小（0-100）。
currentMicVolume	当前麦克风音量（0-100）。
outputVolume	输出音量大小（0-100）。
cameraStatus	摄像头开启状态。
cameraLastError	摄像头最后一次错误状态。

isFrontCamera	是否为前置摄像头。
localMirrorType	本地镜像类型。
localVideoQuality	本地视频质量设置。
currentAudioRoute	当前音频输出路由（扬声器/耳机）。
screenStatus	屏幕共享状态。
networkInfo	网络信息状态。

接口函数

函数列表	描述
openLocalMicrophone	开启本地麦克风。
closeLocalMicrophone	关闭本地麦克风。
setCaptureVolume	设置采集音量。
setOutputVolume	设置输出音量。
openLocalCamera	开启本地摄像头。

CoGuestState

连麦嘉宾管理模块

- 核心功能：处理观众与主播之间的连麦互动，管理连麦申请、邀请、接受、拒绝等完整的连麦流程。
- 技术特点：基于实时音视频技术，支持连麦状态实时同步、音视频质量自适应、网络状况监控等高级功能。
- 业务价值：为直播平台提供观众参与互动的核心能力，增强用户粘性和直播趣味性。
- 应用场景：观众连麦、互动问答、在线K歌、游戏直播等需要观众参与的互动场景。

响应式数据

数据列表	描述
connected	已连接的连麦嘉宾列表。
invitees	被邀请上麦的用户列表。
applicants	申请上麦的用户列表。
candidates	可邀请上麦的候选用户列表。

接口函数

函数列表	描述
applyForSeat	申请连麦座位。
cancelApplication	取消申请。
acceptApplication	接受申请。
rejectApplication	拒绝申请。
inviteToSeat	邀请上麦。
cancelInvitation	取消邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
disconnect	断开连麦连接。
addCoGuestGuestListener	添加连麦嘉宾侧事件监听。
removeCoGuestGuestListener	移除连麦嘉宾侧事件监听。
addCoGuestHostListener	添加连麦主播侧事件监听。
removeCoGuestHostListener	移除连麦主播侧事件监听。

CoHostState

连线主播管理模块

- **核心功能：实现主播间的连线功能，支持主播邀请、连线申请、连线状态管理等主播间互动功能。**
- **技术特点：支持多主播音视频同步、画中画显示、音视频质量优化等高级技术，确保连麦体验的流畅性。**
- **业务价值：为直播平台提供主播间协作的核心能力，支持 PK、合作直播等高级业务场景。**
- **应用场景：主播 PK、合作直播、跨平台连麦、主播互动等高级直播场景。**

响应式数据

数据列表	描述
connected	已连接的连麦主播列表。
invitees	被邀请连线的主播列表。
applicant	当前申请连线的主播信息。

candidates	可邀请连线的候选主播列表。
coHostStatus	当前连线状态。

接口函数

函数列表	描述
requestHostConnection	请求连线。
cancelHostConnection	取消连线请求。
acceptHostConnection	接受连线请求。
rejectHostConnection	拒绝连线请求。
exitHostConnection	退出连线。
addCoHostListener	添加连线主播事件监听。
removeCoHostListener	移除连线主播事件监听。

AudioEffectState

音效处理模块

- 核心功能：提供变声、混响、耳返等高级音效功能，支持多种音效效果和实时音效调节。
- 技术特点：基于腾讯天籁实验室的音频处理算法，支持实时音效处理、低延迟音频传输、音质优化等高级技术。
- 业务价值：为直播平台提供差异化的音效体验，增强用户参与度和直播趣味性。
- 应用场景：变声直播、K 歌直播、音效娱乐、专业音效等需要音频处理的场景。

响应式数据

数据列表	描述
isEarMonitorOpened	耳返开关状态。
earMonitorVolume	耳返音量大小。
audioChangerType	变声状态。
audioReverbType	混响状态。

接口函数

函数列表	描述

setAudioChangerType	设置变声效果。
setAudioReverbType	设置混响效果。
setVoiceEarMonitorEnable	设置耳返开关状态。
setVoiceEarMonitorVolume	设置耳返音量大小。

BarrageState

弹幕消息管理模块

- 核心功能：处理直播间内的文本消息、自定义消息等弹幕功能，支持弹幕发送、消息状态同步等完整弹幕系统。
- 技术特点：支持高并发消息处理、实时消息同步、消息过滤、表情包支持等高级功能。
- 业务价值：为直播平台提供核心的互动能力，增强用户参与度和直播氛围。
- 应用场景：弹幕互动、消息管理、表情包、聊天室等社交互动场景。

响应式数据

数据列表	描述
messageList	当前房间的弹幕消息列表。
allowSendMessage	是否允许发送消息。

接口函数

函数列表	描述
sendTextMessage	发送文本类型弹幕。
sendCustomMessage	发送自定义类型弹幕。

BaseBeautyState

基础美颜模块

- 核心功能：提供磨皮、美白、红润等基础美颜效果调节，支持实时美颜参数调整。
- 技术特点：基于 AI 美颜算法，支持实时美颜处理、参数平滑调节、性能优化等高级技术。
- 业务价值：为直播平台提供基础的美颜能力，提升用户形象和直播质量。
- 应用场景：美颜直播、形象优化、美颜调节、直播美化等需要美颜功能的场景。

响应式数据

数据列表	描述

smoothLevel	磨皮级别 取值范围[0,9]: 0 表示关闭，9 表示效果最明显。
whitenessLevel	美白级别 取值范围[0,9]: 0 表示关闭，9 表示效果最明显。
ruddyLevel	红润级别 取值范围[0,9]: 0 表示关闭，9 表示效果最明显。

接口函数

函数列表	描述
setSmoothLevel	设置磨皮级别。
setWhitenessLevel	设置美白级别。
setRuddyLevel	设置红润级别。

GiftState

礼物系统管理模块

- 核心功能：处理礼物的发送、接收、礼物列表管理等功能，支持礼物分类、礼物动画、礼物统计等完整礼物经济系统。
- 技术特点：支持礼物动画渲染、礼物特效处理、礼物统计、礼物排行榜等高级功能。
- 业务价值：为直播平台提供核心的变现能力，支持礼物经济、虚拟货币等商业模式。 应用场景：礼物打赏、虚拟货币、礼物特效、礼物统计等商业化场景。

响应式数据

数据列表	描述
usableGifts	可用礼物列表。

接口函数

函数列表	描述
refreshUsableGifts	刷新可用礼物列表。
sendGift	发送礼物。
addGiftListener	添加礼物事件监听器。
removeGiftListener	移除礼物事件监听器。

LikeState

点赞互动模块

- 核心功能：处理直播间的点赞功能，支持点赞发送、点赞统计、点赞事件监听等互动功能。
- 技术特点：支持高并发点赞处理、实时点赞统计、点赞动画效果、点赞排行榜等高级功能。
- 业务价值：为直播平台提供基础的互动能力，增强用户参与度和直播氛围。
- 应用场景：点赞互动、人气统计、互动效果、用户参与等基础互动场景。

响应式数据

数据列表	描述
totalLikeCount	总点赞数量。

接口函数

函数列表	描述
sendLike	发送点赞。
addLikeListener	添加点赞事件监听。
removeLikeListener	移除点赞事件监听。