实时音视频

最佳实践

文档版本 01

发布日期 2024-08-01

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

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

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

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 云端录制与回放

1.1 概述

华为云实时音视频服务提供的云端录制回放功能适用于需要将音视频通话或互动直播过程进行录制和存储的业务场景。云端录制功能包含两种模式，如表1-1所示。

表 1-1 云端录制模式

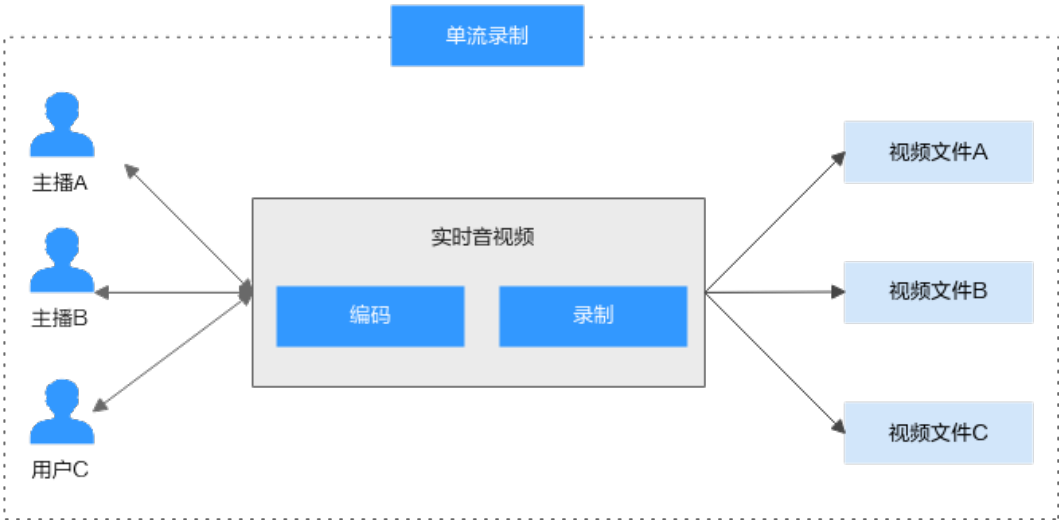
模式	说明	适用场景
单流录制	• 支持单流录制，可按用户进行录制。 • 支持自启动录制，开启后对房间内每一个流进行录制。 • 录制指定的媒体类型，支持仅录制音频、仅录制视频、同时录制音视频。 • 支持录制MP4、HLS文件，音频编码类型支持AAC。 • 支持获取回调消息中的播放URL进行回放，回调消息中的downloadurl字段为OBS播放地址，使用该URL播放将会在OBS中产生对应的下载流量或者带宽费用。 • 支持选择摄像头流或屏幕分享流。 • 支持指定分辨率大小。	在线课堂、内容审核等

模式	说明	适用场景
合流录制	• 支持合流录制，可多路视频或多路音频、视频合流录制。 • 录制指定的媒体类型，支持仅录制音频、仅录制视频、同时录制音视频。 • 支持录制MP4、HLS文件，音频编码类型支持AAC。 • 设置音视频属性，支持设置音视频属性，如码率、分辨率、帧率等。 • 支持获取回调消息中的播放URL进行回放，回调消息中的downloadurl字段为OBS播放地址，使用该URL播放将会在OBS中产生对应的下载流量或者带宽费用。	连麦直播等

1.2 单流录制

场景说明

将房间中的每一个用户的音视频流分别录制成独立的文件。



录制机制

SparkRTC提供的**单流录制**支持自启动模式，即**单流自动录制**，具体的实现机制如下图所示。

图 1-1 单流录制

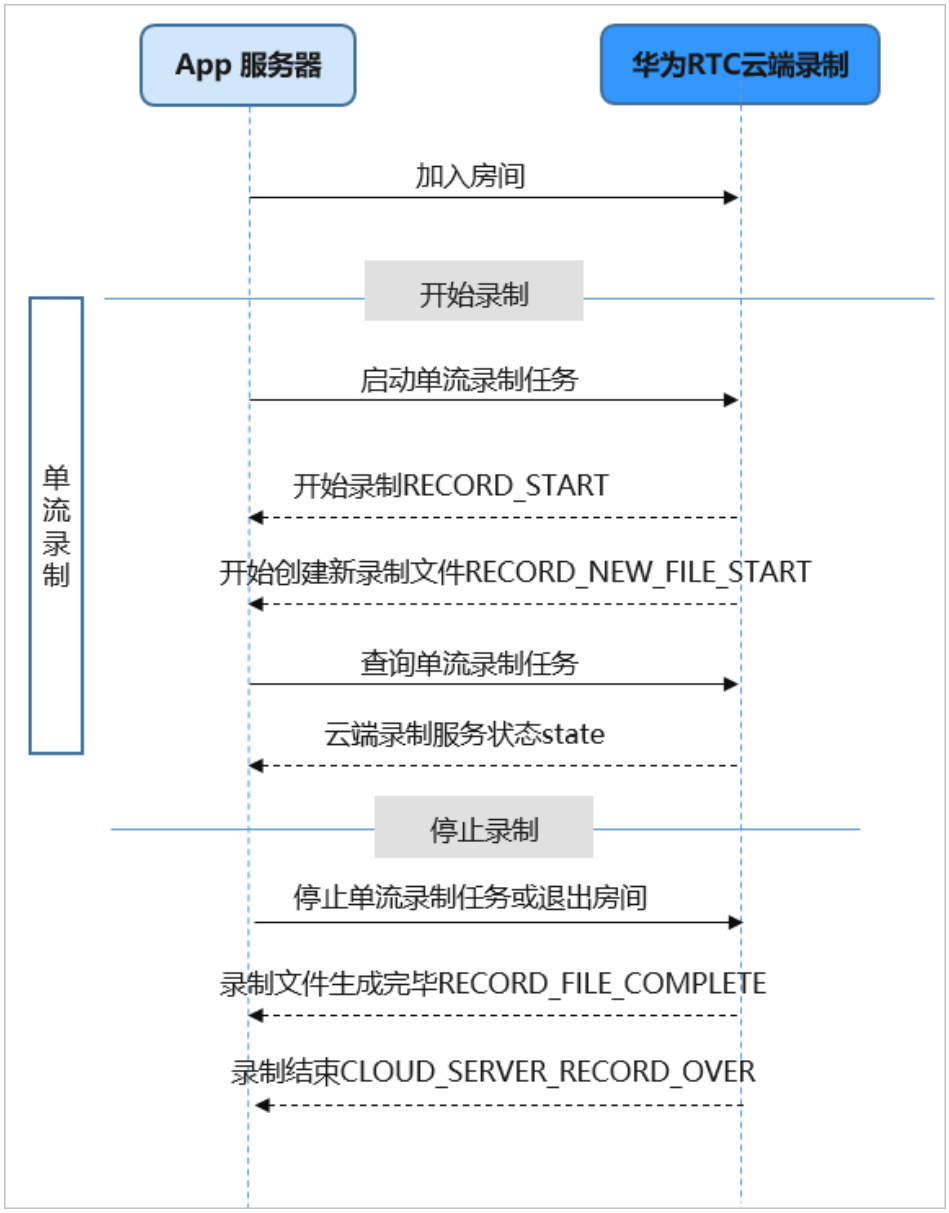
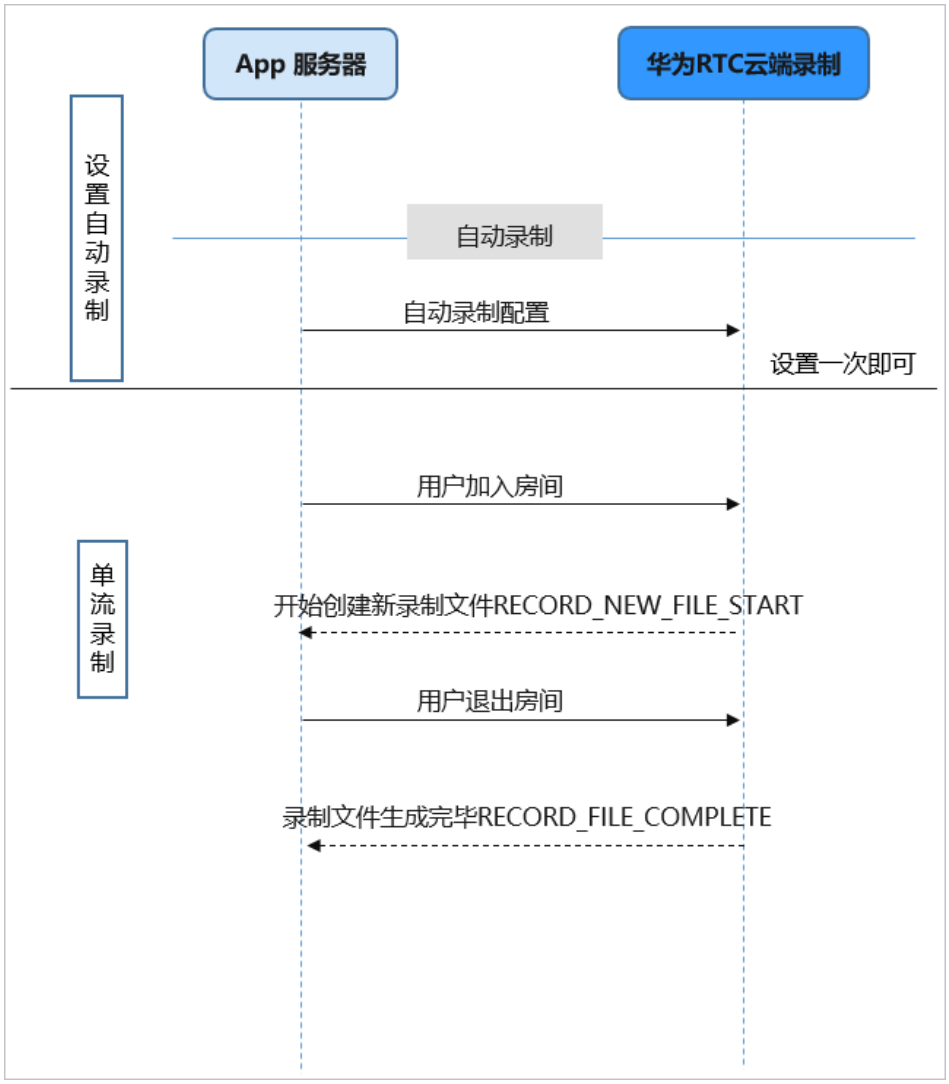
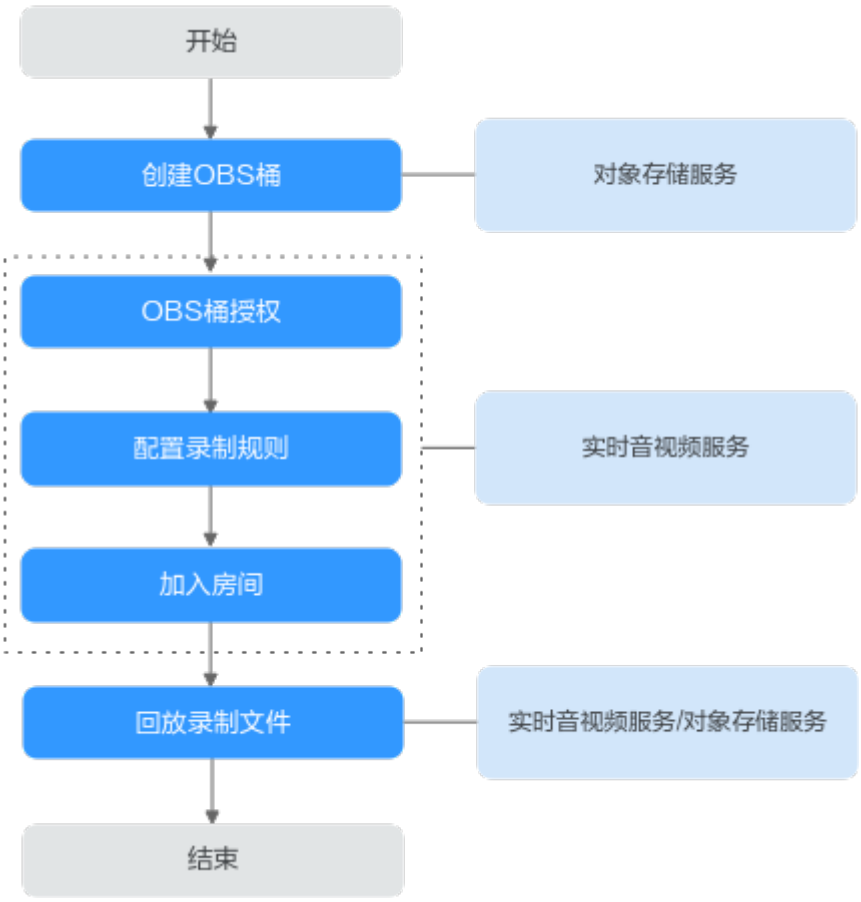


图 1-2 单流自动录制



实现流程



1. **创建OBS桶**：创建用于存储SparkRTC录制文件的OBS桶，若已有OBS桶，请直接执行2。

   说明

   由于单AZ桶的可靠性低于多AZ桶，为避免因OBS服务异常导致录制失败，建议您创建多AZ桶用于录制文件的存储。

2. **OBS桶授权**：在SparkRTC服务中对存储录制文件的OBS桶进行授权，允许SparkRTC服务将录制文件存储在对应的OBS桶中。
3. **配置录制规则**：为实时音视频互动配置录制规则，并开启自动录制功能，加入SparkRTC房间后，与应用中的**录制规则ID**相同的录制模板会自动生效，录制内容按录制设置存储至OBS中。还可以通过设置回调地址获取录制任务状态通知。
4. **加入房间**：录制规则配置完成后，您可以通过SparkRTC APP加入某个SparkRTC房间进行音视频互动，SparkRTC会根据配置的录制规则对正在直播的音视频进行录制。

   说明

   若配置录制规则时，未开启自动录制功能，则加入SparkRTC房间后，您需要调用**实时音视频API**开启、查询和控制云端录制任务。SparkRTC服务才会根据接口中的**录制规则ID**对实时音视频互动画面进行录制。

5. **回放录制文件**：录制完成后，在已配置的回调地址中会收到录制任务的回调消息，您可以获取到录制文件的基本信息，也可以在OBS中管理录制文件，如下载、分享、删除等。

说明

录制文件的分辨率与推流分辨率相关，按推流原分辨率进行录制。

实现步骤

步骤1 请参见**OBS帮助中心**创建桶。若已有OBS桶，请直接执行**步骤2**。

说明

创建的OBS桶所在区域必须为**华北-北京四**。

步骤2 OBS桶授权。

1. 登录实时音视频控制台。
2. 在左侧导航树中选择“云资源授权”，进入桶授权页面。
3. 在对应的OBS桶行单击“授权”，完成桶授权。

步骤3 配置录制规则。

1. 登录实时音视频控制台。
2. 在左侧导航树中选择“应用管理”，进入应用管理页面。
3. 在需要创建录制规则的应用行单击“录制配置”，进入录制配置页面。
4. 在“录制规则”页签，单击“添加”，进入添加录制规则页面。

说明

一个应用ID仅支持创建一个录制规则。

5. 请您按照实际需求配置录制参数，参数说明如**表1-2**所示。

表 1-2 录制参数说明

参数名		描述
存储-桶		存储录制文件的OBS桶。 目前录制文件仅支持存储到**华北-北京四**的OBS桶中。
区域		OBS桶所在的区域。
存储-路径		存储录制文件的OBS桶路径。
录制格式		录制文件的格式，支持HLS和MP4文件格式。
HLS规则	m3u8命名规则	录制m3u8文件的存储路径和文件的前缀。 默认命名格式： {app_id}/{record_format}/{stream}_{file_start_time}/{stream}_{file_start_time} 上述特殊变量的含义如下： - app_id：应用ID。 - record_format：录制格式。 - stream：流名。 - file_start_time：文件生成时间。

参数名		描述
	录制周期	录制时长支持0-720分钟，最小录制周期为1分钟，最大录制周期为12小时，超过12小时，系统将按照命名规则生成新文件。如果录制周期为0，则整个流录制为一个文件。
	最大断流合并时长	支持如下三种配置： – 断流后生成新文件：是指录制的直播流中断后，会立即生成新的录制文件。 – 断流后不生成新文件：是指录制的直播流中断后，会和之前录制的文件合并为一个文件。最大断流合并时长为30天。 – 其他数值：是指录制的直播流中断时间在设置范围内，则和之前录制的文件合并为一个文件，否则，生成新的录制文件。
MP4规则	mp4命名规则	录制mp4文件的存储路径和文件的前缀。 默认命名格式： {app_id}/{record_format}/{stream}_{file_start_time}/{stream}_{file_start_time} 上述特殊变量的含义如下： – app_id：应用ID。 – record_format：录制格式。 – stream：流名。 – file_start_time：文件生成时间。
	录制周期	录制时长支持1-180分钟，最小录制周期为1分钟，最大录制周期为3小时，超过3小时，系统将按照命名规则生成新文件。
	最大断流合并时长	支持如下两种配置： – 断流后生成新文件：是指录制的直播流中断后，会立即生成新的录制文件。 – 其他数值：是指录制的直播流中断时间在设置范围内，则和之前录制的文件合并为一个文件，否则，生成新的录制文件。

6. 单击“确定”，在录制规则列表中会增加一条新的录制规则。

图 1-3 录制规则

7. 您可以在录制规则列表中，根据实际需求选择是否开启自动录制功能。自动录制功能开启后，若该应用下有新创建的房间，则会按照已配置的录制规则自动对该房间中的实时音视频互动过程进行录制。

> **说明**
>
> 自动录制功能开启后，仅对同一应用下新创建的房间生效，自动录制功能开启前已创建的房间不生效。

步骤4 加入房间。

录制规则配置完成后，您可以通过SparkRTC APP加入某个SparkRTC房间进行音视频互动，SparkRTC服务会根据配置的录制规则对正在直播的音视频进行录制。

> **说明**
>
> 若配置录制规则时，未开启自动录制功能，则加入SparkRTC房间后，您需要调用**实时音视频API**开启云端录制任务，SparkRTC才会根据API中的**录制规则ID**对实时音视频互动画面进行录制。

步骤5 回放录制文件。

录制完成后，您可以在OBS控制台中或通过回调消息查看录制文件。

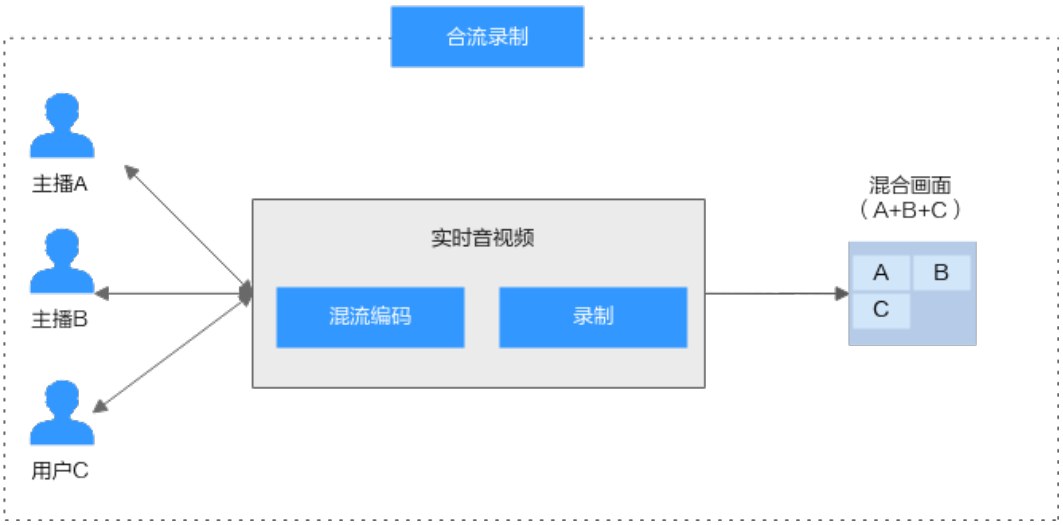
- 通过OBS控制台查看录制文件
  a. 在OBS管理控制台左侧导航栏选择“对象存储”。
  b. 在桶列表中单击存储SparkRTC录制文件的桶，进入“概览”页面。
  c. 在左侧导航栏，单击“对象”，查看录制文件信息。
  d. 您还可以对录制文件进行下载、分享等操作，具体请参见**OBS帮助中心**。

----结束

1.3 合流录制

场景说明

将房间中的多路音视频进行云端混流，再将混合后的音视频流录制成一个文件。

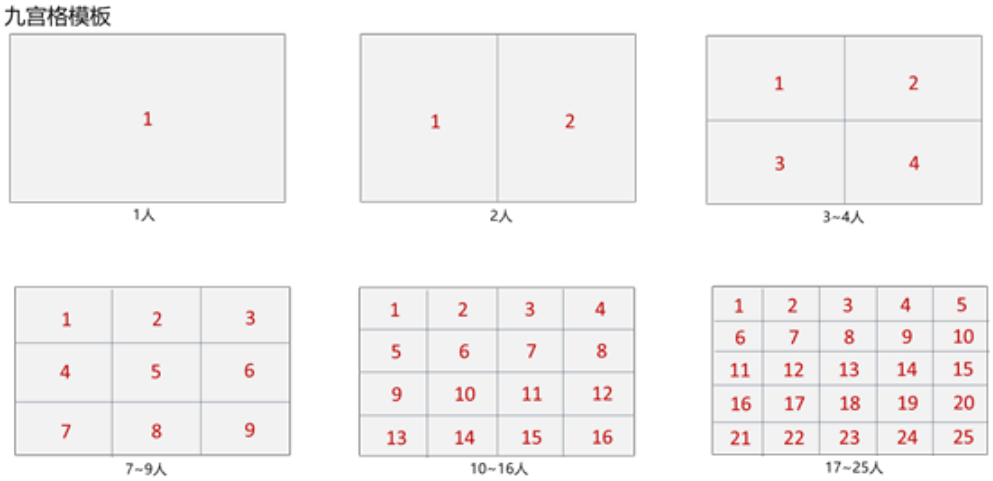


合流布局

在合流录制模式下，支持分屏九宫格模板和共享屏幕模板（主视窗居左/主视窗居右）两种预设合流布局。同时，也支持用户自定义合流布局样式（自定义视频窗格位置）。

- **九宫格模板**

  每个用户画面平铺在画布上，大小一致。根据用户数量，动态调整每个画面的大小和位置。最多支持25个画面。不同人数的实际布局效果如下图所示。

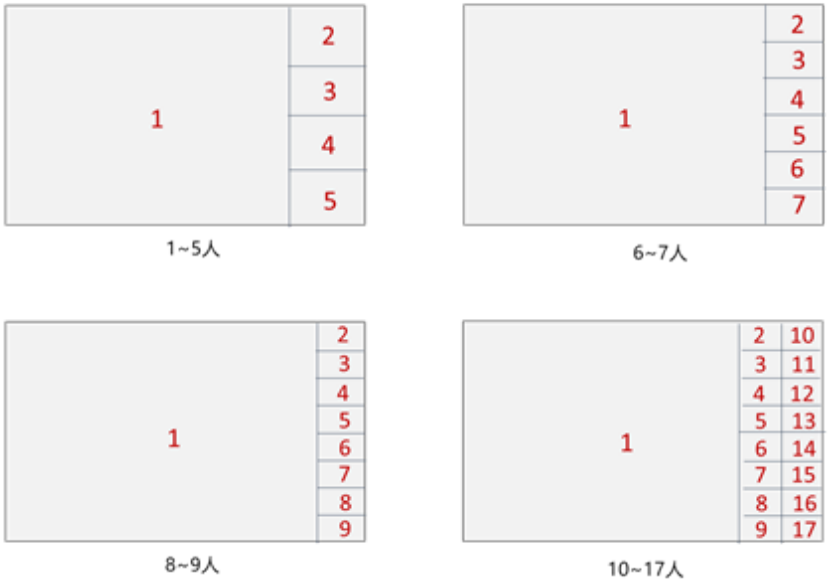


  - 如果实际视频流的宽高比与视窗的宽高比不一致，视频画面会裁剪以适配视窗的大小。
  - 中途有流退出房间，则该条流的画面会被后进入房间的流顶替。
  - 如果房间内的人数不足，则剩余位置显示背景色。
  - 如果用户只发送音频，仍然会占用画面位置。
  - 支持背景图，如果房间内人数不足，显示背景图。

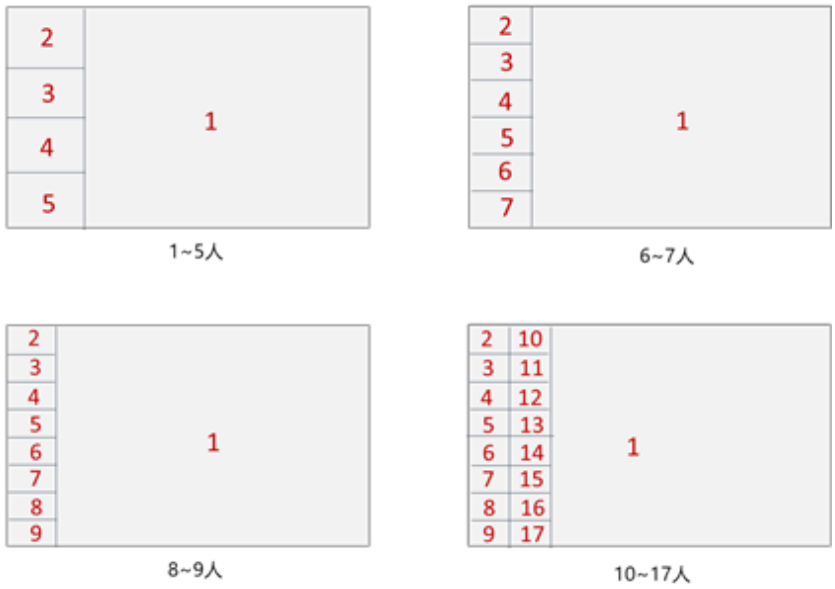
- **屏幕共享模板**

  屏幕分享（或者主讲人摄像头画面）始终占据屏幕左侧或者右侧大画面位置，其他用户依次垂直排列于旁边。最多支持17个画面。不同人数的实际布局效果如下图所示。

屏幕共享模板-大视窗居左——screen_share_left



屏幕共享模板-大视窗右——screen_share_right

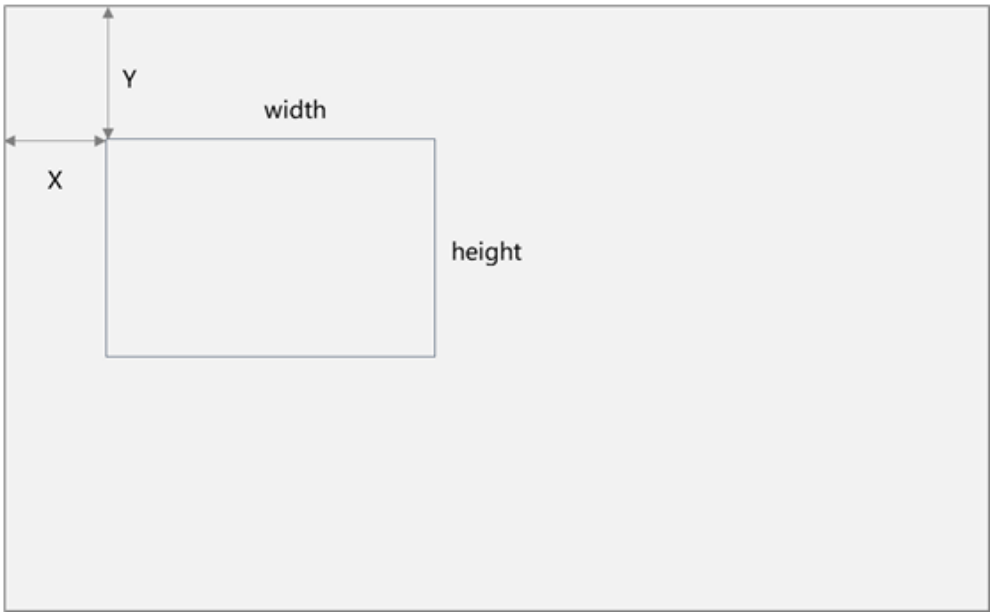


- 大视窗画面，可选择主讲摄像头流或共享屏幕流。
- 大视窗画面，显示指定的UID用户的视频，如果未指定或者指定用户未进入频道，大视窗区域显示背景色。
- 左侧大视窗为了保持内容的完整性采用缩放方式处理，右侧小视窗采用裁剪方式适配视窗的大小。
- 右侧小视窗画面按照加入房间的时间先后顺序排列。
- 右侧小视窗有流退出房间，则该条流的画面会被后进入房间的流顶替。
- 如果房间内的人数不足，则剩余位置显示背景色。
- 如果用户只发送音频，仍然会占用画面位置。

- 支持背景图，如果房间内人数不足，显示背景图。

- **自定义布局模板**

  支持用户自定义合流布局样式，可灵活设置用户画面的大小，指定用户画面在视频画布上的相对位置。

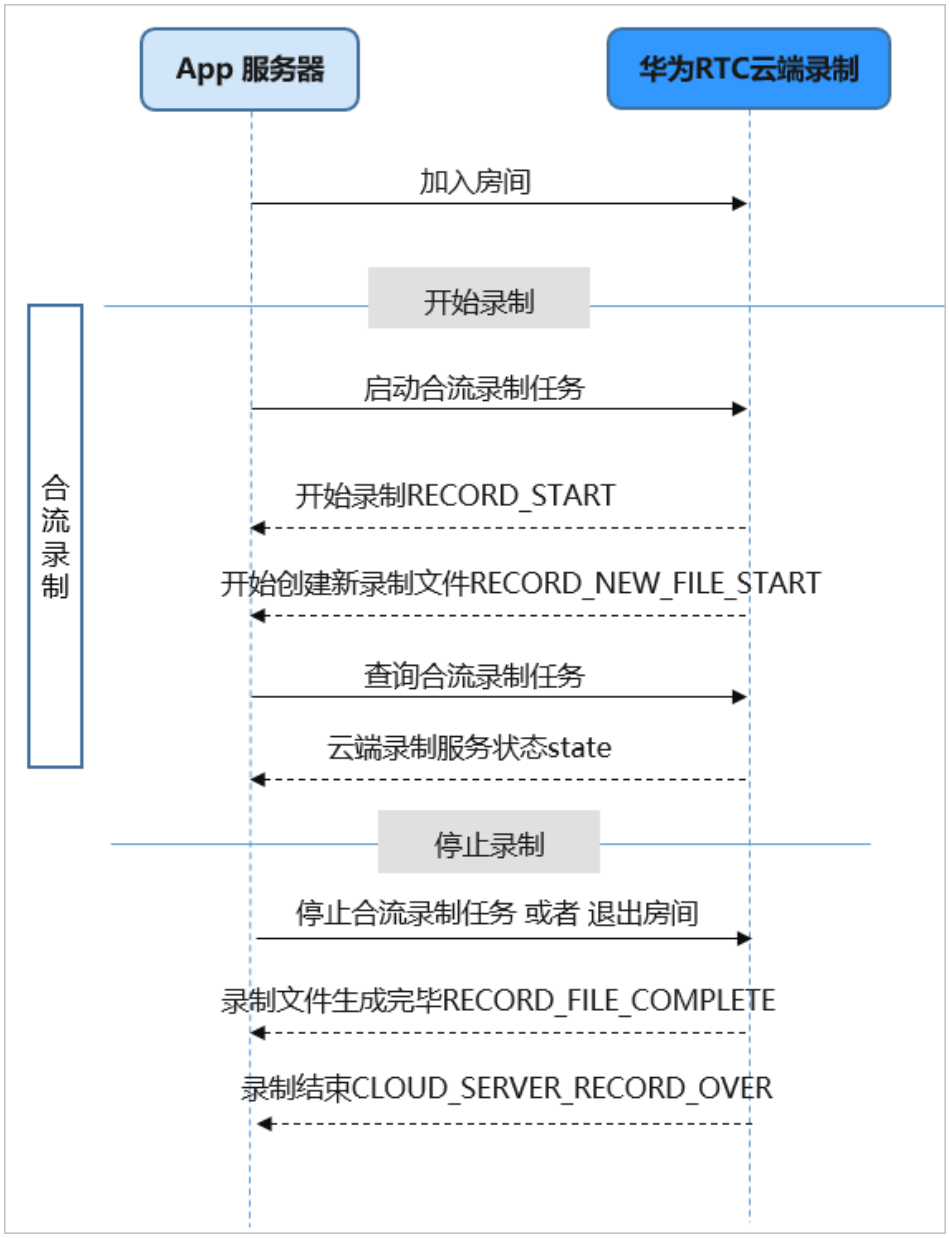
  - 可自定义各个视频窗格在画布上的位置。
  - 可自定义各个视频窗格的宽和高。
  - 针对每一个窗格，可通过user_id指定显示房间内某一用户。
  - 针对窗格，可自定义选择呈现摄像头流或者屏幕分享流。
  - 如果实际视频流的宽高比与视窗的宽高比不一致，自定义布局场景下支持选择裁剪和缩放两种模式。
  - 如果房间内的人数不足，则剩余位置显示背景色。
  - 如果用户只发送音频，仍然会占用画面位置。
  - 支持背景图，如果房间内人数不足，则显示背景图。

录制机制

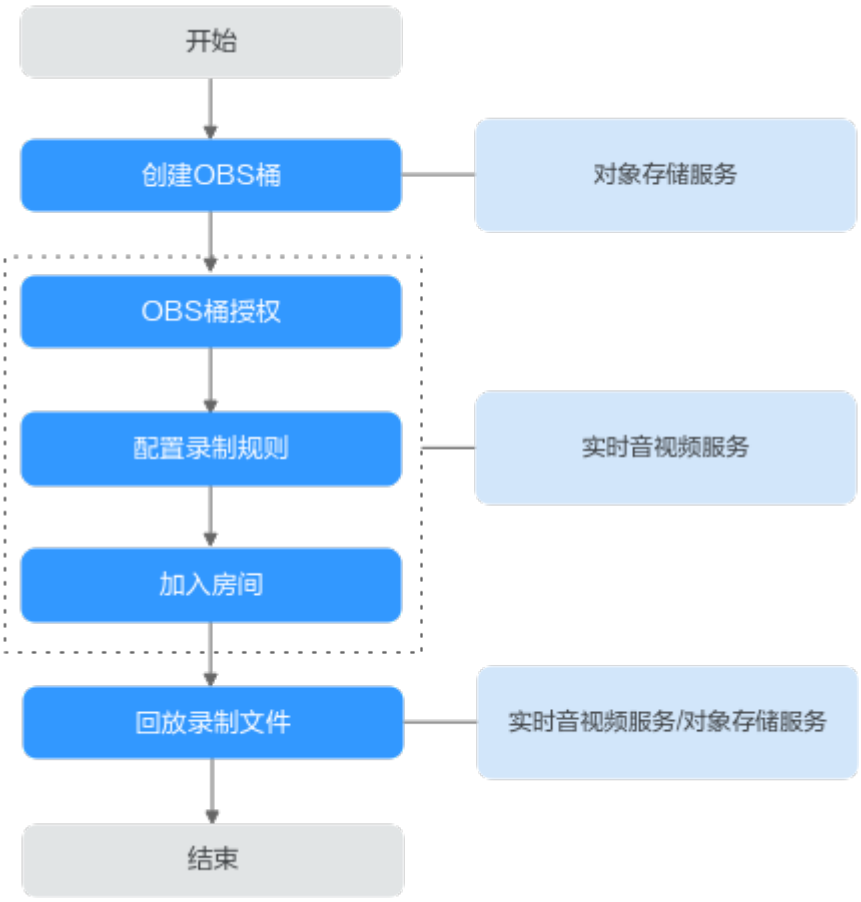
SparkRTC提供的**合流录制**，具体的实现机制如**图1-4**所示。

说明：启动合流录制任务时，需要设定**MixParam**合流参数，指定**layout_template**视频布局模板。

图 1-4 合流录制



实现流程



1. **创建OBS桶**：创建用于存储SparkRTC录制文件的OBS桶，若已有OBS桶，请直接执行2。

   说明

   由于单AZ桶的可靠性低于多AZ桶，为避免因OBS服务异常导致录制失败，建议您创建多AZ桶用于录制文件的存储。

2. **OBS桶授权**：在SparkRTC服务中对存储录制文件的OBS桶进行授权，允许SparkRTC服务将录制文件存储在对应的OBS桶中。
3. **配置录制规则**：为实时音视频互动配置录制规则，并开启自动录制功能，加入SparkRTC房间后，与应用中的**录制规则ID**相同的录制模板会自动生效，录制内容按录制设置存储至OBS中。还可以通过设置回调地址获取录制任务状态通知。
4. **加入房间**：录制规则配置完成后，您可以通过SparkRTC APP加入某个SparkRTC房间进行音视频互动，SparkRTC会根据配置的录制规则对正在直播的音视频进行录制。

   说明

   若配置录制规则时，未开启自动录制功能，则加入SparkRTC房间后，您需要调用**实时音视频API**开启、查询和控制云端录制任务。SparkRTC服务才会根据接口中的**录制规则ID**对实时音视频互动画面进行录制。

5. **回放录制文件**：录制完成后，在已配置的回调地址中会收到录制任务的回调消息，您可以获取到录制文件的基本信息，也可以在OBS中管理录制文件，如下载、分享、删除等。

说明

录制文件的分辨率与推流分辨率相关，按推流原分辨率进行录制。

实现步骤

步骤1 请参见**OBS帮助中心**创建桶。若已有OBS桶，请直接执行**步骤2**。

说明

创建的OBS桶所在区域必须为**华北-北京四**。

步骤2 OBS桶授权。

1. 登录实时音视频控制台。
2. 在左侧导航树中选择“云资源授权”，进入桶授权页面。
3. 在对应的OBS桶行单击“授权”，完成桶授权。

步骤3 配置录制规则。

1. 登录实时音视频控制台。
2. 在左侧导航树中选择“应用管理”，进入应用管理页面。
3. 在需要创建录制规则的应用行单击“录制配置”，进入录制配置页面。
4. 在“录制规则”页签，单击“添加”，进入添加录制规则页面。

说明

一个应用ID仅支持创建一个录制规则。

5. 请您按照实际需求配置录制参数，参数说明如**表1-3**所示。

表 1-3 录制参数说明

参数名		描述
存储-桶		存储录制文件的OBS桶。 目前录制文件仅支持存储到**华北-北京四**的OBS桶中。
区域		OBS桶所在的区域。
存储-路径		存储录制文件的OBS桶路径。
录制格式		录制文件的格式，支持HLS和MP4文件格式。
HLS规则	m3u8命名规则	录制m3u8文件的存储路径和文件的前缀。 默认命名格式： {app_id}/{record_format}/{stream}_{file_start_time}/{stream}_{file_start_time} 上述特殊变量的含义如下： - app_id：应用ID。 - record_format：录制格式。 - stream：流名。 - file_start_time：文件生成时间。

参数名		描述
	录制周期	录制时长支持0-720分钟，最小录制周期为1分钟，最大录制周期为12小时，超过12小时，系统将按照命名规则生成新文件。如果录制周期为0，则整个流录制为一个文件。
	最大断流合并时长	支持如下三种配置： – 断流后生成新文件：是指录制的直播流中断后，会立即生成新的录制文件。 – 断流后不生成新文件：是指录制的直播流中断后，会和之前录制的文件合并为一个文件。最大断流合并时长为30天。 – 其他数值：是指录制的直播流中断时间在设置范围内，则和之前录制的文件合并为一个文件，否则，生成新的录制文件。
MP4规则	mp4命名规则	录制mp4文件的存储路径和文件的前缀。 默认命名格式： {app_id}/{record_format}/{stream}_{file_start_time}/{stream}_{file_start_time} 上述特殊变量的含义如下： – app_id：应用ID。 – record_format：录制格式。 – stream：流名。 – file_start_time：文件生成时间。
	录制周期	录制时长支持1-180分钟，最小录制周期为1分钟，最大录制周期为3小时，超过3小时，系统将按照命名规则生成新文件。
	最大断流合并时长	支持如下两种配置： – 断流后生成新文件：是指录制的直播流中断后，会立即生成新的录制文件。 – 其他数值：是指录制的直播流中断时间在设置范围内，则和之前录制的文件合并为一个文件，否则，生成新的录制文件。

6. 单击“确定”，在录制规则列表中会增加一条新的录制规则。

图 1-5 录制规则

7. 您可以在录制规则列表中，根据实际需求选择是否开启自动录制功能。自动录制功能开启后，若该应用下有新创建的房间，则会按照已配置的录制规则自动对该房间中的实时音视频互动过程进行录制。

> **说明**
>
> 自动录制功能开启后，仅对同一应用下新创建的房间生效，自动录制功能开启前已创建的房间不生效。

步骤4 加入房间。

录制规则配置完成后，您可以通过SparkRTC APP加入某个SparkRTC房间进行音视频互动，SparkRTC服务会根据配置的录制规则对正在直播的音视频进行录制。

> **说明**
>
> 若配置录制规则时，未开启自动录制功能，则加入SparkRTC房间后，您需要调用**实时音视频API**开启云端录制任务，SparkRTC才会根据API中的**录制规则ID**对实时音视频互动画面进行录制。

步骤5 回放录制文件。

录制完成后，您可以在OBS控制台中或通过回调消息查看录制文件。

- 通过OBS控制台查看录制文件
  a. 在OBS管理控制台左侧导航栏选择“对象存储”。
  b. 在桶列表中单击存储SparkRTC录制文件的桶，进入“概览”页面。
  c. 在左侧导航栏，单击“对象”，查看录制文件信息。
  d. 您还可以对录制文件进行下载、分享等操作，具体请参见**OBS帮助中心**。

----结束

2 实现音视频通话

2.1 实现音视频通话（Web）

2.1.1 环境准备

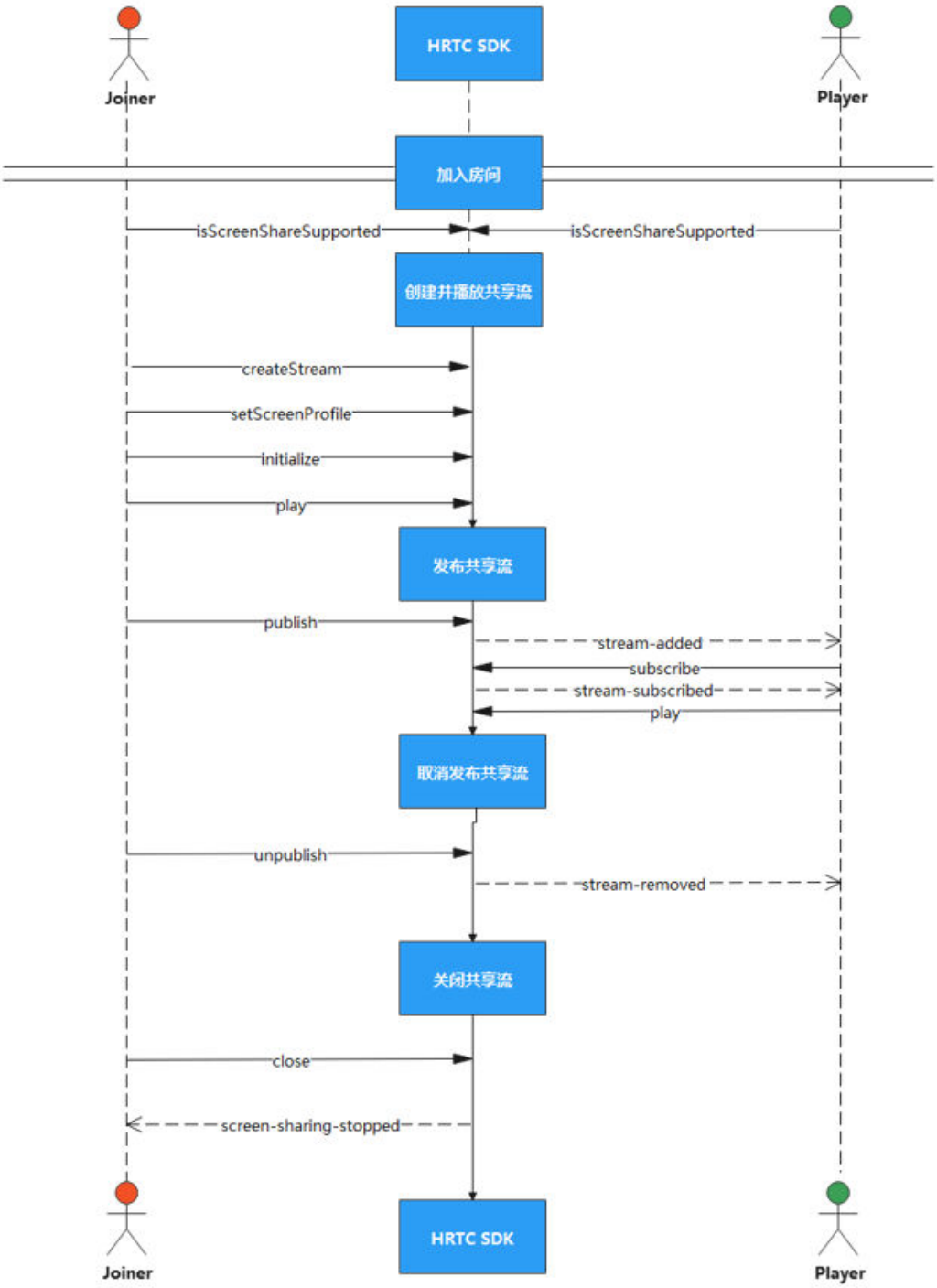
详情请参考**开发前准备**。

2.1.2 屏幕分享

功能描述

屏幕共享用于在音视频会议中，把一个与会者的屏幕内容，以视频的方式分享给其他与会者。屏幕共享可以共享整个桌面，也可以共享某一个程序窗口。

接口调用流程



实现屏幕共享

1. **加入房间**

   参考**接口调用流程**中加入房间的时序图步骤加入房间。

2. **创建并播放共享流**

   入会后调用isScreenShareSupported检测浏览器是否支持共享流。确认支持后，通过调用createStream创建共享流，通过调用setScreenProfile设置共享流的分辨

率，通过调用initialize初始化共享流，通过调用play播放共享流，通过调用bindScreenAudio2RelatedStream设置是否绑定屏幕共享背景音乐至关联流对象。

示例代码如下：

```
// screenAudio 设置是否共享主流音频
let co = {screen: true , screenAudio: false}
// 创建共享流
let localAuxStream = HRTC.createStream(co)
// 设置共享流分辨率
localAuxStream.setScreenProfile('1080p')
// 初始化共享流
localAuxStream.initialize()
.then(() => {   // 播放共享流 c${this.clientIndex}-aux 播放共享流的DOM元素
localAuxStream.play(`c${this.clientIndex}-aux`)
localAuxStream.bindScreenAudio2RelatedStream(this.localStream, screenAudio)})
.catch((error) => {
 console.error(error)
})
```

3. **发布共享流**

   本地播放共享流后，通过调用publish发布本地共享流。

   示例代码如下：

   ```
   this.client.publish(localAuxStream)
   .then(() => {console.info(`发布共享流成功`)})
   .catch((error) => {
    console.error(`发布共享流失败`, error)
   })
   ```

4. **接收远端用户的共享流**

   收到远端用户开启共享流通知stream-added后，通过调用subscribe订阅远端用户的共享流，当共享流订阅成功会收到stream-subscribed回调通知，然后通过调用play在指定的窗口里播放接收到的远端用户的共享流。

   示例代码如下：

   ```
   this.client.on('stream-added', (event: any) => {
   // 流类型为共享流的话
   if (event.stream.getType() === 'auxiliary') {
   // remoteAuxStream 远端共享流
   let remoteAuxStream = event.stream
   this.client.subscribe(remoteAuxStream )
   .then(() => {console.info(`订阅共享流成功`)})
   .catch( (error) =>(console.info(`订阅共享流失败`)))
    }})
   this.client.on('stream-subscribed', (event: any) => {
   // 流类型为共享流的话
   if (event.stream.getType() === 'auxiliary') {
   // c${ clientIndex}-remoteAux 播放共享流的DOM元素
    event.stream.play(`c${ clientIndex}-remoteAux`)
    }
   })
   ```

5. **取消发布共享流**

   共享流发布后，可以通过调用unpublish取消共享流。

   示例代码如下：

   ```
   this.client.unpublish(localAuxStream)
   .then(() => {console.info(`取消发布共享流成功`)})
   .catch((error) => {
   console.error(`取消发布共享流失败`, error)
   })
   ```

6. **停止播放共享流**

   共享流发布后，可以通过调用close停止播放共享流，共享流停止成功后本地会收到screen-sharing-stopped回调信息。

示例代码如下：

```
localAuxStream.close()
localAuxStream.on('screen-sharing-stopped', () => {
 console.log(`屏幕共享停止`)
 localAuxStream = null
})
```

7. **停止接收远端用户的共享流**

   远端用户的共享流取消发布后，本地自动取消订阅，同时会收到stream-removed回调消息。

   示例代码如下：

```
this.client.on('stream-removed', (event: any) => {
 if (event.stream.getType() === 'auxiliary') {
 remoteAuxStream = null
 }
})
```

API 参考

createStream

isScreenShareSupported

setScreenProfile

initialize

play

bindScreenAudio2RelatedStream

publish

subscribe

unpublish

stream-added

stream-subscribed

stream-removed

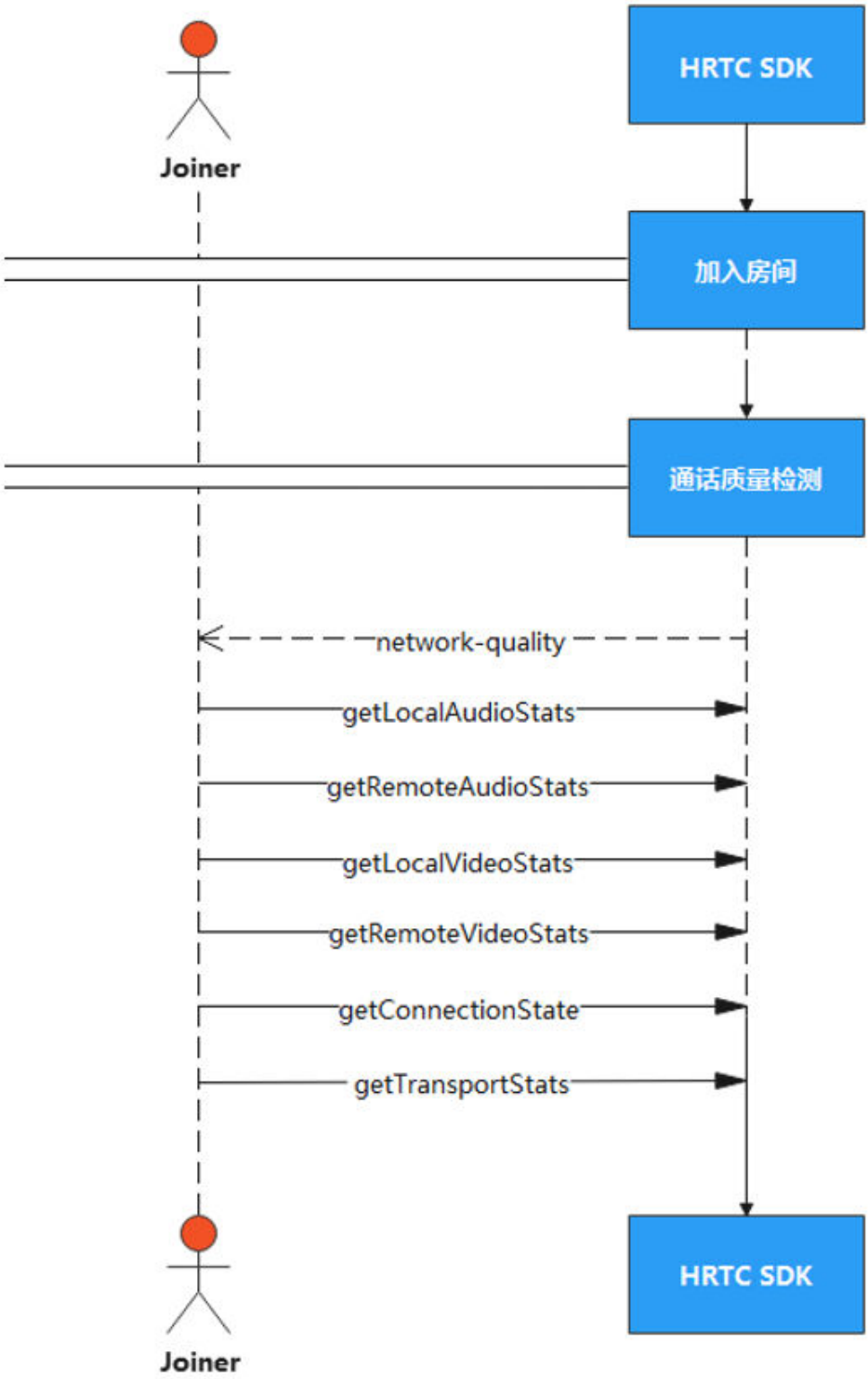
screen-sharing-stopped

2.1.3 通话质量监测

功能描述

加入房间后，SDK上报通话质量相关的回调，上报当前通话的网络质量、本地和远端的音视频统计信息。

接口调用流程



通话质量上报

network-quality网络上下行质量报告事件，用户加入房间后，在网络质量变化的时候会触发一次该事件，报告用户的本地网络上下行质量情况。

示例代码如下：

```
this.client.on('network-quality', (networkQualityInfo) => {
console.info(`network-quality:
uplinkNetworkQuality=${networkQualityInfo.uplinkNetworkQuality},
downlinkNetworkQuality = ${networkQualityInfo.downlinkNetworkQuality}`)
})
```

获取本地音频流统计信息

getLocalAudioStats获取本地设备发送音频流的统计信息。您可以查看当前通话中发送音频的字节数和包数等。

示例代码如下：

```
this.client.getLocalAudioStats().then((stats) => {
 console.info(`getLocalAudioStats: ${stats}`)
})
```

获取远端音频流统计信息

getRemoteAudioStats获取当前通话中远端用户音频流的统计信息。您可以查看远端用户发送的音频流字节数、收包数、丢包率等信息。

示例代码如下：

```
this.client.getRemoteAudioStats().then((stats) => {
 console.info(`getRemoteAudioStats: ${stats}`)
})
```

获取本地视频流统计信息

getLocalVideoStats获取当前本地视频流统计信息，您可以查看本地已发送的字节数、包数、编码帧数，帧数、视频宽度、视频高度等信息。

示例代码如下：

```
this.client.getLocalVideoStats().then((stats) => {
 console.info(`getLocalVideoStats: ${stats}`)
})
```

获取远端视频流统计信息

getRemoteVideoStats获取当前远端视频流统计信息，您可以查看远端用户发送的视频流字节数、包数、编码帧数，帧数、视频宽度、视频高度等信息。

示例代码如下：

```
this.client.getRemoteVideoStats().then((stats) => {
 console.info(`getRemoteVideoStats: ${stats}`)
})
```

获取客户端连接状态

getConnectionState获取客户端的连接状态，分别为：

- CONNECTING：连接建立中。
- CONNECTED：连接已建立。
- RECONNECTING：重新连接中。
- DISCONNECTED：连接已断开。

示例代码如下：

```
console.info(`getConnectingState: ${this.client.getConnectionState()}`)
```

获取当前网络传输状况统计数据

getTransportStats获取当前网络传输状况统计数据，包括已发送字节数、已接收字节数、当前出流码率、当前入流码率等信息，该方法需要publish后调用。

示例代码如下：

```
this.client.getTransportStats().then(
 (rtt) => {
 console.info('###getTransportStats: bytesSent ' + rtt.bytesSent)
 console.info('###getTransportStats: bytesReceived ' + rtt.bytesReceived)
 console.info('###getTransportStats: sendBitrate ' + rtt.sendBitrate)
 console.info('###getTransportStats: recvBitrate ' + rtt.recvBitrate)
 console.info(`getTransportStats: ${rtt.rtt}`)
 },
 (error) => {
 console.info(`getTransportStats: ${error}`)
})
```

API 参考

network-quality

getLocalAudioStats

getRemoteAudioStats

getLocalVideoStats

getRemoteVideoStats

getConnectionState

getTransportStats

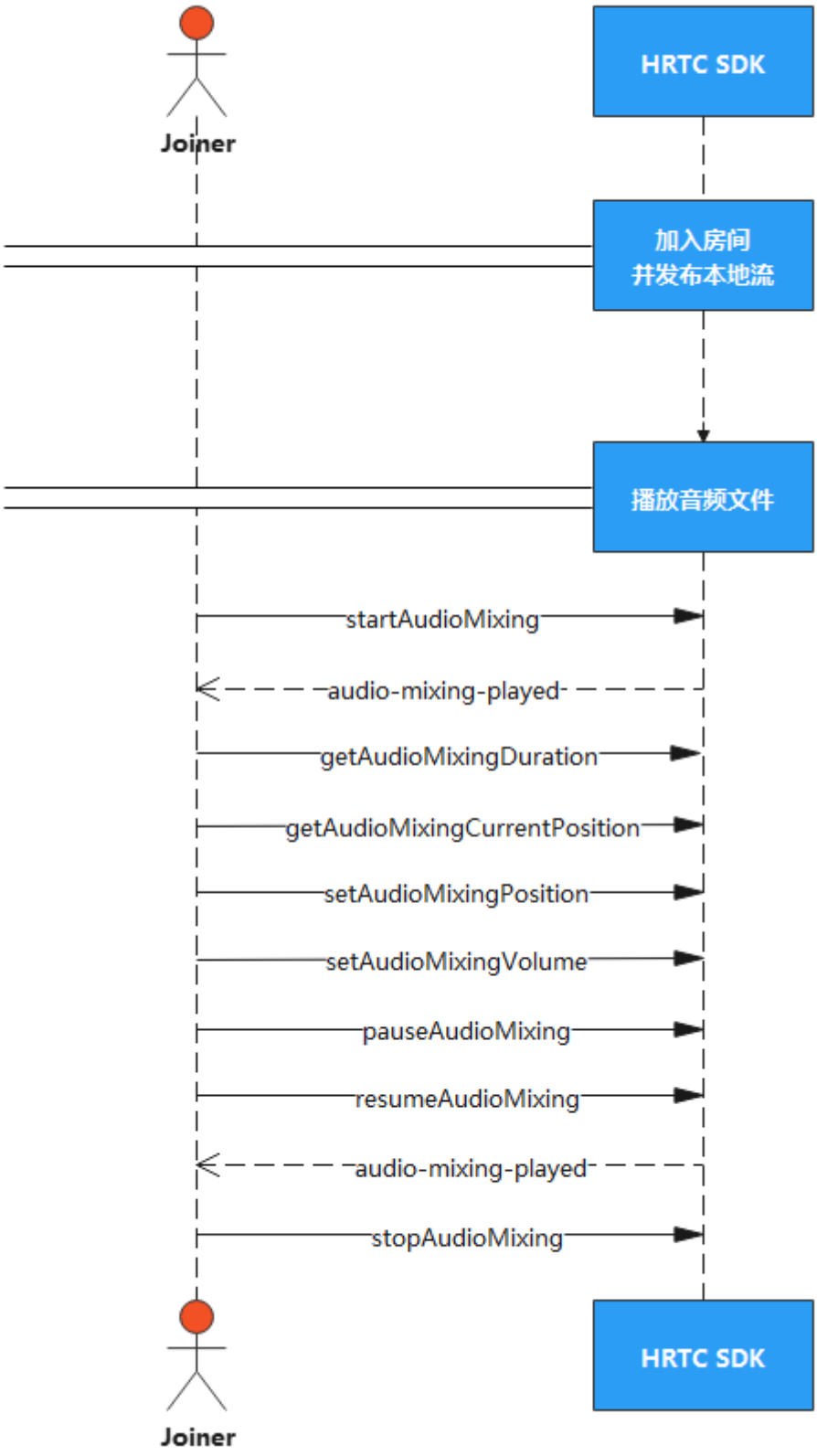
2.1.4 播放音频文件（混音）

功能描述

混音是将音频文件和麦克风音频混合，一般用于播放时长较长的背景音乐或者伴奏。同一时间只能播放一个音频文件，播放给房间内的其他用户听。

支持播放本地或在线音乐文件，文件格式支持播放wav、pcm和单声道mp3音频格式。

接口调用流程



实现过程

1. **加入房间并发布本端主流**

   参考**接口调用流程**中加入房间的时序图步骤加入房间，并发布本地主流。

2. **播放音频文件**

   调用startAudioMixing可以播放一个音频文件，参数设置参见如下示例。同一时刻只能播放一个音频文件。

   ```
   // localStream 本地主流
   localStream.startAudioMixing({
   // filePath 表示在线音频文件的下载路径
   "filePath":"https://***.***.***.***:50007/music.mp3",
   // startTime 表示音频文件开始播放的时间点，默认值为0。
   "startTime":0,
   // replace 表示是否要用音频文件替换本地音频流
   "replace":false,
   // loop 表示是否需要无限循环播放
   "loop":false,
   // repeatCount 表示音频文件循环播放次数
   "repeatCount":0
   })
   ```

3. **设置音频文件音量值**

   音频文件播放过程中，可以调用setAudioMixingVolume设置音频文件播放音量。

   ```
   // volume：音量值
    let volume = 50
    localStream.setAudioMixingVolume (volume)
   ```

4. **获取音频文件总时长**

   音频文件播放成功后，可以调用getAudioMixingDuration获取音频文件总时长，用于刷新界面进度条的总时长。

   ```
   localStream.getAudioMixingDuration()
   ```

5. **获取音频文件播放进度**

   音频文件播放成功后，可以调用getAudioMixingCurrentPosition获取音频文件当前播放时间点。

   ```
   localStream.getAudioMixingCurrentPosition()
   ```

6. **设置音频文件播放位置**

   播放过程中可以调用setAudioMixingPosition设置音频文件播放位置，用于通过拖动进度条改变音频文件的播放位置。

   ```
   localStream.getAudioMixingDuration()
   ```

7. **暂停播放音频文件**

   音频文件播放成功后，可以调用pauseAudioMixing接口暂停播放的音频文件。

   ```
   localStream.pauseAudioMixing()
   ```

8. **恢复播放暂停的音频文件**

   音频文件暂停播放后，可以调用resumeAudioMixing接口恢复播放。

   ```
   localStream.resumeAudioMixing()
   ```

9. **停止播放音频文件**

   音频文件播放成功后，可以调用stopAudioMixing接口停止播放音频文件。

   ```
   localStream.stopAudioMixing()
   ```

10. **音频文件播放开始回调事件**

    音频文件播放开始时，会触发audio-mixing-played回调信息通知App应用。

    ```
    localStream.on('audio-mixing-played', () => {
     console.info('audioMixing: audio-mixing-played')
    })
    ```

11. **音频文件播放结束回调事件**

    音频文件播放结束时，会触发audio-mixing-finished回调信息通知App应用。

```
localStream.on('audio-mixing-finished', () => {
 console.info('audioMixing: audio-mixing-finished')
})
```

API 参考

startAudioMixing

getAudioMixingDuration

getAudioMixingCurrentPosition

setAudioMixingPosition

setAudioMixingVolume

pauseAudioMixing

resumeAudioMixing

stopAudioMixing

2.1.5 切换音频模式

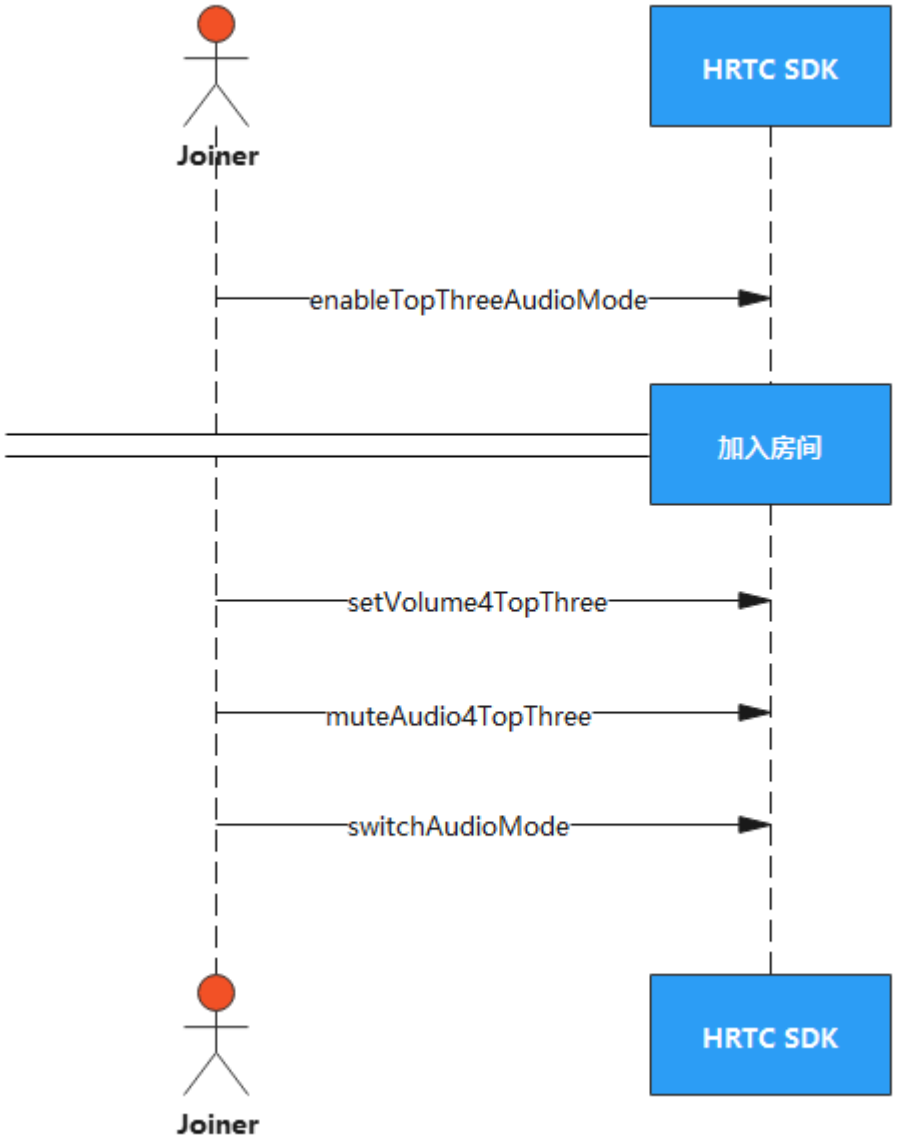
功能描述

用户在入会前可通过调用enableTopThreeAudioMode切换为音频最大三方模式。

会中通过调用switchAudioMode(2)将音频切换为订阅模式。订阅模式下，本地用户必须通过主动订阅远端用户音频流，才可接收该用户音频。

会中通过调用switchAudioMode(3)将音频切换为最大三方模式。最大三方模式下，本地用户不需要单独主动订阅某个远端用户音频流，即可接收当前房间内声音最大的三个用户的音频。

接口调用流程



实现过程

1. **加入房间前设置音频最大三方模式**

   入会前调用enableTopThreeAudioMode设置是否开启音频最大三方模式。true表示开启音频最大三方，false表示不开启。

   示例代码如下：

   ```
   this.client.enableTopThreeAudioMode(true)
   ```

2. **加入房间**

   参考**接口调用流程**中加入房间的时序图步骤。

3. **设置音频最大三方音量值**

调用setVolume4TopThree接口可以设置音频最大三方模式的音量值，取值范围为[0,100]。

示例代码如下：

```
// volume：音量值
let volume = 50
this.client.setVolume4TopThree(volume)
```

4. **开启/禁用音频最大三方模式的音轨**

   调用muteAudio4TopThree接口可以开启/禁用音频最大三方模式的音轨，true表示禁用音频最大三方模式的音轨，false表示开启音频最大三方模式的音轨。

   示例代码如下：

   ```
   this.client.muteAudio4TopThree(true)
   ```

5. **切换音频订阅模式/音频最大三方模式**

   调用switchAudioMode可以切换音频模式。

   当调用switchAudioMode(2)时，表示当前为音频订阅模式；当调用switchAudioMode(3)时，表示当前为音频最大三方模式。

   示例代码如下：

   ```
   this.client.switchAudioMode(2)
   this.client.switchAudioMode(3)
   ```

API 参考

enableTopThreeAudioMode

setVolume4TopThree

muteAudio4TopThree

3 修订记录

表 3-1 表 1 修订记录

发布日期	修订说明
2023-11-30	实时音视频所有客户端类型均增加“环境准备”章节。
2022-09-30	更新“单流录制”和“合流录制”章节的操作内容。
2021-10-30	第一次正式发布。