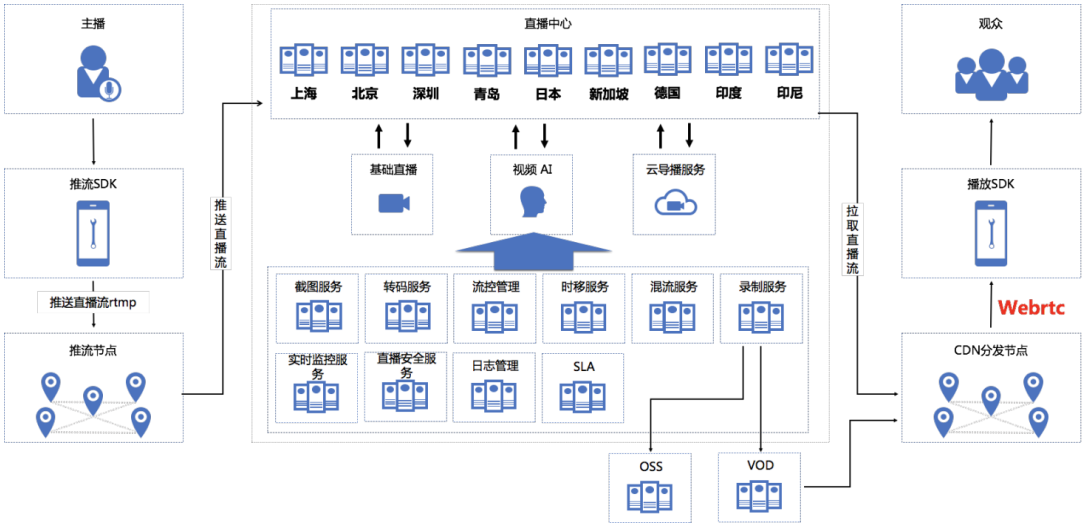
# WebRTC 标准接入 GRTN 信令协议规范

## 背景描述

阿里云视频直播服务（ApsaraVideo Live）支持以标准 WebRTC 接入方式播放直播流。主播以 RTMP 协议将直播流推入阿里云视频直播系统后，观众使用标准 WebRTC 接入方式将直播流从阿里云视频直播系统拉取播放。借助直播 CDN 的全球覆盖能力，帮助客户实现大规模低延迟直播。

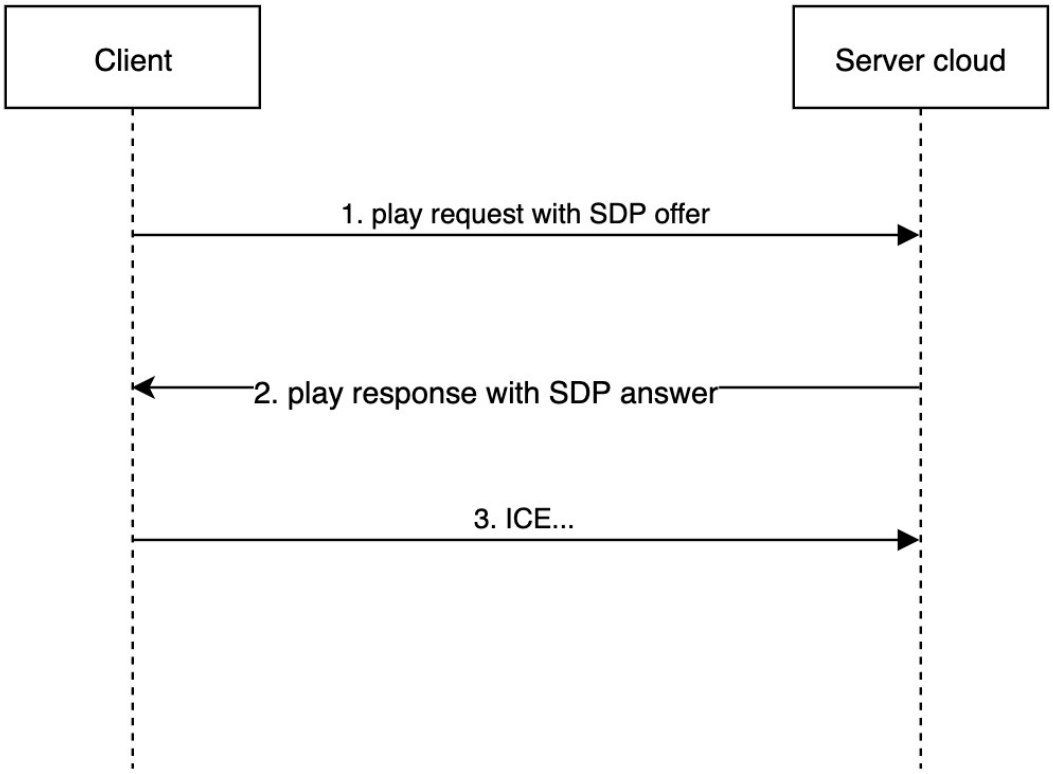
工作原理




## 前提条件

您已经完成阿里云视频直播服务开通，具体操作请参见[阿里云视频直播服务开通流程]。

## 接入流程




## 客户端 Offer 请求

1. 客户端本地创建拉流 RTCPeerConnection，设置接收音视频属性并创建 offer SDP。

```
// 开启音视频, recvonly.

{ offerToReceiveVideo: true, offerToReceiveAudio: true }
```

2. 客户端向直播服务发送拉流请求，通过 HTTPS POST 方式将 json 格式的请求信息发送至直播服务。

   协议格式请参见信令协议：

- version 字段为协议版本，当前版本固定为 2。
- sdk_version 字段为 sdk 版本，用户可以自定义该字段。
- 目前仅支持 H264 编解码，生成 offer SDP 后检查视频是否支持 H264 参数。

3. 生成协议内容后，通过 URL 如:https://domain/app/name?auth=xxx POST 给直播服务，表明要拉取的流为 https://domain/app/name，与 RTMP 拉流 URL: rtmp://domain/app/name 类似。

```
POST /app/name?auth=xxx HTTP/1.1
Host: domain
Connection: keep-alive
Content-Length: 2205
Content-Type: application/json
```

## 服务端 Answer 响应

直播服务生成 SDP answer，封装在响应中给客户端。

协议格式请参见信令协议

## 客户端 ICE 建联

1. 客户端收到 SDP Answer 响应后，设置到 RTCPeerConnection 中。

```
peerConnection.setRemoteDescription(new
RTCSessionDescription(answer.jsep));
```

2. RTCPeerConnection 启动 ICE 建联流程以及后续的 DTLS 流程，建立媒体链接成功获取到直播服务输出的媒体流，实现 WebRTC 标准接入拉流播放。

## H5 Demo 示例

```
// Create peer connection and local offer sdp.
peerConnection = new RTCPeerConnection();
peerConnection.onicecandidate = iceCandidateCallback;
peerConnection.ontrack = remoteStreamCallback;
peerConnection.createOffer({ offerToReceiveVideo: true,
offerToReceiveAudio: true })
      .then(signaling_pull).catch(errorHandler);


// CDN live post pull stream request.
function signaling_pull(offer_sdp) {
  console.log('local offer sdp', offer_sdp);

  peerConnection.setLocalDescription(offer_sdp).then(function() {
    // Get pull stream url.
    var stream_url = $("#stream_url").val();
    console.log("stream url:" , stream_url);

    // Add sdk and protocol versions.
    var protocol_version = 2;
    var sdk_version = "0.0.1";

    $.ajax({url: stream_url, data: JSON.stringify({
          mode: "live",
          version: protocol_version,
          sdk_version: sdk_version,
          jsep:description,
```

```
        }),
        type: "post",
        success:function(result){
            var signal = JSON.parse(result);
            peerConnection.setRemoteDescription(new
RTCSessionDescription(signal.jsep)).then(function() {
                console.log("get remote answer sdp: ",
signal.jsep.sdp);
            }).catch(errorHandler);
        }});
    }).catch(errorHandler);
}
```

# 信令协议定义

## 协议通道

https or http，短连接

## 协议格式

json

## 协议描述

### 请求参数

| 字段 | 类型 | 必须 | 描述 | 取值范围 |
| --- | --- | --- | --- | --- |
| mode | string | 是 | 模式：直播 | live |
| version | int | 是 | 协议版本号 | 2 |
| push_stream | string? | 否 | 推流 URL | 文本 |

| | | | | |
|---|---|---|---|---|
| pull_streams | []object? | 否 | 拉流对象，支持多个拉流 | 参见[pull_stream参数] |
| sdk_version | string? | 否 | sdk 版本号 | 文本 |
| jsep.type | string | 是 | sdp 类型：offer | "offer" |
| jsep.sdp | string | 是 | sdp 描述 | |

**pull_stream 参数**

| 字段 | 类型 | 必须 | 描述 |
|---|---|---|---|
| url | string | 是 | 拉流 URL |
| amsid | []string | 是 | 拉流音频 msid,直播场景填写一个成员"rts audio" |
| vmsid | []string | 是 | 拉流视频 msid,直播场景填写一个成员"rts video" |

MSID 描述见文末描述

**应答参数**

| 字段 | 类型 | 必须 | 描述 |
|---|---|---|---|
| code | int | 是 | 正确返回 200，错误参考错误处理 |
| trace_id | string | 是 | 连接标识 ID，由直播 CDN 返回 |
| jsep.type | string | 是 | sdp 类型：answer |

| | | | |
|---|---|---|---|
| jsep.sdp | string | 是 | 直播 CDN 回源拉流生成 SDP |

## 请求示例

```
Request:
{
    "version":2,
    "sdk_version":"0.0.1",
    "mode":"live",
    "pull_streams":[
        {
            "url":"artc://your.domain.com/live/testname",
            "amsid":[
                "rts audio"
            ],
            "vmsid":[
                "rts video"
            ]
        }
    ],
    "jsep":{
        "type":"offer",
        "sdp":"v=0\n\ro=- 6839248142876176651 2 IN IP4
127.0.0.1\n\rs=-\n\r 以下省略"
    }
}

Response:
{

"trace_id":"2_1591173296_101.227.0.169_702080732320_dec327eb6eed0e0b0
7b349c8a5653eca",
```

```
    "code":200,
    "jsep":{
        "type":"answer",
        "sdp":"v=0\r\no=- 1591173291 2 IN IP4 127.0.0.1\n\r 以下省略"
    }
}
```

## 错误处理

信令请求合法的情况下响应 200 ，具体的处理结果需要解析响应 body 的 json 内的 code 属性。code 为 http 响应码模式，具体定义如下：

```
Response:
{
   "code": 200, // 200-成功 非 200-见后续定义
   "message": "success"  // 非 200 错误码描述
}
```

**应答参数**

| 参数 | 类型 | 描述 |
| --- | --- | --- |
| code | int | 响应码，格式和定义参见下面 |
| message | string | 错误描述 |

**错误码描述**

| 错误码 | 描述 |
| --- | --- |
| 200 | success<br>成功 |
| 403 | 鉴权失败 |
| 404 | 流不存在 |
| 611 | 需要客户端强制降级 |
| 302 | 需要客户端向新的服务地址发起信令请求 |
| | |

# 附录：

## 关于 MSID 的关键概念描述：

https://tools.ietf.org/html/draft-ietf-mmusic-msid-17#page-5

**划重点：**

This document defines a new SDP [RFC4566] media-level "msid" attribute. This new attribute allows endpoints to associate RTP streams that are described in different media descriptions with the same MediaStreams as defined in [W3C.WD-webrtc-20160531], and to carry an identifier for each MediaStreamTrack in its "appdata" field.

The value of the "msid" attribute consists of an identifier and an optional "appdata" field.

The name of the attribute is "msid".

The value of the attribute is specified by the following ABNF [RFC5234] grammar:

```
msid-value = msid-id [ SP msid-appdata ]
msid-id = 1*64token-char ; see RFC 4566
msid-appdata = 1*64token-char   ; see RFC 4566
```

An example msid value for a group with the identifier "examplefoo" and application data "examplebar" might look like this:

```
msid:examplefoo examplebar
```

The identifier is a string of ASCII characters that are legal in a "token", consisting of between 1 and 64 characters.

Application data (msid-appdata) is carried on the same line as the identifier, separated from the identifier by a space.

The identifier (msid-id) uniquely identifies a group within the scope
of an SDP description.