

07 | 你竟然不知道SDP？它可是WebRTC的驱动核心！

2019-07-30 李超

从0打造音视频直播系统

[进入课程 >](#)



讲述：李超

时长 22:49 大小 18.30M



在前面[《01 | 原来通过浏览器访问摄像头这么容易》](#)[《04 | 可以把采集到的音视频数据录制下来吗？》](#)等文章中，我向你讲解了 WebRTC 如何采集音视频数据，以及如何将它们录制成文件等相关内容。但那些知识不过是个“开胃菜”，WebRTC 真正核心的知识将从本文开始陆续向你展开。不过从本文开始，知识的难度会越来越高，你一定要做好心理准备。

说到 WebRTC 运转的核心，不同的人可能有不同的理解：有的人认为 WebRTC 的核心是音视频引擎，有的人认为是网络传输，而我则认为 WebRTC 之所以能很好地运转起来，完全是由 SDP 驱动的，因此**SDP 才是 WebRTC 的核心**。

掌握了这个核心，你就知道 WebRTC 都支持哪些编解码器、每次通话时都有哪些媒体（通话时有几路音频 / 视频）以及底层网络使用的是什么协议，也就是说你就相当于拿到了打开 WebRTC 大门的一把钥匙。

由此可见，SDP 在 WebRTC 中是何等重要。下面就让我们正式进入正题吧！

SDP 是什么

在正式讲解 SDP 之前，你首先要弄清楚 SDP 是什么？SDP (Session Description Protocal) 说直白点就是用文本描述的各端 (PC 端、Mac 端、Android 端、iOS 端等) 的**能力**。这里的**能力**指的是各端所支持的音频编解码器是什么，这些编解码器设定的参数是什么，使用的传输协议是什么，以及包括的音视频媒体是什么等等。

下面让我们来看一个真实的 SDP 片段吧！

 复制代码

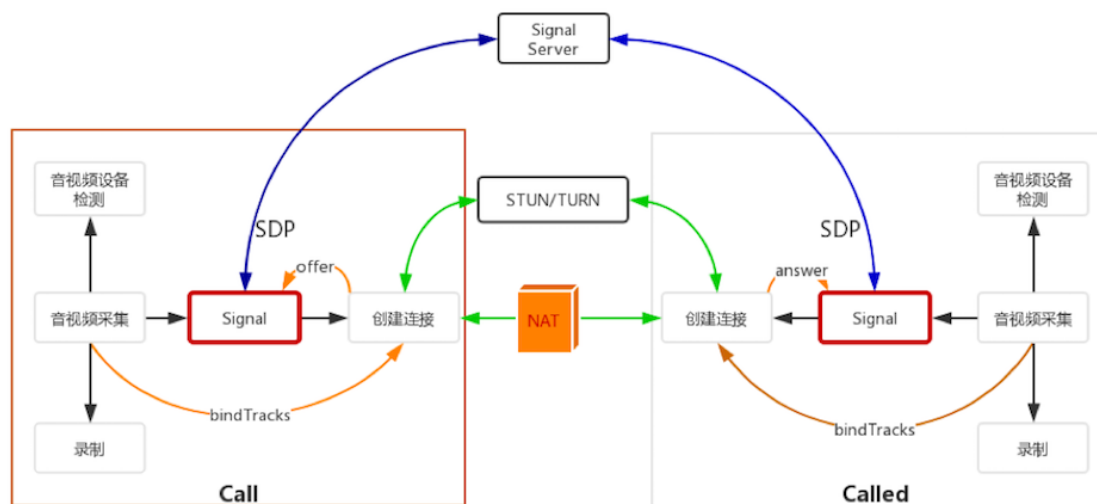
```
1 v=0
2 o=- 3409821183230872764 2 IN IP4 127.0.0.1
3 ...
4 m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 110 112 113 126
5 ...
6 a=rtpmap:111 opus/48000/2
7 a=rtpmap:103 ISAC/16000
8 a=rtpmap:104 ISAC/32000
9 ...
```

如上面的 SDP 片段所示，该 SDP 中描述了一路音频流，即**m=audio**，该音频支持的 Payload (即数据负载) 类型包括 111、103、104 等等。

在该 SDP 片段中又进一步对 111、103、104 等 Payload 类型做了更详细的描述，如 **a=rtpmap:111 opus/48000/2** 表示 Payload 类型为 111 的数据是 OPUS 编码的音频数据，并且它的采样率是 48000，使用双声道。以此类推，你也就可以知道 **a=rtpmap:104 ISAC/32000** 的含义是音频数据使用 ISAC 编码，采样频率是 32000，使用单声道。

交换 SDP 信息

下面是 1 对 1 WebRTC 处理过程图：



WebRTC 处理过程图

如上图所示，两个客户端 / 浏览器进行 1 对 1 通话时，首先要进行信令交互，而**交互的一个重要信息就是 SDP 的交换**。

交换 SDP 的目的是为了让对方知道彼此具有哪些**能力**，然后根据双方各自的能力进行协商，协商出大家认可的音视频编解码器、编解码器相关的参数（如音频通道数，采样率等）、传输协议等信息。

举个例子，A 与 B 进行通讯，它们先各自在 SDP 中记录自己支持的音频参数、视频参数、传输协议等信息，然后再将自己的 SDP 信息通过信令服务器发送给对方。当一方收到对端传来的 SDP 信息后，它会接收到的 SDP 与自己的 SDP 进行比较，并取出它们之间的交集，这个交集就是它们协商的结果，也就是它们最终使用的音视频参数及传输协议了。

标准 SDP 规范

了解了 SDP 是什么，接下来我们看一下 SDP 规范。其实单论 SDP 规范它并不复杂，但 WebRTC 使用时又对其做了不少修改，所以当你初见完整的 WebRTC 的 SDP 时，可能会一脸茫然。

不过没关系，万事总有头。在本文中，我先带你了解 SDP 的标准规范，然后再一步步深入，相信通过本文的学习，最终你将能基本读懂 WebRTC 所产生的 SDP 信息，从而为后面学习 WebRTC 打下坚实的基础。

标准 SDP 规范主要包括**SDP 描述格式**和**SDP 结构**，而 SDP 结构由**会话描述**和**媒体信息描述**两个部分组成。

其中，媒体信息描述是整个 SDP 规范中最重要的知识，它又包括了：

媒体类型

媒体格式


传输协议

传输的 IP 和端口

下面我们就以上这些知识逐一进行讲解。

1. SDP 的格式

SDP 是由多个 `<type>=<value>` 这样的表达式组成的。其中，`<type>`是一个字符，`<value>`是一个字符串。需要特别注意的是，“=” **两边是不能有空格的**。如下所示：

 复制代码

```
1 v=0
2 o=- 7017624586836067756 2 IN IP4 127.0.0.1
3 s=-
4 t=0 0
5 ...
```

SDP 由一个会话级描述 (session level description) 和多个媒体级描述 (media level description) 组成。

会话级 (session level) 的作用域是整个会话，其位置是从 **v= 行开始到第一个媒体描述为止**。

媒体级 (media level) 是对单个的媒体流进行描述，其位置是从 **m= 行开始到下一个媒体描述 (即下一个 m=) 为止**。

另外，除非媒体部分重新对会话级的值做定义，否则会话级的值就是各个媒体的缺省默认值。让我们看个例子吧。

```
1 v=0
2 o=- 7017624586836067756 2 IN IP4 127.0.0.1
3 s=-
4 t=0 0
5
6 // 下面 m= 开头的两行，是两个媒体流：一个音频，一个视频。
7 m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
8 ...
9 m=video 9 UDP/TLS/RTP/SAVPF 96 97 98 99 100 101 102 122 127 121 125 107 108 109 124 120
10 ...
```

上面是一个特别简单的例子，每一行都是以一个字符开头，后面紧跟着**等于号(=)**，等于号后面是一串字符。

从“v=”开始一直到“m=audio”，这之间的描述是会话级的；而后面的两个“m=”为媒体级。从中可以看出，在该 SDP 描述中有两个媒体流，一个是音频流，另一个是视频流。

2. SDP 的结构

了解了 SDP 的格式，下面我们来看一下 SDP 的结构，它由**会话描述**和**媒体描述**两部分组成。

(1) 会话描述

会话描述的字段比较多，下面四个字段比较重要，我们来重点介绍一下。

第一个，v= (protocol version , 必选)。例子：v=0，表示 SDP 的版本号，但不包括次版本号。

第二个，o= (owner/creator and session identifier , 必选)。例子：o=<username> <session id> <version> <network type> <address type> <address>，该例子是对一个会话发起者的描述。其中，

o= 表示的是对会话发起者的描述；

<username>：用户名，当不关心用户名时，可以用“-”代替；

<session id> : 数字串，在整个会话中，必须是唯一的，建议使用 NTP 时间戳；

<version> : 版本号，每次会话数据修改后，该版本值会递增；

<network type> : 网络类型，一般为 “IN” ，表示 “internet” ；

<address type> : 地址类型，一般为 IP4 ；

<address> : IP 地址。

第三个，Session Name (必选)。例子：s=<session name>，该例子表示一个会话，在整个 SDP 中有且只有一个会话，也就是只有一个 s=。

第四个，t= (time the session is active , 必选)。例子：t=<start time> <stop time>，该例子描述了会话的开始时间和结束时间。其中，<start time> 和 <stop time> 为 NTP 时间，单位是秒；当<start time>和<stop time>均为零时，表示持久会话。

(2) 媒体描述

媒体描述的字段也不少，下面我们也重点介绍四个。

第一个，m= (media name and transport address , 可选)。例子：m=<media> <port> <transport> <fmt list>，表示一个会话。在一个 SDP 中一般会有多个媒体描述。每个媒体描述以 “m=” 开始到下一个 “m=” 结束。其中，

<media> : 媒体类型，比如 audio/video 等；

<port> : 端口；

<transport> : 传输协议，有两种——RTP/AVP 和 UDP ；

<fmt list> : 媒体格式，即数据负载类型 (Payload Type) 列表。

第二个，a=* (zero or more media attribute lines , 可选)。例子：a=<TYPE>或 a=<TYPE>:<VALUES>，表示属性，用于进一步描述媒体信息；在例子中，指属性的类型，a= 有两个特别的属性类型，即下面要介绍的 rtpmap 和 fmp。

第三个，rtptime (可选)。例子：`a=rtptime:<payload type> <encoding name>/<clock rate>[/<encodingparameters>]`。

rtptime 是 rtp 与 map 的结合，即 RTP 参数映射表。

`<payload type>`：负载类型，对应 RTP 包中的音视频数据负载类型。

`<encoding name>`：编码器名称，如 VP8、VP9、OPUS 等。

`<sample rate>`：采样率，如音频的采样率频率 32000、48000 等。

`<encodingparameters>`：编码参数，如音频是否是双声道，默认为单声道。

第四个，fmtp。例子：`a=fmtp:<payload type> <format specific parameters>`。

fmtp，格式参数，即 format parameters；

`<payload type>`，负载类型，同样对应 RTP 包中的音视频数据负载类型；

`<format specific parameters>`指具体参数。

以上就是 SDP 规范的基本内容，了解了上面这些内容后，下面我们来看一下具体的例子，你就会对它有更清楚的认知了。

 复制代码

```
1 v=0
2 o=- 4007659306182774937 2 IN IP4 127.0.0.1
3 s=-
4 t=0 0
5 // 以上表示会话描述
6 ...
7 // 下面的媒体描述，在媒体描述部分包括音频和视频两路媒体
8 m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 110 112 113 126
9 ...
10 a=rtptime:111 opus/48000/2 // 对 RTP 数据的描述
11 a=fmtp:111 minptime=10;useinbandfec=1 // 对格式参数的描述
12 ...
13 a=rtptime:103 ISAC/16000
14 a=rtptime:104 ISAC/32000
15 ...
16 // 上面是音频媒体描述，下面是视频媒体描述
17 m=video 9 UDP/TLS/RTP/SAVPF 96 97 98 99 100 101 102 122 127 121 125 107 108 109 124 120
18 ...
```

```
19 a=rtpmap:96 VP8/90000
20 ...
```



从上面的例子中，你可以清楚地看到在这段 SDP 片段里包括**会话信息**与**媒体信息**。在媒体信息中又包括了**音频流信息**和**视频流信息**。

在音频流和视频流信息中，通过 rtpmap 属性对它们做了进一步的说明。如音频流支持 OPUS 和 ISAC 编码，OPUS 编码的采样率是 48000，双声道，而 ISAC 编码的采样率可以是 16000 或 32000，它们都是单声道。视频流支持 VP8，采样率是 90000。

WebRTC 中的 SDP

WebRTC 对标准 SDP 规范做了一些调整，更详细的信息可以看[这里](#)，它将 SDP 按功能分成几大块：

Session Metadata，会话元数据

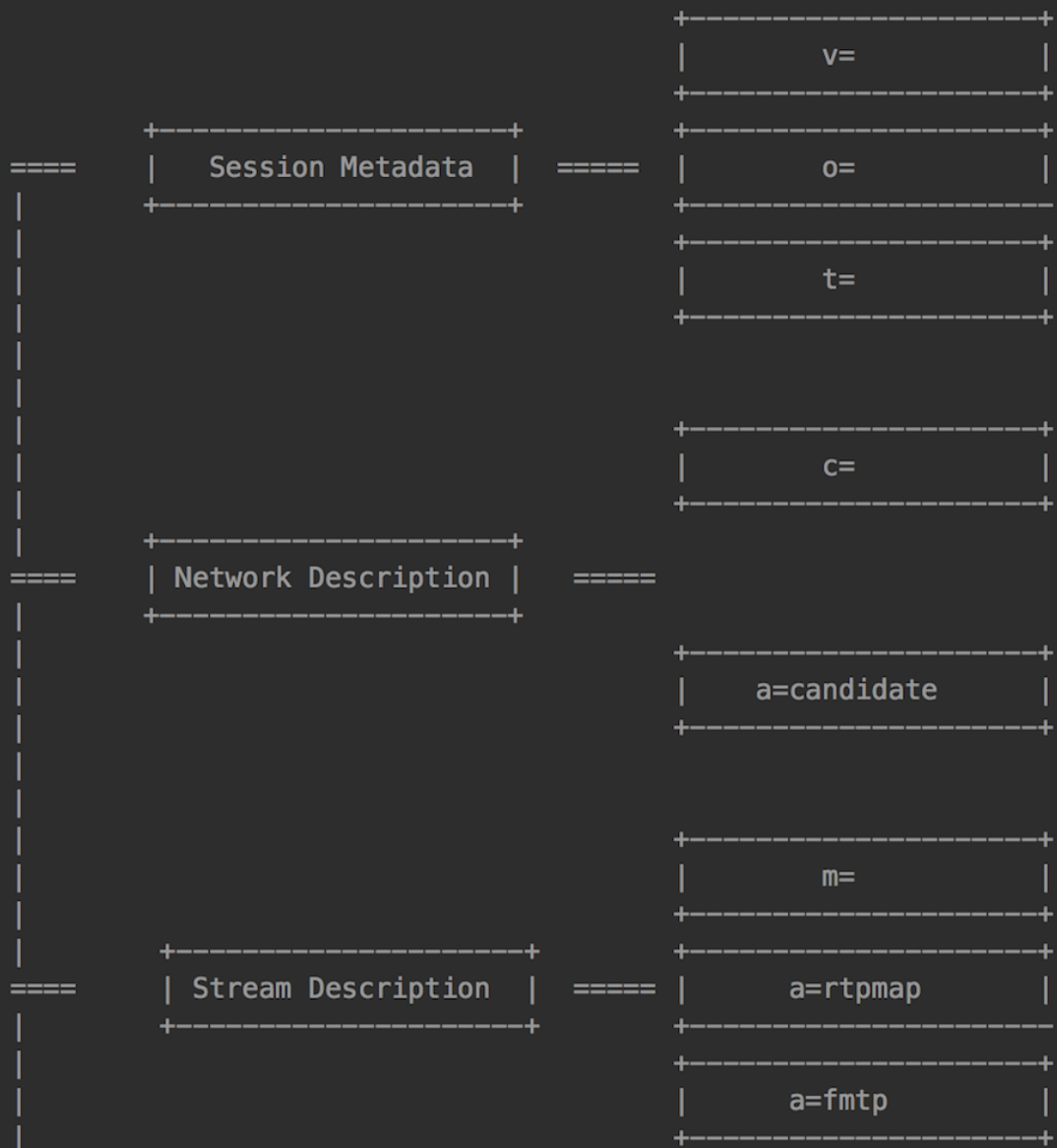
Network Description，网络描述

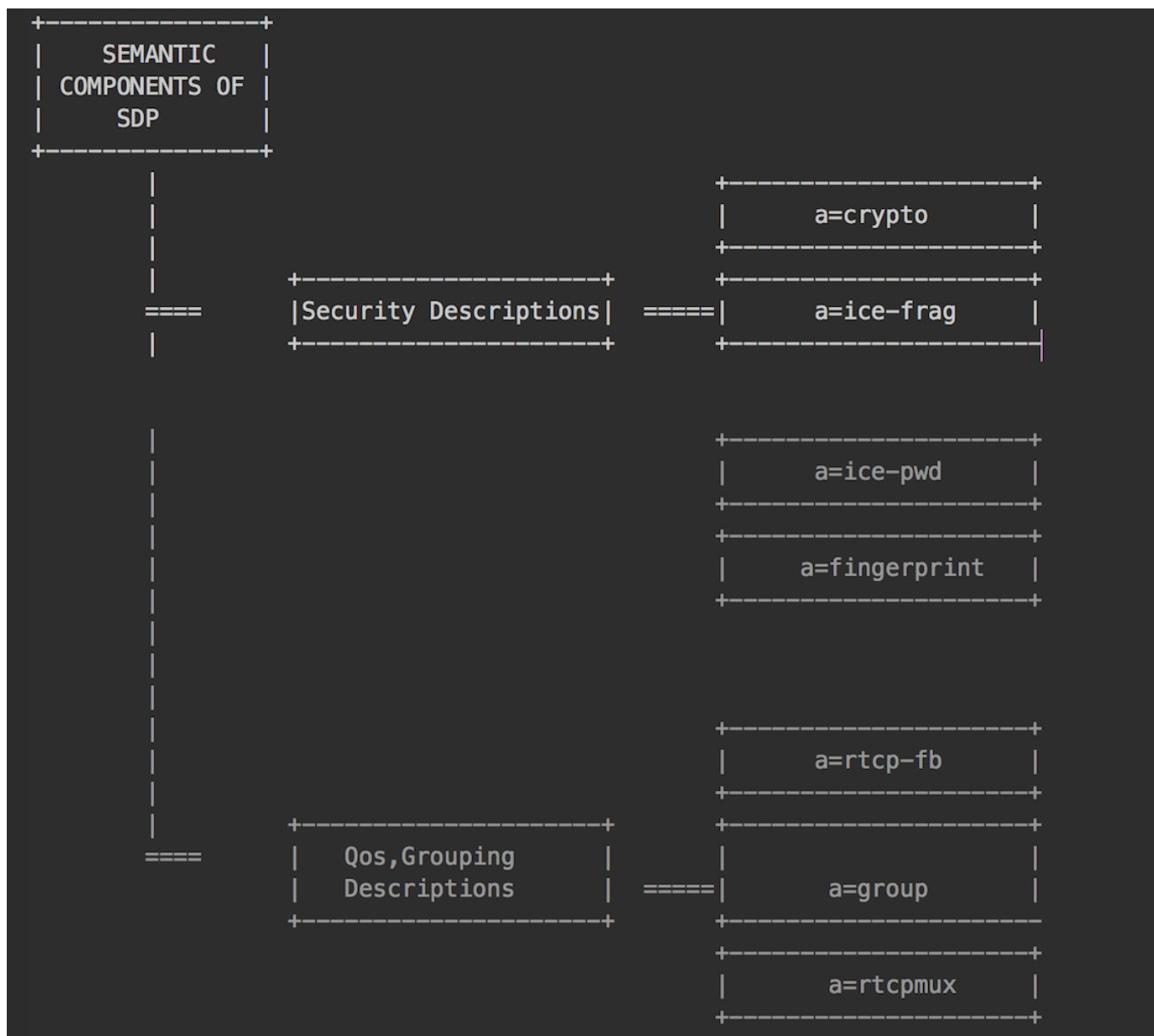
Stream Description，流描述

Security Descriptions，安全描述

Qos Grouping Descriptions，服务质量描述

下面这张图清晰地表达了它们之间的关系：





WebRTC 使用的 SDP 结构图

通过上图我们可以看出，WebRTC 按功能将 SDP 划分成了五部分，即会话元数据、网络描述、流描述、安全描述以及服务质量描述。WebRTC SDP 中的会话元数据（Session Metadata）其实就是 SDP 标准规范中的**会话层描述**；**流描述**、**网络描述**与 SDP 标准规范中的**媒体层描述**是一致的；而**安全描述**与**服务质量描述**都是新增的一些属性描述。下图我们来看一个具体的例子：

[复制代码](#)

```

1 ...
2 //===== 安全描述 =====
3 a=ice-ufrag:1uEe // 进入连通性检测的用户名
4 a=ice-pwd:RQe+y7SOLQJET+duNJ+Qbk7z// 密码，这两个是用于连通性检测的凭证
5 a=fingerprint:sha-256 35:6F:40:3D:F6:9B:BA:5B:F6:2A:7F:65:59:60:6D:6B:F9:C7:AE:46:44:B4
6 ...

```

```

7 //===== 服务质量描述 =====
8 a=rtcp-mux
9 a=rtcp-rsize
10 a=rtpmap:96 VP8/90000
11 a=rtcp-fb:96 goog-remb // 使用 google 的带宽评估算法
12 a=rtcp-fb:96 transport-cc // 启动防拥塞
13 a=rtcp-fb:96 ccm fir // 解码出错, 请求关键帧
14 a=rtcp-fb:96 nack // 启用丢包重传功能
15 a=rtcp-fb:96 nack pli // 与 fir 类似
16 ...

```


上面的 SDP 片段是摘取的 WebRTC SDP 中的安全描述与服务质量描述，这两块描述在标准 SDP 规范中没有明确定义，它更多属于 WebRTC 业务的范畴。

其中，安全描述起到两方面的作用，一方面是进行网络连通性检测时，对用户身份进行认证；另一方面是收发数据时，对用户身份的认证，以免受到对方的攻击。从中可以看出 WebRTC 对安全有多重视了

服务质量描述指明启动哪些功能以保证音视频的质量，如启动带宽评估，当用户发送数据量太大超过评估的带宽时，要及时减少数据包的发送；启动防拥塞功能，当预测到要发生拥塞时，通过降低流量的方式防止拥塞的发生等等，这些都属于服务质量描述的范畴。

为便于你更好地理解和使用 SDP，接下来我再分享一个真实的例子。

下面这段 SDP 是我从一个真实的 1 对 1 场景中截取出来的 WebRTC SDP 的片段。我在这段 SDP 上做了详细的注释，通过上面知识的学习，现在你应该也可以看懂这段 SDP 的内容了。

 复制代码

```

1 //===== 会话描述 =====
2 v=0
3 o=- 7017624586836067756 2 IN IP4 127.0.0.1
4 s=-
5 t=0 0
6 ...
7
8 //===== 媒体描述 =====
9 //===== 音频媒体 =====
10 /*
11 * 音频使用端口 1024 收发数据

```

```
12 * UDP/TLS/RTP/SAVPF 表示使用 dtls/srtp 协议对数据加密传输
13 * 111、103 ... 表示本会话音频数据的 Payload Type
14 */
15 m=audio 1024 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
16
17 //===== 网络描述 =====
18 // 指明接收或者发送音频使用的 IP 地址，由于 WebRTC 使用 ICE 传输，这个被忽略。
19 c=IN IP4 0.0.0.0
20 // 用来设置 rtcp 地址和端口，WebRTC 不使用
21 a=rtcp:9 IN IP4 0.0.0.0
22 ...
23
24 //===== 音频安全描述 =====
25 //ICE 协商过程中的安全验证信息
26 a=ice-ufrag:khLS
27 a=ice-pwd:cxLzteJaJBou3DspNaPsJhlQ
28 a=fingerprint:sha-256 FA:14:42:3B:C7:97:1B:E8:AE:0C2:71:03:05:05:16:8F:B9:C7:98:E9:60:4:
29 ...
30
31 //===== 音频流媒体描述 =====
32 a=rtpmap:111 opus/48000/2
33 //minptime 代表最小打包时长是 10ms，useinbandfec=1 代表使用 opus 编码内置 fec 特性
34 a=fmtp:111 minptime=10;useinbandfec=1
35 ...
36 a=rtpmap:103 ISAC/16000
37 a=rtpmap:104 ISAC/32000
38 a=rtpmap:9 G722/8000
39 ...
40
41 //===== 视频媒体 =====
42 m=video 9 UDP/TLS/RTP/SAVPF 100 101 107 116 117 96 97 99 98
43 ...
44 //===== 网络描述 =====
45 c=IN IP4 0.0.0.0
46 a=rtcp:9 IN IP4 0.0.0.0
47 ...
48 //===== 视频安全描述 =====
49 a=ice-ufrag:khLS
50 a=ice-pwd:cxLzteJaJBou3DspNaPsJhlQ
51 a=fingerprint:sha-256 FA:14:42:3B:C7:97:1B:E8:AE:0C2:71:03:05:05:16:8F:B9:C7:98:E9:60:4:
52 ...
53
54 //===== 视频流描述 =====
55 a=mid:video
56 ...
57 a=rtpmap:100 VP8/90000
58 //===== 服务质量描述 =====
59 a=rtcp-fb:100 ccm fir
60 a=rtcp-fb:100 nack // 支持丢包重传，参考 rfc4585
61 a=rtcp-fb:100 nack pli
62 a=rtcp-fb:100 goog-remb // 支持使用 rtcp 包来控制发送方的码流
63 a=rtcp-fb:100 transport-cc
```



从上面这段 SDP 中你应该可以总结出：**SDP 是由一个会话层和多个媒体层组成的；而对于每个媒体层，WebRTC 又将其细划为四部分，即媒体流、网络描述、安全描述和服务质量描述。**

并且在上面的例子中有两个媒体层——音频媒体层和视频媒体层，而对于每个媒体层，也都有对应的媒体流描述、网络描述、安全描述及服务质量描述，是不是非常清晰？

小结

本文为你详细描述了 SDP 的标准规范以及 WebRTC 对 SDP 规范的修改，为便于你理解，还通过一系列实际的例子，向你展示了在 WebRTC 中使用的 SDP 是什么样子。

总结起来就是，SDP 是由一个会话层与多个媒体层组成，每个媒体层又分为媒体流描述、网络描述、安全描述和服务质量描述，而每种描述下面又是一堆细节的知识点。

当然，通过本篇文章你也许不能一下将 SDP 的所有细节都了解清楚。但有了这个基础之后，通过后面不断地积累，最终你将在大脑中形成一个 SDP 的知识图谱，到那时你再看 SDP 时就游刃有余了。

思考时间

学习完上面的正文后，现在请你思考一下，在 SDP 中如何设置音视频的传输码率呢？

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

从 0 打造音视频直播系统

手把手教你打造实时互动音视频直播系统

李超

新东方音视频直播技术专家
前沪江音视频架构师



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 06 | WebRTC中的RTP及RTCP详解

下一篇 08 | 有话好商量，论媒体协商

精选留言 (8)

写留言



诸葛亮了

2019-07-31

浏览器和ios app之间用webrtc建立视频直播，ios app退出到后台，再次进入时浏览器的直播会卡住是什么原因呢？

展开

作者回复: 因为你的网络连接断了，切回来之后要重新来一遍。你想你都切到后台了，如果应用程序还在跑着流量，而且流量要花钱你乐意吗？

1

2



Geek_miao

2019-08-02

这里边设置传输码率 a=rtcp-fb:100 后边



Jason

2019-07-31

a=fmtp:122 profile-level-id=008016;max-mbps=42000;max-fs=3600;max-smbps=323500
--max-mbps=42000

展开 ▾

1



良师益友

2019-07-30

多个用户加入一个房间，需要sdp关于编解码部分必须一样吗？

作者回复: 不需要，每路流都有它自己的PayloadType, webrtc会根据 PayloadType进行解码



许童童

2019-07-30

短期记忆已经记住了SDP，晚上再回顾一下。

展开 ▾

作者回复: 这个一定要弄清楚，清楚之后才能对 WebRTC有更深入的理解



佛学渣

2019-07-30

是在fmtp下设置传输码率吗？

展开 ▾

作者回复: 不是，你再找找答案



1





佛学渣

2019-07-30

I传输协议好像还有RTP/SAVP吧...

展开 ∨

作者回复: RTP/RTCP、SRTP/SRTCP, 后面会有讲到!



Beast-Of-Prey

2019-07-30

打卡 一遍过去记不住 明天再读一遍

展开 ∨

作者回复: 这块一定要多看几遍

