E-mail：bryanj@163.com
译者： Bryan.Wong（王晶，宁夏固原）

实时流协议(RTSP)

摘要：
实时流协议（RTSP）是应用层协议，控制实时数据的传送 。RTSP提供了一个可扩展框架，使受控、按需传输实时数据（如音频与视频）成为可能。数据源包括现场数据与存储在剪辑中的数据。本协议旨在于控制多个数据发送会话，提供了一种选择传送途径（如UDP、组播UDP与TCP）的方法，并提供了一种选择基于RTP (RFC1889)的传送机制的方法。

目录：

1 介绍
1.1 目的
1.2 要求
1.3 术语
1.4 协议特性
1.5 RTSP扩展
1.6 整体运作
1.7 RTSP状态
1.8 与其他协议的关系
2 符号协定
3 协议参数
3.1 RTSP版本
3.2 RTSP URL
3.3 会议标识
3.4 会话标识
3.5 SMPTE 相对时间戳
3.6正常播放时间
3.7 绝对时间
3.8 选项标签
3.8.1 用IANA注册新的选项标签
*4 RTSP消息
4.1 消息类型
4.2 消息头
4.3 消息主体
4.4 消息长度
*5 普通头部段
*6 请求
6.1 请求行
6.2 请求消息头段
*7 响应
7.1 状态行
7.1.1 状态码和原因短语
7.1.2 响应头部段
*8 实体
8.1 实体头部域
8.2 实体主体 24
*9 连接
9.1 流水线化 25
9.2 可靠性及确认 25
*10 方法定义 25
10.1 可选项 26
10.2 描述 26
10.3 通知 26
10.4 建立 26
10.5 播放 27
10.6 暂停 27
10.7 断开 27
10.8 获取参数 28
10.9 设置参数 28
10.10 重定向 28

10.11 录制 29
10.12 嵌入（交织）的二进制数据 29
*11状态码定义 29
11.1成功2xx 30
11.1.1 存储空间低 250 30
11.2 重定向 3xx 31
11.3 客户端错误 4xx 31
11.3.1方法不允许 32
11.3.2无法理解参数 32
11.3.3会议未找到 33
11.3.4 带宽不足 33
11.3.5 会话未找到 34
11.3.6 本状态下该方法无效 34
11.3.7 头部域与资源不匹配 34
11.3.8 无效范围 35
11.3.9 参数为只读 35
11.3.10 不允许合操作 36
11.3.11 只允许合操作 36
11.3.12 不支持的传输 36
11.3.13 目标不可达 37
11.3.14 不支持的选项 37
12 头部段定义（Header Field Definitions） 38
12.1 接受 38
12.2 接受-编码 38
12.3 接受-语言 39
12.4 允许（Allow） 39
12.5 授权（Authorization） 40
12.6 带宽 40
12.7 块大小 40
12.8 缓存控制 41
12.9 会议 41
12.10 连接 41
12.11 内容-基础 42
12.12 内容-编码（Content-Encoding） 42
12.13 内容-语言 43
12.14 内容-长度（Content-Length） 43
12.15 内容-位置 43
12.16 内容-类型（Content-Type） 44
12.17 命令序列题头（CSeq） 44
12.18 日期（Date） 44
12.19 过期（Expires） 45
12.20 来自（From） 45
12.21 主机 45
12.22 如果匹配 45
12.23如果-被修改-自从（If-Modified-Since） 46
12.24 最后修改（Last-Modified） 46
12.25 位置（Location） 46
12.26 代理认证 47
12.27 代理要求 47
12.28 公布 47
12.29 范围 49
12.30 提交方（Referer） 49
12.31 稍后重试 49
12.32 要求 49
12.33 RTP信息 49
12.34 倍速（Scale）
12.35 速度 49
12.36 服务器（Server） 49

12.37 会话 49
12.38 时间戳 49
12.39 传输 49
12.40 不支持 49
12.41 用户代理（User-Agent） 49
12.42 变化 49
12.43 通过 49
12.44 WWW-认证（WWW-Authenticate） 50
*13 缓存 50
*14 例子 50
14.1 按需点播（单播） 50
14.2 容器文件的流化 51
14.3 单个流容器文件 51
14.4 实况媒体表示的组播 51
14.5 在存在的会话中播放媒体 51
14.6 录制 52
*15 语法 52
15.1 基本语法 52
16 安全考虑（Security Considerations） 52
*附录A RTSP协议状态机 53
*A.1 客户端状态机 53
*A.2 服务器端状态机 53
*附录B 与RTP协议的交互 53
*附录C 使用SDP进行RTSP会话描述 54
+C.1 定义 54
o C.1.1 控制URL 55
o C.1.2 媒体流 55
o C.1.3 有效载荷类型 55
o C.1.4 详细格式参数 55
o C.1.5 表示的范围 56
o C.1.6 有效时间 56
o C.1.7 连接信息 56
o C.1.8 实体标签 57
+C.2 合控制不可用 57
+C.3 合控制可用 57
*附录D 最小RTSP实现 58
+D.1 客户端 58
D.1.1基本回放 58
D.1.2 认证enabled 58
+D.2 服务器 59
D.2.1基本回放 59
D.2.2认证enabled 59
*附录E 作者地址 60
*附录F 致谢 60
*参考书目 60
*版权申明 61

1 介绍

1.1 目的

实时流协议（RTSP）建立并控制一个或几个时间同步的连续流媒体，比如音频或视频。尽管在连续媒体流中有可能插入控制流（见10.12节），但RTSP本身通常并不发送连续媒体流。换言之，RTSP充当多媒体服务器的"网络遥控器"。

表示描述定义了流的控制操作的集合，但本文并没有规定表示描述的格式。

RTSP没有"连接"这个概念，而由RTSP会话(session)代替（服务器端保持一个由识别符标记的会话）。RTSP会话没有绑定传输层连接（如TCP连接）。在RTSP会话期间，RTSP客户端可以打开或关闭多个到服务器端的可靠传输连接以发出RTSP请求。但也可以使用无连接传输协议，比如UDP，来发送RTSP请求。

RTSP所控制的流可能用到RTP，但RTSP的操作并不依赖用来传送连续媒体的传输机制。实时流协议在语法和操作上有意地类似于HTTP/1.1，使得HTTP的扩展机制大都可加入RTSP。尽管如此，RTSP在很多重要方面与HTTP有所不同：

*RTSP引入了很多新方法并且有不同的协议标识符。

*RTSP服务器在绝大多数默认情况下需要维持状态，而HTTP是无状态协议。

*RTSP客户机和服务器都可以发出请求。

*数据由信带外的另一个协议传送（但有一个特例）。

*RTSP使用ISO 10646(UTF-8) 而不是ISO 8859-1，以配合当前HTML的国际化。

*RTSP的URI请求时总是包含绝对URI。而由于历史原因造成的后向兼容性问题，HTTP/1.1只在请求中包含绝对路径，把主机名放入单独的头部域中。

当只有一个IP的主机要提供多个文档树时，可使"虚拟主机"的实现更简单。

协议支持以下操作：

从媒体服务器上获得媒体：

用户可通过HTTP或其它途径请求一个表示描述。如果该表示是组播，表示描述就包含用于该连续媒体的的多播地址和端口。如表示仅通过单播发送给用户，用户为了安全应起见要提供目的地址。

邀请媒体服务器进入会议：

媒体服务器可被"邀请"加入已存在的的会议，包括向该表示内回放媒体，或记录此表示中的一部分或全部媒体。这种模式在分布式教学应用上很有用。会议中的各方可轮流"按网络遥控器的按钮"。

将媒体加到已存在的表示中：

现场表示 的专用概念。当服务器可以告诉客户端"可以附加媒体"时有用。

和HTTP/1.1类似，RTSP的请求可由代理、通道与缓存处理。

1.2 要求

在本文档中的关键字"必须"，"必须不"、"需要"、"必须"、"必须不"、"应该"、"不应该"、"推荐"、"可能"、和"可选的"，都和RFC2119 [4]中的解释一致。

1.3 术语

一些HTTP/1.1的术语被采用。这里没有举出的术语，其定义与HTTP/1.1相同。

合控制：

服务器使用一条时间线对多个流进行控制。对音频/视频的回放来讲，这意味着客户端仅需发送一条播放或者暂停消息就可同时控制音频和视频的回放。

会议：

多方参与的多媒体表示，这里的多方意味着大于或等于一方。

客户端：

指请求媒体服务器上连续流媒体数据的客户端。

连接：

以通讯为目的，在传输层建立的两个程序间的虚拟信道。

容器文件：

可以容纳多个媒体流的文件，而这些媒体流共同播放时通常还包含一个表示。RTSP服务器可以为这些容器文件提供合控制，但容器文件的概念本身并不包含在本协议中。

连续媒体：

接受器和数据源之间存在时序关系的数据。也就是说，接受器需要重放原来存在于源数据中的时序关系。最普通的连续媒体的例子是音频和动画视频。连续媒体可以是实时的（交互的），它们在源和接受器之间是一种紧密的时序关系；或者是流（回放）的形式，时序关系没那么严格。

实体：

请求或者响应的载荷部分中所传输的信息。实体由信息元组成，而每个信息元由由实体头部域和实体主体组成。实体头部域内是信息格式，实体主体内是信息内容，如第8章所述。

媒体初始化：

数据类型/编码的具体初始化。这包括时钟频率，颜色空间等。客户端请求一个媒体流回放时所需的任何独立于传输的信息，都是在流创建时媒体初始化阶段产生的。

媒体参数:

对于某种特定的媒体类型来说，回放前或者回放中有可能会发生改变的一些参数。

媒体服务器：

提供一个或多个媒体流之回放或录制服务的服务器。同一个表示(presentation)中不同的媒体流可能来自于不同的媒体服务器。媒体服务器可以建在激活该表示(presentation)的Web服务器上，也可以建立在不同的主机上。

媒体服务器重定向：

重新把媒体客户端定向到另外一个媒体服务器。

(媒体)流：

单个媒体实例，比如，一个音频流或者一个视频流，连同一个白板或者共享程序组。当使用RTP时，流包括由RTP会话(session)中同一个源所创建的所有RTP和RTCP包。这和DSM-CC流([5])的定义相同。

消息：

RTSP通讯的基本单元。由15章语法定义的结构化八位位组序列组成，并通过有连接或者无连接协议传送。

参与者：

??一个会议的成员。参与者可以是机器，比如是媒体记录或回放服务器。

表示(presentation)：

作为一个完整的媒体信息，回馈性地表述给客户端的一个或多个流的集合。表示使用下面的表示描述进行表述。大部分情况下，在RTSP中的文字部分中，这暗示集合中的流的合控制，但并非必须。

表示描述(presentation description)：

表示描述包含在表示(presentation)中一个或者多个媒体流的信息。比如，编码，网络地址和内容的信息，的集合。另外，其他IETF协议，如SDP协议使用术语"会话(session)"代替"现场表示"。表示描述可以采用包括会话描述(session description)SDP在内的多种格式。

响应：

??RTSP响应。如果能理解HTTP响应，就能清楚地理解RTSP响应。

请求：

RTSP请求。如果能理解HTTP请求，就能清楚地理解RTSP请求。

RTSP会话(session)：

包括一次RTSP"事务"（transaction）的全过程。比如，一个电影的观看过程。会话(session)一般包括由客户端为连续媒体建立传输机制(SETUP)，使用播放(PLAY)或录制(RECORD)开始传送流，用停止(TEARDOWN)关闭流。

传输初始化：

客户端和服务器端之间关于传输所需的相关信息（端口号，传输协议等）的协商。

1.4 协议特点

RTSP有以下特性：

易于扩展：

可以很容易地向RTSP加入新方法和参数。

易解析：

RTSP可由标准HTTP或MIME解析器解析。

安全：

RTSP重用 了网页安全机制。所有HTTP授权机构如basic (RFC 2068 [2, Section 11.1])和digest (RFC 2069 [8])授权都可直接使用。也可以重用传输层或网络层安全机制。

独立于传输：

RTSP即可使用不可靠数据报协议（UDP）、可靠数据报协议（RDP），如要实现应用级可靠，也可使用可靠流协议如TCP。

多服务器支持：

表示(presentation)中的每个流可放在不同服务器上，客户端自动同不同服务器建立几个并发控制的会话，媒体同步在传输层执行。

录制设备控制：

协议可控制记录和回放设备，以及可在两种模式之间切换的设备（VCR）。

流控制与会议初始化分离：

流控制与邀请媒体服务器入会相分离；仅要求会议初始化协议可提供，或可用来创建具有唯一性的会议标识号。具体地说，SIP或H.323 可用来邀请服务器入会。

适合专业应用：

通过SMPTE 时标，RTSP支持帧级别精度，以支持远程数字编辑。

适合专业应用：

RTSP依赖SMPTE时间戳支持帧级精度，使得可以进行远程数字编辑。

表示描述中立：

协议没强行指定特定的表示或元文件格式，可传达所用的格式类型；然而，表示描述必须至少包含一个RTSP URI。

代理与防火墙友好：

协议需由应用层协同传输层(SOCKS [14])防火墙友好地进行处理。防火墙需要理解SETUP方法，以为UDP媒体流打开一个"洞口"。

HTTP友好：

此处，RTSP明智地重用了HTTP概念，使现有的基础结构可被重用。这些基础结构包括Internet 内容选择平台（PICS：Platform for Internet Content Selection [15,16]），以便通过相关标签访问内容。但由于在大多数情况下，控制连续媒体需要服务器状态， RTSP不仅仅向HTTP 添加方法。

合适的服务器控制：

若客户端能启动一个流，它必须也能停止一个流。服务器不能启动一个用户不能停止的流。

传输协商：

实际处理连续媒体流前，客户端可协商传输方法。

性能协商：

若基础特性被禁用，必须有某种明确的机制让用户决定哪种方法将不不被实现。这允许用户提出适合的用户界面。 例如，如果不允许寻找，用户界面必须能禁止位置条滑动。

早期曾要求RTSP支持多用户，但现在有了更好的方案，就是保证RTSP能很容易扩展成支持多用户即可。因为流的标志可以被多个控制流使用，因此可以"轮换持有控制器"。协议不涉及到多个客户端如何协调入口--这项任务被留给"社会协议"或其他层。

1.5 扩展RTSP

由于不是所有媒体服务器有着相同的功能，媒体服务器有必要支持不同的请求集。例如：

服务器可能只能回放，因此不必支持录制请求。

用于提供现场直播的服务器可能不支持寻找（绝对位置）功能。

一些服务器可能不支持设置流参数，因此不支持GET_PARAMETER和SET_PARAMETER请求。

但服务器应该实现所有12章中要求的标题域。

表示描述(presentation description)应当保证不提出服务器不支持的功能，这种情形和HTTP/1.1 [2]中，[H19.6]所描述的方法不太可能被所有服务器都支持的情形一致。

RTSP 可以如下三种方式扩展，按所支持的改变多少排序：

*已有的方法可以扩展加入新参数，只要这些参数可以被接收方安全地忽略。（这和给一种HTML tag增加新标签是一样的）如果客户端在请求失败时需要一个拒绝确认，需要在请求：字段(见Section 12.32)中加入一个对应于该扩展的标签。

*可以加入新方法。如果接收方不理解请求，它就返回一个501错误码（意指未实现），发送方就不应再尝试这种方法。客户端可以用OPTIONS方法去询问服务器支持的方法。服务器应该在公共回应头里列出它所支持的所有方法。

*可以定义新版本的协议，以支持几乎所有方面的改变（除了版本协议号的位置）。

1.6 整体运作

每个表示和媒体流可用RTSP URL识别。表示组成的整个表示与媒体属性由表示描述(presentation description)文件定义，其格式不在本协议中定义。客户端可以通过HTTP或其它途径（如email）获得此表示描述文件，它没有必要保存在媒体服务器上。

根据此规范的目标，我们假想一个表示描述(presentation description)描述了多个表示(presentation)，每个表示(presentation)维持一个统一的时间轴。为简明但不失一般性，假定表示描述(presentation description)正好包含一个这样的表示(presentation)。表示(presentation)可包含多个媒体流。

表示描述(presentation description)包含组成表示的流的描述，包括它们的编码，语言和使用户可以选择最符合要求媒体的其他参数。在表示描述中，各个由RTSP分别控制的媒体流各有一个RTSP URL。RTSP URL指出了处理具体媒体流的服务器以及存在于该服务器上流的名字。多个媒体流可以放到不同的服务器上，比如音频和视频流可以分别放到不同服务器而实现均分负载。描述（description)还列出了服务器可使用的传输方法。

除媒体参数外，网络目标地址和端口也需要决定。下面区分几种操作模式：

单播：

以用户选择的端口号将媒体发送到RTSP请求的来源处。另一种选择是，用和RTSP相同的可靠流传输媒体 。

多播，服务器选择地址：

媒体服务器选择多播地址和端口，这是现场直播或准点播常用的方式。

多播，用户选择地址：

若服务器要加入正在进行的多播会议，多播地址、端口和密匙由会议描述给出。会议描述的建立不在此规范中讨论。

1.7 RTSP状态

RTSP控制通过与控制通道无关的独立协议发送的流。例如，RTSP控制可能是使用TCP连接，而数据流使用UDP。因此，即使媒体服务器没有收到请求，数据也会继续发送。在会话生命期，单个媒体流可通过不同TCP连接按顺序发出的请求来控制。所以，服务器需要维护"会话状态"以便使RTSP请求和流相互关联。状态之间的转换在附录A中描述。

RTSP中很多方法与状态无关，但下列方法在服务器流资源的定位和应用上起着重要的作用：SETUP, PLAY, RECORD, PAUSE, 和TEARDOWN.

SETUP：

让服务器给流分配资源，启动RTSP会话。

PLAY与RECORD：

启动SETUP所分配的流的数据传输。

PAUSE：

临时暂停流，而不释放服务器资源。

TEARDOWN：

释放流占用的资源，RTSP会话停止，从服务器端退出。

与状态相关的RTSP方法使用会话头部域（Session header field (Section 12.37)）来识别哪个RTSP会话的状态需要处理，在SETUP请求(章节10.4)的响应中，服务器生成会话标识。

1.8 与其他协议关系

RTSP在功能上与HTTP有重叠。它也可能与HTTP相互作用，体现在与流内容的初始接触是通过网页的。目前的协议规范目的在于允许网页服务器与RTSP媒体服务器之间有多种接力点。例如，表示描述(presentation description)可通过HTTP和RTSP得到，这降低了基于浏览器的应用模式的往返传递，也允许完全不依赖HTTP的独立RTSP 服务器与客户端。

但是，RTSP与HTTP 的本质差别在于数据发送以信带外的不同协议 进行。HTTP是不对称协议，用户发送请求，服务器作出响应。RTSP中，媒体用户和服务器都可发送请求。RTSP请求也不是无状态 的；在请求确认后很长时间后，仍可设置参数，继续控制媒体流。

重用HTTP功能至少在两个方面有好处，即安全和代理。要求非常接近，在缓存、代理和授权上采用HTTP功能是有价值的。

虽然大多数实时媒体使用RTP作为传输层协议，RTSP并没有绑定到RTP。

RTSP假设存在可表示包含几个媒体流的表示的静态与临时属性的表示描述格式。

2 符号约定

既然很多定义和语法与HTTP/1.1中相同，这里仅指出它们在HTTP/1.1中定义的位置而并没有拷贝它们到本文档。为简便起见，本文档中[HX.Y]表示对应HTTP/1.1（RFC 2068 [2]）中的X.Y部分。（[译者注：]为更方便学习RTSP，本翻译文档将相关段落完全译出）

与[H2.1]类似，本文对所有机制的说明都是以增广BNF的形式来描述的。此形式在RFC 2234中有详细的描述，唯一的不同是RTSP中以"1#"代替","为分隔符。

****

简单说明增广BNF如下：

增广BNF（augmented BNF）包括下面的结构：

要解释的名词 = 名词解释（name = definition）

规则的名字（name）就是它本身（不带任何尖括号，"<"，">"），后面跟个等号 = ，然后就是该规则的定义。如果规则需要用多个行来描述，利用空格进行缩进格式排版。某些基本的规则使用大写，如SP, LWS, HT, CRLF, DIGIT, ALPHA,等等。定义中还可以使用尖括号来帮助理解规则名的使用。

字面意思（"literal"）

文字的字面意思放在引号中间，若无特别指定，则该段文字是大小写敏感的。

规则1｜规则2（rule1 | rule2）

"｜"表示其分隔的元素是可选的，比如，"是｜否"要选择'是'或'否'。

（规则1 规则2）（(rule1 rule2)）

在圆括号中的元素表明必然出现。如（元素1（元素2｜元素3）元素4）可表明两种意思，即"元素1 元素2 元素4"和"元素1 元素3 元素4"

*规则（*rule）

在元素前加星号"*"表示循环，其完整形式是"<n>*<m>元素"，表明元素最少产生<n>次，最多<m>次。缺省值是0到无限，例如，"1*元素"意思是至少有一个，而"1*2元素"表明允许有1个或2个。

［规则］（[rule]）

方括号内是可选元素。如"［元素1 元素2］"与"*1（元素1 元素2）"是一回事。

N 规则（N rule）

表明循环的次数："<n>（元素）"就是"<n>*<n>（元素）"，也就是精确指出<n>取值。因而，2DIGIT 就是2位数字, 3ALPHA 就是由三个字母组成字符串。

＃规则（#rule）

"#"与"*"类似，用于定义元素列表。

完整形式是"<n>#<m>元素"表示至少有<n>个至多有<m>个元素，中间用","或任意数量的空格（LWS）来分隔，这将使列表非常方便，如"（*LWS 元素 *（ *LWS "," *LWS 元素 ））"就等同于"1#元素"。

空元素在结构中可被任意使用，但不参与元素个数的计数。也就是说，"（元素1），，（元素2）"仅表示2个元素。但在结构中，应至少有一个非空的元素存在。缺省值是0到无限，即"#（元素）"表示可取任何数值，包括0；而"1#元素"表示至少有1个；而"1#2元素"表示有1个或2个。

；注释（; comment）

分号后面是注释，仅在单行使用。

隐含*LWS（implied *LWS）

本文的语法描述是基于单词的。除非另有指定，否则线性空格（LWS）可以在两个邻近符号或分隔符（tspecials）之间任意使用，而不会对整句的意思造成影响。在两个符号之间必须有至少一个分隔符，因为它们也要做为单独的符号来解释。实际上，应用程序在产生HTTP结构时，应当试图遵照"通常方式"，因为现在的确有些实现方式在通常方式下无法正常工作。

****

在本备忘录中，我们用缩进的小型段落来提供背景和动机。这将使没有参与制定RTSP规范的读者更容易理解RTSP中各部分为什么要以该方式来实现。

3 协议参数

3.1 RTSP版本

同[H3.1]定义，仅用RTSP代替HTTP即可。

****

[H3.1]：

RTSP采用主从（<major>.<minor>）数字形式来表示版本。协议的版本政策倾向于让发送方表明其消息的格式及功能，而不仅仅为了获得通讯的特性，这样做的目的是为了与更高版本的RTSP实现 通讯。只增加扩展域的值或增加了不影响通讯行为的消息组件都不会导致版本数据的变化。当协议消息的主要解析算法没变，而消息语法及发送方的隐含功能增加了，将会导致从版本号（<minor>）增加；当协议中消息的格式变化了，主版本号（<major>）也将发生改变。

RTSP消息的版本由消息第一行中的RTSP版本域来表示。

RTSP-Version = "RTSP" "/" 1*DIGIT "." 1*DIGIT

注意，主从版本应当被看作单独的整数，因为它们都有可能增加，从而超过一位整数。因而，RTSP/2.4比RTSP/2.13版本低，而RTSP/2.13又比RTSP/12.3版本低。

版本号前面的0将被接收方忽略，而在发送方处也不应产生。

本文档定义了RTSP协议的1.0版本。发送本规范定义的请求（Request）或响应（Response）消息的应用必须指明RTSP

的版本为"RTSP/1.0"。使用该版本号意味着发送消息的应用至少有条件的遵循本规范。

应用的RTSP版本即为应用至少能有条件遵循的RTSP版本中的最高版本。

当代理及网关收到与其自身版本不同的RTSP请求时，必须小心处理请求的推送，因为协议版本表明发送方的能力，代理或网关不应发出高于自身版本的消息。如果收到高版本的请求，代理或网关必须降低该请求的版本，并响应一个错误。而低版本的请求也应在被推送前升级。代理或网关响应请求时必须和请求的版本相同。

****

3.2 RTSP URL

"rtsp"和"rtspu"前缀表示要通过RTSP协议来定位网络资源。本节详细定义了RTSP URL的语法和语义。

rtsp_URL[?]=[?] ("rtsp:" | "rtspu:") "//" host [":" port] [abs_path]

host[?][?][?]=[?] <合法的Internet主机域名或IP地址(用十进制数及点组成), 见RFC1123，2.1节定义>

port[?] [?][?]=[?] *DIGIT

abs_path 在 [H3.2.1]中定义。

****

[H3.2.1]:

```
    abs_path     = "/" rel_path
   rel_path     = [ path ] [ ";" params ] [ "?" query ]
    path         = fsegment *( "/" segment )
   fsegment     = 1*pchar
   segment      = *pchar
    params       = param *( ";" param )
   param        = *( pchar | "/" )
    scheme       = 1*( ALPHA | DIGIT | "+" | "-" | "." )
   net_loc      = *( pchar | ";" | "?" )
   query        = *( uchar | reserved )
   fragment     = *( uchar | reserved )
    pchar        = uchar | ":" | "@" | "&" | "=" | "+"
   uchar        = unreserved | escape
   unreserved   = ALPHA | DIGIT | safe | extra | national
   escape       = "%" HEX HEX
   reserved     = ";" | "/" | "?" | ":" | "@" | "&" | "=" | "+"
   extra        = "!" | "*" | "'" | "(" | ")" | ","
   safe         = "$" | "-" | "_" | "."
   unsafe       = CTL | SP | <"> | "#" | "%" | "<" | ">"
   national     = <any OCTET excluding ALPHA, DIGIT,
            reserved, extra, safe, and unsafe>
```

权威的URL语法及语义信息请参见RFC1738[4]和RFC1808[9]。

****

注意：fragment和query标识符在这时没有明确的定义，需要到RTSP服务器上解释。

rtsp前缀要求使用可靠协议（在Internet上指TCP协议）发出命令，而rtspu前缀则说明使用不可靠协议（在Internet指UDP协议）。

如是端口为空或没指定，则缺省为554端口。语义如下：拥有被请求的资源的服务器主机通过监听TCP连接(rtsp方案)或主机上相应端口的UDP包(rtspu方案)，来控制所标记的资源。资源的请求URI是rtsp_URL。

应尽可能避免在URL中直接使用IP地址。（请参考RFC1924）

一个表示或者流是通过基于文本的媒体标记来标识的，此媒体标记使用URLs (RFC 1738 [20])中的字符集和转义规则[H3.2]。URLs可以指向一个流或者一个流的集合，即是说，一个表示。请求视情况可以指向一个完整的表示或者表示中的单个流，见第十章。注意，某些请求方法只能用于流，而不能用于表示，反之亦然。

例如：RTSP URL:

rtsp://media.example.com:554/twister/audiotrack

标识了表示"twister"中的音频流，它可以通过发送基于TCP连接的RTSP请求至主机media.example.com的554端口进行控制。

也可以是这样RTSP URL:

rtsp://media.example.com:554/twister

标识了表示"twister"，它可能是由音频和视频流组成的。

这里并没有暗示相关流URL的标准。表示的结构关系和各个流的URL在表示描述中定义。如一个表示描述可能将一个流命名为a.mov，而将整个表示命名为b.mov。

RTSP URL的路径组成对客户端是不透明的，也不暗含任何服务器的具体文件系统结构。

简单替换URL中的前缀后，表示描述同样可以用于非RTSP媒体控制协议。

3.3 会议标识

会议标识采用URI标准编码方法（即是说，LWS被转义为%）编码，并对RTSP不透明。它们能包含任意字节值。【必须】保证

会议标识在全局中的唯一性。在H.323中，将用到会议的标识值。

conference-id =? 1*xchar

会议标识用以允许RTSP会话从媒体服务器参与的多媒体会议中获取参数。这些会议是用该规范之外的协议创建的，例如H.323 [13] 或 SIP [12]协议。这样就不用RTSP客户端显式地提供传输信息，而改用其他方式代替，例如，客户端要求媒体服务器使用会议描述中的值。

3.4 会话标识

会话标识符是非直读 的任意长度的字符串。线性空格必须是URL转义的。会话标识符【必须】随机产生并且【必须】至少由8个字节组成，以保证其难以被猜出。（详见16章）

session-id? =? 1*(ALPHA | DIGIT | safe)

3.5 SMPTE 相对时间戳

SMPTE 相对时间戳表示相对于开始剪辑的时间。相对时间戳以SMPTE时间编码形式表示以保证帧级的访问精度。时间编码的格式为：时：分：秒：帧.子帧，并以 剪辑开始为起点。缺省的SMPTE格式为"SMPTE 30 drop"格式，其帧率是29.97帧每秒。也可能可通过选择使用不同"SMPTE time"来选择其他SMPTE编码格式（如"SMPTE 25"）。帧域（"frames" field）的时间值在0到29之间。30帧每秒和29.97帧每秒的不同之处在于后者除了整十分钟外的每分钟都要丢掉头两个帧（00和01）。忽略帧值为0的帧，子帧以百分之一帧为单位。

```
  smpte-range =  smpte-type "=" smpte-time "-" [ smpte-time ]
 smpte-type  =  "smpte" | "smpte-30-drop" | "smpte-25"
                        ; other timecodes may be added
 smpte-time  =  1*2DIGIT ":" 1*2DIGIT ":" 1*2DIGIT [ ":" 1*2DIGIT ]
               [ "." 1*2DIGIT ]
```

?例如：

? smpte=10:12:33:20-

? smpte=10:07:33-

? smpte=10:07:00-10:07:33:05.01

? smpte-25=10:07:00-10:07:33:05.01

3.6正常播放时间

正常播放时间(NPT)指示流相对于表示(presentation)开始的位置。时间戳由一个十进制小数组成，以秒为单位，小数点左边可以是秒或者以小时：分：秒的形式表示。小数点右边表示秒的小数部分。

表示开始时对应0.0秒。负值没有意义。特殊的常数"now"定义为现场事件当前瞬间。它只能用于现场事件。

在DSM -CC中，正常播放时间（NPT）是这样定义的："直观地讲，NPT是用户和程序联系的时钟。它经常在VCR上数字显示出来。当处于普通播放模式 (倍速= 1)时，NPT正常前进。当处于快进扫描模式时(倍速为大于1的正数)，NPT快速前进。当处于反向扫描模式(倍速小于-1)时，NPT快速后退。当处于暂停模式时，NPT停止。NPT（逻辑上）等同于SMPTE时间编码。

```
 npt-range??=? ( npt-time "-" [ npt-time ] ) | ( "-" npt-time )
npt-time?? =? "now" | npt-sec | npt-hhmmss
npt-sec???=? 1*DIGIT [ "." *DIGIT ]
npt-hhmmss? =? npt-hh ":" npt-mm ":" npt-ss [ "." *DIGIT ]
npt-hh???? =? 1*DIGIT?? ; any positive number
npt-mm???? =? 1*2DIGIT??; 0-59
npt-ss???? =? 1*2DIGIT??; 0-59
```

? 例如：

? npt=123.45-125

? npt=12:05:35.3-

? npt=now-

语法遵循ISO 8601规则。npt-sec标志法便于自动生成， ntp-hhmmss标志法便于人阅读。"now"常数允许客户端请求接收实时反馈而不是存储或者延时的版本。因为对于这种情况而言，绝对时间和0时间都不适用。

3.7 绝对时间

绝对时间表示为ISO 8601时间戳，使用UTC(GMT)时间。秒的小数部分也可能会出现。

```
 utc-range??=? "clock" "=" utc-time "-" [ utc-time ]
utc-time?? =? utc-date "T" utc-time "Z"
utc-date??? =? 8DIGIT?????????; < YYYYMMDD >
utc-time??? =? 6DIGIT [ "." fraction ]? ; < HHMMSS.fraction >
```

比如，1996年11月8日14点37分20.25秒

UTC时间为：

19961108T143720.25Z

3.8 选择标签

选项标签是用来指示RTSP新选项的唯一标识符。这些标签用于要求(Require)(12.32节)和代理要求(Proxy Require)(12.27节)头部域中。

语法：

option-tag = 1*xchar

要建立新的RTSP选项，可以在选项前加入反转域名的前缀（如：对于能访问到foo.com则com.foo.mynewfeature" 是个合适的名字），或者在英特网权威数字分派委员会注册（IANA）新的选项。

3.8.1 用IANA注册新的选项标签

当注册新RTSP选项标签的时候，应该提供以下信息：

*选项的名字和描述。名字长度不限，但是应该不多于20字符。名字【必须不】包含任何空格，控制符或句点。

*指出谁拥有选项的改变控制权（例如，IETF，国际标准化组织，国际电信联盟-T，其他的国际标准化体，一个团体，一个公司，或者一组公司）。

*描述更为详细的参考文档（如果有），比如（按推荐程度排序），RFC，公开发表的论文，专利文档，技术报告，源代码，或者计算机手册。

*对于私有的选项，需要给出联系地址（邮政地址及电子邮箱）。

4 RTSP消息

RTSP是基于文本的协议，采用ISO 10646 字符集和UTF-8编码方案。每行结束处行以CRLF标记，但接收方需有能力将CR和LF自行解释成行终止符。

基于文本的协议使得易于以自描述方式增加可选参数。由于参数的数量和命令出现的频率较低，处理效率不予考虑。如定义得较仔细，文本协议很容易以脚本语言（如：Tcl、Visual Basic与Perl）实现研究原型。

10646字符集避免了繁琐的字符集切换，但若应用程序使用US-ASCII字符集，它将不可见 。RTCP也采用这种编码方案。ISO 8859-1通过在高位填充0，直接转成Unicode。标志位不为0的ISO 8859-1字符被表示如100001x 10xxxxxx.。(见 RFC 2279 [21])

RTSP信息可通过任何8-bit clean 的低层传输协议传送。

请求包括方法、方法作用于其上的对象和进一步描述方法的参数。除非另外说明，否则方法是幂等 的。方法还被设计为在服务器端只需要少量或不需要状态维护。

4.1 消息类型

见[H4.1]

****

[H4.1]:

RTSP消息由客户端到服务器的请求和由服务器到客户端的响应组成。

RTSP -message = Request | Response ; RTSP /1.0 messages

请求（Request）和响应（Response）消息都使用RFC822中实体传输部分规定（作为消息中的有效载荷）的消息格式。两者的消息都可能包括一起始行，一个或多个头部域（headers）、一行表示头部域结束的空行（即CRLF前没有内容的行），和一个消息主体（message-body, 可选）。

generic-message = start-line *message-header

CRLF

[message-body]

start-line = Request-Line | Status-Line

为了健壮性考虑，服务器应该忽略任何在期望收到请求行时收到的空行。换句话说，如果服务器正在读协议流，在一个消息开始时如果首先收到了CRLF，这个CRLF符应被忽略。

****

4.2 消息头部

见[H4.2]。

****

[H4.2]:

RTSP头部域，包括主头部（General-Header,4.3节）、请求头部（Request-Header ,5.2节）、响应头部（Response-Header ,6.2节）及实体头部（Entity-Header,7.1节），都遵照RFC822-3.1节[7]给出的通用格式定义。每个头部域由后紧跟冒号的名字，单空格（SP），字符及域值组成。域名是大小写敏感的。虽然不提倡，头部域还是可以扩展成多行使用，只要这些行以一个以上的SP或HT开头就行。

RTSP-header = field-name ":" [field-value] CRLF

field-name = token

field-value = *(field-content | LWS)

field-content = <the OCTETs make up the field-value

and consisting of either *TEXT or combinations

of token, tspecials, and quoted-string>

头部域接收的顺序并不重要，但良好的习惯是，先发送主头部，然后是请求头部或响应头部，最后是实体头部。

当且仅当头部域的全部域值都用逗号分隔的列表示时（即，#（值）），多个有相同域名的RTSP头部域才可以表示在一个消息里。而且必须能在不改变消息语法的前提下，将并发的域值加到第一个值后面，之间用逗号分隔，最终能将多个头部域结合成"域名：域值"对。

****

4.3 消息主体

见[H4.3]。
****

[H4.3]:

RTSP消息的消息主体（如果有）用来携带请求或响应的主体。仅在使用传输编码(Transfer-Encoding)时消息主体和实体主体才有所不同，这种情况在传输编码头部域中有详细说明。（见[H14.40]）

message-body = entity-body

| <entity-body encoded as per Transfer-Encoding>

传输编码必须能解释所有保证传输安全和正确的应用程序的传输编码。传输编码是消息而不是实体的一个属性，因此可以由任一应用程序随着请求/响应链添加或者删除。

什么时候允许消息带消息体的规则在请求和响应两种情况下有所不同。

在请求中有无消息主体的标志是是否包含内容长度或请求消息头部域中的传输编码头部域。只有当请求方法允许有实体主体的时候才能在请求中包含消息主体。

而对于响应消息来说，无论消息中是否存在消息主体都与请求方法和响应状态编码无关。所有响应头部请求方法的消息都不能包含消息主体，尽管有时会因为存在实体 头部域而使人产生误解。所有1××（信息），204（无内容），304（未修改）响应都不包含消息主体。而其他响应则都包含主体，尽管其长度有可能长度为零。

****

4.4 消息长度

当信息体包含在信息中时，信息体长度由如下因素决定（按优先度排列）：

1. 任何【必须不】包含消息体message body的响应消息（如1XX，204，及304响应），总是在头域后的第一个空行后就结束，而不管实体头部域是否出现在信息中。（注意：空行中只有CRLF。）

2. 如果存在内容长度头部域（Content-Length header field），它的值（单位为byte）就表示消息体的长度。如果此头部域没有出现，则假设其值为0。

3. 通过服务器关闭连接。（关闭连接不能被用于指示请求主体（request body）的结束，因为那样将使服务器无法回送响应。）

注意：RTSP目前并不支持HTTP/1.1"块"传输编码(见 [H3.6])，需要有内容头部域。

假如返回了长度适当的表示描述，服务器应该总是可以确定它的长度--即便它是动态产生，使得没有必要采用块传输编码。如有实体，即使必须有内容长度，且长度没显式给出，上述规则也可确保行为的合理性。

5 普通头部域

除了Pragma、Transfer-Encoding 和 Upgrade头部，其余见[H4.5]

```
general-header  =  Cache-Control  ; Section 12.8
                |  Connection     ; Section 12.10
                |  Date           ; Section 12.18
                |  Via            ; Section 12.43
```

6 请求

从客户端到服务器端或与之相反的请求消息，在消息首行中包括：应用于资源的方法、资源的标识符及所使用的协议。

```
Request = Request-Line ; 6.1节
  *( general-header ; 5章
  | request-header ; 6.2节
  | entity-header ) ; 8.1节
  CRLF
  [ message-body ] ; 4.3节
```

6.1 请求行

请求行 = 方法 空格 请求-URI 空格 RTSP-版本 CRLF

```
Method      =      "DESCRIBE"           ; Section 10.2
            |      "ANNOUNCE"           ; Section 10.3
            |      "GET_PARAMETER"      ; Section 10.8
            |      "OPTIONS"            ; Section 10.1
            |      "PAUSE"              ; Section 10.6
            |      "PLAY"               ; Section 10.5
            |      "RECORD"             ; Section 10.11
            |      "REDIRECT"           ; Section 10.10
            |      "SETUP"              ; Section 10.4
            |      "SET_PARAMETER"      ; Section 10.9
            |      "TEARDOWN"           ; Section 10.7
            |      extension-method
extension-method = token
Request-URI = "*" | absolute_URI
RTSP-Version = "RTSP" "/" 1*DIGIT "." 1*DIGIT
```

6.2 请求头部域

```
request-header =        Accept                  ; Section 12.1
               |        Accept-Encoding         ; Section 12.2
               |        Accept-Language         ; Section 12.3
               |        Authorization           ; Section 12.5
               |        From                    ; Section 12.20
               |        If-Modified-Since       ; Section 12.23
               |        Range                   ; Section 12.29
               |        Referer                 ; Section 12.30
               |        User-Agent              ; Section 12.41
```

注意：相对于HTTP/1.1[2]而言，RTSP请求总是包含绝对URL（包括rtsp或rtspu前缀，主机，端口号）而不仅仅是绝对路径。

HTTP/1.1 要求服务器能够理解绝对URL，但还是期望客户端使用主机请求头部。这样做完全是为了向后兼容HTTP/1.0服务器端，因此在RTSP中不需要这样做。

在请求-URI中星号"*"表示此请求不用于其他资源，只用于服务器本身，且只能在该方法对于资源并非必要方法时才能使用。如下面的例子：

OPTIONS * RTSP/1.0。

7 响应

使用 [H6] 的规则，只是其中的HTTP版本号要被替换成RTSP版本号。还有，RTSP增加了一些状态码，也弃用了一些HTTP状态码。在表一中，对有效的响应码和可用的方法进行了定义。

在收到并解释了一条请求消息后，接收方以一条RTSP响应消息作为回应。

```
Response    =    Status-Line        ; Section 7.1
            *(   general-header     ; Section 5
            |    response-header    ; Section 7.1.2
            |    entity-header )    ; Section 8.1
                 CRLF
                 [ message-body ]   ; Section 4.3
```

7.1 状态行

响应消息的第一行就是状态行，它由协议版本、数字形式的状态码、与状态码对应的文本解释依次组成，各元素间以空格（SP）分隔，除了结尾的CRLF外，不允许出现CR或LF符。

状态行 = HTTP-版本 空格 状态码 空格 原因解释 CRLF

7.1.1 状态码和原因解释

状态码（Status-Code）由3位数字组成，表示请求是否被理解或被满足。这些状态码的完整定义在第十一章。原因解释（Reason-Phrase）是用简短的文字来描述状态码产生的原因。状态码用来支持自动操作，原因解释用来方便人的查看。客户端不需要检查或显示原因解释。

状态码的第一位数字定义了响应的类别，后面两位数字没有具体分类。首位数字有5取值可能：

*1xx：通知 - 已收到请求，继续处理

*2xx：成功 - 操作被成功接收和理解，并被接受

*3xx：重定向 - 要完成请求必须进行进一步操作。

*4xx：客户端出错 - 请求有语法错误或无法实现

*5xx：服务器端出错 - 服务器无法满足合法的请求。

HTTP/1.0的状态码、对应原因解释在下面给出。下面的原因解释只是建议采用，可用其他等价形式替换，而不会对协议造成影响。注意：RTSP采用了大多数HTTP/1.1 [2]状态码，并增加了一些形如x50的RTSP特有的状态码以避免与最新定义的HTTP状态码冲突。

```
状态码 =    "100"     ; 继续
       |    "200"     ; OK
       |    "201"     ; 已创建
       |    "250"     ; 存储空间不足
       |    "300"     ; 有多个选项
       |    "301"     ; 被永久移除
       |    "302"     ; 被临时移除
       |    "303"     ; 见其他
       |    "304"     ; 没有修改
       |    "305"     ; 使用代理
       |    "400"     ; 错误的请求
       |    "401"     ; 未通过认证
       |    "402"     ; 需要付费
       |    "403"     ; 禁止
```

| "404" ; 没有找到
| "405" ; 不允许该方法
| "406" ; 不接受
| "407" ; 代理需要认证
| "408" ; 请求超时
| "410" ; 不在服务器
| "411" ; 需要长度
| "412" ; 预处理失败
| "413" ; 请求实体过长
| "414" ; 请求-URI过长
| "415" ; 媒体类型不支持
| "451" ; 不理解此参数
| "452" ; 找不到会议
| "453" ; 带宽不足
| "454" ; 找不到会话
| "455" ; 此状态下此方法无效
| "456" ; 此头部域对该资源无效
| "457" ; 无效范围
| "458" ; 参数是只读的
| "459" ; 不允许合控制
| "460" ; 只允许合控制
| "461" ; 传输方式不支持
| "462" ; 无法到达目的地址
| "500" ; 服务器内部错误
| "501" ; 未实现
| "502" ; 网关错误
| "503" ; 无法得到服务
| "504" ; 网关超时
| "505" ; 不支持此RTSP版本
| "551" ; 不支持选项
| 扩展码

扩展码 = 3位数字

原因解释 = *<文本, 包括 CR, LF>

RTSP状态码是可扩展的。RTSP应用程序不被要求了解全部注册的状态码，当然很明显这种了解是被期望的。尽管如此，应用程序【必须】理解任何一个状态码的首位所标识的类别，将任何不理解的状态码等同为那个类的x00状态码，只是【必须不】缓存无法识别 的响应。例如，如果客户端收到一个无法识别的状态码431，可以安全地假定是请求出了问题，认为其响应的状态码就是400。在这种情况下，用户界面应当在把响应消息的实体显示给用户，因为实体中可能包括一些人类可以识别的关于此非正常状态的描述信息。

代码	原因	
100	继续	所有
200	OK	所有
201	已创建	录制
250	存储空间不足	录制
300	有多个选项	所有
301	被永久移除	所有
302	被临时移除	所有
303	见其他	所有
305	使用代理	所有
400	错误的请求	所有
401	未通过认证	所有
402	需要付费	所有
403	禁止	所有
404	没有找到	所有
405	不允许该方法	所有
406	不接受	所有
407	代理需要认证	所有
408	请求超时	所有
410	不在服务器	所有

411 需要长度 所有
412 预处理失败 DESCRIBE, SETUP
413 请求实体过长 所有
414 请求-URI过 所有
415 媒体类型不支持 所有
451 不理解此参数 SETUP
452 找不到会议 SETUP
453 带宽不足 SETUP
454 找不到会话 所有
455 此状态下不接受此方法 所有
456 此头部域对该资源无效 所有
457 无效范围 PLAY
458 参数是只读的 SET_PARAMETER
459 不允许合控制 所有
460 只允许合控制 所有
461 传输方式不支持 所有
462 无法到达目的地址 所有
500 服务器内部错误 所有
501 未实现 所有
502 网关错误 所有
503 无法得到服务 所有
504 网关超时 所有
505 不支持此RTSP版本 所有
551 不支持选项 所有

表一: 状态码及适用RTSP方法

7.1.2 响应头部域

响应头部域使得请求接收方可以发送不能放在状态行中的附加响应信息。这些头部域提供服务器的信息，以及对请求-URI所指定资源进行进一步访问信息。

```
response-header =  Location            ; Section 12.25
                |  Proxy-Authenticate   ; Section 12.26
                |  Public              ; Section 12.28
                |  Retry-After         ; Section 12.31
                |  Server              ; Section 12.36
                |  Vary                ; Section 12.42
                |  WWW-Authenticate    ; Section 12.44
```

响应头部域名只有在与协议版本的变化结合起来后，才能进行可靠的扩展。尽管如此，响应头部域语法中可能会加入新的或实验性的头部域--只要通讯各方能识别它是响应头部域，其语法就可使用。而无法识别的头部域都将被视为实体头部域。

8 实体

如不受请求方法或响应状态码限制，请求和响应消息可以传送任何实体。实体由若干实体头部域和一个实体主体组成，但有些响应仅包括实体头部域。

在本章中，根据谁发送实体、谁接收实体，用户和服务器都可能指发送者或者接收者。

8.1 实体头部域

实体头部域定义了实体主体的可选元信息；如没有实体主体，那些定义则是关于请求标识的资源的。

```
entity-header   =  Allow             ; Section 12.4
                |  Content-Base       ; Section 12.11
                |  Content-Encoding   ; Section 12.12
                |  Content-Language   ; Section 12.13
                |  Content-Length     ; Section 12.14
                |  Content-Location   ; Section 12.15
                |  Content-Type       ; Section 12.16
                |  Expires            ; Section 12.19
                |  Last-Modified      ; Section 12.24
                |  extension-header
extension-header  =  message-header
```

扩展头部机制允许定义附加实体头部域，而不用改变协议，但并不能假定接收者能识别这些附加域。不被识别的头部域应被接收者忽略，而让代理发送。

8.2 实体主体

见[H7.2]及其全部子章节

9 连接

RTSP请求可以几种不同方式传送:

*1、持久传输连接，用于多个请求-响应传输。

*2、每个请求-响应传输对应一个连接。

*3、无连接模式。

传输连接类型由RTSP URI（见3.2节）来定义。"rtsp" 方案说明需要持续连接；而"rtspu"方案，则要求不建立连接就直接发送RTSP请求。

和HTTP不同的是，RTSP允许媒体服务器给媒体用户发送请求。然而，这仅在持久连接模式中才支持，否则媒体服务器没有可靠途径到达用户。这也是通过防火墙从媒体服务器传送请求到用户的唯一办法。

9.1 管道

支持持久连接或无连接的客户端可能会使用"管道方式"传送请求（即是说：发送多个请求而不需等待单个的响应）。服务器必须以和收到的请求同样的顺序发出响应。

9.2 可靠性及确认

接受方需要确认请求，除非请求是发给多播组。如没有确认信息，发送者可在超过一个来回时间（RTT） 后重发同一信息。RTT在TCP中估计，初始值为500 ms。一个实现可能会缓存最后所测量的RTT，作为将来连接的初始值。

如使用可靠传输协议来承载RTSP，则请求不允许重发，RTSP应用程序必须依赖低层传输协议来保证可靠性。

如低层可靠传输协议（如TCP）和RTSP应用程序都重发请求，有可能每次丢包都导致两次重传。由于传输栈在第一次尝试到达接收者以前不会发送应用层重传，所以接收者并不能充分利用应用层重传。若丢包由网络阻塞引起，多个层上的的共同重发将使阻塞进一步恶化。

如果RTSP被用在小RTT 网络，使用标准的初始TCP RTT时间测量优化过程，像T/TCP (RFC 1644) [22]那样，将是有益的。

时间戳头部（见章节12.38）被用来避免重传后乱序的问题[23, p. 301]，以及回避对Karn算法的依赖。

每个请求都带有一个序列号，位于CSeq头部（见章节12.17），每发出一个单独的请求，这个序列号就加一。如果由于缺少确认而重发一个请求，该请求必须携带原来的序列号（即是说，序列号不增加）。

支持RTSP的系统必须支持通过TCP承载RTSP，然后也可以支持通过UDP承载。UDP和TCP的默认RTSP端口都是554.

一定数量的发往同一个控制末端的包可以放进一个低层PDU或者封装进一个TCP流里。RTSP数据中间可以交叉插入RTP和RTCP包。和HTTP不同，一条RTSP消息只要包含载荷（payload），就必须包含内容长度头部。否则，一个RTSP包通过最后一个消息头部后的第一个空行马上截止。

10 方法定义

方法关键字表示将要用于请求-URL所指示的资源上的方法。方法是大小写敏感的。 将来可能会定义新的方法。方法名称不应以$符号（数字24）开始，并且必须是一个关键字。所有的方法被列在表二上。

方法	方向	对象	要求
DESCRIBE	C->S	P,S	推荐
ANNOUNCE	C->S, S->C	P,S	可选
GET_PARAMETER	C->S, S->C	P,S	可选
OPTIONS	C->S, S->C	P,S	必要
			(S->C: 可选)
PAUSE	C->S	P,S	推荐
PLAY	C->S	P,S	必要
RECORD	C->S	P,S	可选
REDIRECT	S->C	P,S	可选
SETUP	C->S	S	必要
SET_PARAMETER	C->S, S->C	P,S	可选
TEARDOWN	C->S	P,S	必要

表二：RTSP方法一览，它们的方向，以及它们用来操作的对象（P：表示； S：流）

注意 关于表二：PAUSE是推荐的，但是不强制要求一个实现了完整功能的服务器支持此方法，例如，对于直播节目。如果一个服务器不支持一个特定的方法，它必须返回"501 未实现"，同时客户端不应该在此服务器上再次尝试此方法。

10.1 OPTIONS（可选项）

它的行为和[H9.2]所描述的相同。OPTIONS请求在任何时候都可能产生，例如：当一个客户端准备尝试一个非标准的请求时。它不影响服务器的状态。

例子:

```
C->S:  OPTIONS * RTSP/1.0
       CSeq: 1
       Require: implicit-play
       Proxy-Require: gzipped-messages

S->C:  RTSP/1.0 200 OK
       CSeq: 1
       Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE
```

注意：这些是有需要的虚构特性（有些人可能希望，我们不会有意忽略一个确实有用的特性，以便在这章得到有力的例子）。

10.2 描述（DESCRIBE）

DESCRIBE方法从服务器取得请求URL所标识的表示或者媒体对象的描述。它可能使用同意头部（Accept header）来指出客户端能理解的描述格式。服务器以所请求的资源的描述作为回应。DESCRIBE 回复-响应对继续了RTSP的媒体初始化阶段。

例子：

C->S: DESCRIBE rtsp://server.example.com/fizzle/foo RTSP/1.0
CSeq: 312
Accept: application/sdp, application/rtsl, application/mheg

S->C: RTSP/1.0 200 OK
CSeq: 312
Date: 23 Jan 1997 15:35:06 GMT
Content-Type: application/sdp
Content-Length: 376

v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 3456 RTP/AVP 0
m=video 2232 RTP/AVP 31
m=whiteboard 32416 UDP WB
a=orient:portrait

DESCRIBE响应必须包含它所描述的资源的所有媒体初始化信息。如果媒体客户端通过DESCRIBE以外的途径从一个源得到了一个表示描述，并且此描述包含了媒体初始化参数的完整集合，那么客户端应该使用那些参数，并不再通过RTSP为同一个媒体请求描述。

另外，服务器不应该把DESCRIBE响应当作媒体重定向的手段。

为了使客户端清楚无误地知道什么时候通过DESCRIBE去请求媒体初始化信息，什么时候不需要，需要建立明确的章程。通过强制要求DESCRIBE响应包含媒体初始化包含它要描述的流集合的所有媒体初始化信息，以及不推荐的媒体间接DESCRIBE用法，我们避免了其他途径可能带来的死循环问题。

媒体初始化是所有基于RTSP的系统的要求，但是RTSP规范没有规定这必须通过DESCRIBE方法来实现。RTSP客户端可以有三种途径来得到初始化信息：

*通过RTSP的DESCRIBE方法；

*通过其他协议（HTTP、email 附件，等等）；

*通过命令行或标准输入（从而像浏览器辅助程序以SDP文件或其他媒体初始化格式启动一样工作 ）。

为方便实际交互，强烈建议在最小服务 中包括DESCRIBE方法，同时强烈建议最小客户端 支持"辅助程序"这样的功能来从标准输入、命令行，及其他适合客户端工作环境的途径，来接受媒体初始化文件。

10.3 通知（ANNOUNCE）

ANNOUNCE方法有两个目的：

当从客户端发往服务器端，ANNOUNCE向服务器端上传请求URL所标识的表示或媒体对象的描述。当从服务器端发往客户端，ANNOUNCE实时更新会话描述。

当一个新的媒体流加入一个表示（例如：在一个现场表示活动期间）时，整个表示而不仅是所增加的部分，应该被重发，以便部分删除。

例子：

C->S: ANNOUNCE rtsp://server.example.com/fizzle/foo RTSP/1.0
CSeq: 312
Date: 23 Jan 1997 15:35:06 GMT
Session: 47112344
Content-Type: application/sdp
Content-Length: 332
v=0
o=mhandley 2890844526 2890845468 IN IP4 126.16.64.4
s=SDP Seminar

i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 3456 RTP/AVP 0
m=video 2232 RTP/AVP 31
S->C: RTSP/1.0 200 OK
CSeq: 312

10.4 建立（SETUP）

针对一个URI的SETUP详细说明了将要用于流媒体的传输机制。客户端可以针对已开始播放的流发出SETUP请求，来改变传输参数--服务器可能会同意。如果它不同意，它必须响应一个"455 此状态下此方法无效"错误。为便于穿透防火墙，客户端必须指示传输参数，即便它不能影响这些参数，例如：服务器向哪里放固定的多播地址的广告。

由于SETUP包含了所有的传输初始化信息，防火墙和其他中间网络设备（那些需要这些信息的）就不用去解析相对费力些的DESCRIBE响应--DESCRIBE响应是为媒体初始化预留的。

传输头部（Transport header）详细列出了客户端能接受的数据传输参数； 响应中会包含服务器选定的传输参数。

C->S: SETUP rtsp://example.com/foo/bar/baz.rm RTSP/1.0
CSeq: 302
Transport: RTP/AVP;unicast;client_port=4588-4589
S->C: RTSP/1.0 200 OK
CSeq: 302
Date: 23 Jan 1997 15:35:06 GMT
Session: 47112344
Transport: RTP/AVP;unicast;
client_port=4588-4589;server_port=6256-6257

服务器在响应SETUP请求时生成会话标识。如果发往服务器的SETUP请求中包含了会话标识，服务器必须把这个SETUP请求绑定到已存在的会话中，或是返回"459 不允许合控制"错误。

10.5 播放（PLAY）

PLAY方法告诉服务器通过SETUP规定的机制开始传输数据。客户端【必须不】在SETUP请求被明确确认为成功以前发送PLAY请求。

PLAY请求通过给出相对于正常播放时间的开始时间和结束时间，来给出播放范围。从开始时间开始传输流数据直到到达结束时间。PLAY请求可以用管道（队列）的形式；服务器【必须】按顺序执行收到的PLAY请求。即：在前一个PLAY请求依然活动期间收到的PLAY请求，要延迟到前一个PLAY请求执行完毕后才开始生效。

这允许了精确控制。

例如：不管下面例子中的两个PLAY请求的到达间隔多短，服务器将先从第10秒播放到第15秒，然后马上继以20到25秒，再从第30秒放到结束。

C->S: PLAY rtsp://audio.example.com/audio RTSP/1.0
CSeq: 835
Session: 12345678
Range: npt=10-15
C->S: PLAY rtsp://audio.example.com/audio RTSP/1.0
CSeq: 836
Session: 12345678
Range: npt=20-25
C->S: PLAY rtsp://audio.example.com/audio RTSP/1.0
CSeq: 837
Session: 12345678
Range: npt=30-

更多例子请见关于PAUSE的叙述。

没有范围头部（Range header）的PLAY请求是合法的。这表示从流的开始处播放，除非流被暂停。如果流被PAUSE暂停，继续流传输时从暂停点开始。如果流正在播放，这种PLAY请求不会引起任何动作，客户端可以用它来测试服务器是否在直播。

范围头部也可能包含时间参数。这个参数给出回放开始的UTC时间 。如果消息是在所指定的时间之后收到的，回放就立即从当前开始。时间参数可以被用于帮助同步从不同源得来的流。

对于一个按需点播的流，服务器以将实际回放的范围回应。如果媒体源要求把所请求的范围转换为有效的帧范围，这将有可能和所请求的范围不同 。如果请求中没有指定范围，在回应中返回当前位置。回应中范围的单位和请求一致。

播放完所需范围以后，表示将自动暂停，如同收到了PAUSE请求一样。

下面的例子从表示的0:10:20（SMPTE时间码）处开始播放，直到剪辑结束。回放动作从1997年1月23日15:36开始。

C->S: PLAY rtsp://audio.example.com/twister.en RTSP/1.0

CSeq: 833

Session: 12345678

Range: smpte=0:10:20-;time=19970123T153600Z

S->C: RTSP/1.0 200 OK

CSeq: 833

Date: 23 Jan 1997 15:35:06 GMT

Range: smpte=0:10:22-;time=19970123T153600Z

为回放一个现场表示的录像，使用时钟单位比较理想：

C->S: PLAY rtsp://audio.example.com/meeting.en RTSP/1.0

CSeq: 835

Session: 12345678

Range: clock=19961108T142300Z-19961108T143520Z

S->C: RTSP/1.0 200 OK

CSeq: 835

Date: 23 Jan 1997 15:35:06 GMT

只支持回放的服务器【必须】支持npt时间格式，【可能】支持时钟格式、smpte格式。

10.6 暂停（PAUSE）

PAUSE请求使得流传输被临时暂停（中断）。如果请求URL指向一个流，仅该流的回放和录制会被中断。例如：对于音频，这相当于静音 。如果请求URL指向一个表示或者一组流，该表示或该组流中所有正在活动的流的传输都被中断。继续回放或录制后，各个多媒体轨【必须】进行同步。所有服务器资源被保留--尽管服务器【可能】会根据SETUP消息中会话头部（Session header）里的超时（timeout）参数，在暂停一段时间后关闭会话并释放资源。

例子：

C->S: PAUSE rtsp://example.com/fizzle/foo RTSP/1.0

CSeq: 834

Session: 12345678

S->C: RTSP/1.0 200 OK

CSeq: 834

Date: 23 Jan 1997 15:35:06 GMT

PAUSE请求可能包含一个范围头部来指示从什么时候起开始中断流或者表示。我们把这个点叫做"暂停点"。该头部包含的必须是一个精确的值而不是一个范围。流的正常播放时间被设为暂停点。服务器遇到第一个包含此暂停点的当前未决的PLAY请求时，pause请求 生效。如果如果范围头部指示的时间范围在所有当前未决的PLAY请求的范围之外，返回一个"457 无效范围"错误。如果一个媒体单元（如音频或视频的一个帧）正好从暂停点开始表示，将不被播放或录制。如果没有范围头部，则流传输从收到该消息的第一时间起暂停，暂停点被设置为当前的正常播放时间。

PAUSE请求丢弃所有列队等候中的PLAY请求。但是【必须】维护媒体流的暂停点。下一个没有范围头部的PLAY请求将从该暂停点继续播放。

例如：如果服务器有范围从10到15和从20到29的两个未决play请求，之后收到一个要求在NTP 21处暂停的pause请求，它将开始播放第二个范围并在NPT 21处停止。如果pause请求是在NPT 12处暂停，而服务器正在把第一个请求播放到NPT 13处，服务器马上暂停。如果pause请求在NPT 16处暂停，服务器将在播放完第一个play请求后停止并丢弃第二个play请求。

另一个例子：如果服务器收到了要求先播放范围10到15，再播放13到20（即范围有重叠）的两个请求，位于NPT=14的PAUSE请求将在服务器播放第一个范围时起作用，并使第二个PLAY请求被忽略--假设PAUSE请求在服务器开始播放第二个重叠的范围之前到达。不管PAUSE请求何时到达，它使NPT被设置为14。

如果服务器已经开始发送范围头部所指示时间之后的数据，PLAY仍然从该时间点开始继续，就如假设客户端抛弃了那些处于时间点之后的数据一样。这保证连续的 暂停/播放 循环间没有间隙。

10.7 断开（TEARDOWN）

TEARDOWN请求会停止所给URI的流传输，释放与它相关的资源。如果所给的URI是这个表示的表示URI，任何与此会话相关的会话标识都将不再有效。除非所有传输参数都在会话描述中定义了，否则在再次播放之前必须发送SETUP请求。

例子：

C->S: TEARDOWN rtsp://example.com/fizzle/foo RTSP/1.0

CSeq: 892

Session: 12345678

S->C: RTSP/1.0 200 OK

CSeq: 892

10.8 获取参数（GET_PARAMETER）

GET_PARAMETER请求用于获取URI所标识的一个表示或者流的参数的值。回复和响应的内容被留给具体实现。没有实体的GET_PARAMETER可能被用于测试客户端或者服务器端是否现场直播（"ping"）。

例子：

S->C: GET_PARAMETER rtsp://example.com/fizzle/foo RTSP/1.0

CSeq: 431

Content-Type: text/parameters

Session: 12345678

Content-Length: 15

packets_received

jitter

C->S: RTSP/1.0 200 OK

CSeq: 431

Content-Length: 46

Content-Type: text/parameters

packets_received: 10

jitter: 0.3838

"text/parameters"只是一个用于示范的参数类型。这个方法被有意地定义得比较松散，意在通过将来的实践来定义回应内容和响应内容。

10.9 设置参数（SET_PARAMETER）

该方法用来请求设置URI对应的表示或者流的参数的值。

一个请求【应该】只包含一个参数，以便客户端知道为什么一个特定的请求会失败。如果请求包含多个参数，服务器【必须】仅在所有参数都能成功设置时才去做所请求的动作 。服务器【必须】允许同一个参数被重复设定为同一个值，但是它【可能】不同意改变参数的值。

注意：媒体流的传输参数【必须】只通过SETUP命令来设置。

把传输参数的设定严格限制在SETUP是为了利于通过防火墙。

参数被划分为细粒度的形式，以便可给出更有意义的错误说明。 尽管如此，当需要原子设置操作 时，允许一次设置多个参数还是有意义的。想象一下这里的设备控制：除非摄像头能实时调整到合适的角度，否则客户不想让摄像头拍特写。

例子：

C->S: SET_PARAMETER rtsp://example.com/fizzle/foo RTSP/1.0

CSeq: 421

Content-length: 20

Content-type: text/parameters

barparam: barstuff

S->C: RTSP/1.0 451 Invalid Parameter

CSeq: 421

Content-length: 10

Content-type: text/parameters

barparam

"text/parameters"只是一个用于示范的参数类型。这个方法被有意地定义得比较松散，意在通过将来的实践来定义回应内容和响应内容。

10.10 重定向（REDIRECT）

REDIRECT请求用于通知客户端：必须连接到另一个服务器地址。它包含一个强制的地址（Location）头部，该头部指示客户端该向哪个URL发送请求。它可能包含范围（Range）参数，该参数指示重定向何时生效。如果客户端想要转向所指示的URI以继续发送或接受媒体，客户端【必须】向当前会话发送一个TEARDOWNQ请求，并为新会话向所指示的主机发送一个SETUP请求。

下面这个请求的例子，把传输在给定播放时间 重定向到新服务器的下述URI:

S->C: REDIRECT rtsp://example.com/fizzle/foo RTSP/1.0

CSeq: 732

Location: rtsp://bigserver.com:8001

Range: clock=19960213T143205Z-

10.11 录制（RECORD）

该方法根据表示描述，开始录制一个范围内的媒体数据。时间戳反映开始和结束时间 （UTC）.如果没给出时间范围，就使用表示描述提供的开始及结束时间。如果会话已经开始，则立即开始录制。

服务器决定是在请求-URI还是其他URI来录制数据。如果服务器不使用请求-URI，响应【应该】是201（已建立），并包含一个实体-该实体描述了请求的状态并给出新资源，还包含一个位置头部（Location header）。

一个支持录制现场表示的服务器【必须】支持时钟范围（clock range）格式；smpte格式是没有意义的。

在此例子中，媒体服务器在之前被邀请进所指示的会议中。

C->S: RECORD rtsp://example.com/meeting/audio.en RTSP/1.0

CSeq: 954

Session: 12345678

Conference: 128.16.64.19/32492374

10.12 嵌入（交织）的二进制数据

一些防火墙设计和其他环境可能会迫使服务器交织插入RTSP方法和流数据。除非必要，否则应该避免这种交织，因为它使客户端和服务器的操作复杂化并强加了额外的负担。交织的二进制数据【应该】只在RTSP通过TCP承载时使用。

像RTP包这样的流数据被封装为这样的形式：以ASCII码的"$"符号（0x24）为封装标志，后面是一个单字节信道标识符，然后紧跟一个网络字节序的双字节二进制整数值，该整数表示所封装二进制数据的长度。再后面紧跟着没有CRLF，但包含上层协议头部的流数据。每个 $ 块包含且仅包含一个上层协议数据单元，例如，一个RTP包。

信道标识符在传输头部（Transport header）通过参数交织（interleaved）定义(章节12.39)。

当选择RTP作为应用层传输协议，TCP连接上的RTCP消息也被服务器交织处理。默认把RTCP包放在比RTP信道高的第一个可用信道上。客户端【可能】会显式地在另一信道上请求RTCP包。这通过在传输头部（Transport header，章节12.39）的交织（interleaved）参数上给出两个信道来实现。

如果这种形式下有两个或更多流被交织，就需要RTCP。同时，在网络配置有要求时 ，这提供了用TCP控制连接来隧道式地传输RTP/RTCP包的便利方法，并在可能时用UDP传输它们。

C->S: SETUP rtsp://foo.com/bar.file RTSP/1.0

CSeq: 2

Transport: RTP/AVP/TCP;interleaved=0-1

S->C: RTSP/1.0 200 OK

CSeq: 2

Date: 05 Jun 1997 18:57:18 GMT

Transport: RTP/AVP/TCP;interleaved=0-1

Session: 12345678

C->S: PLAY rtsp://foo.com/bar.file RTSP/1.0

CSeq: 3

Session: 12345678

S->C: RTSP/1.0 200 OK

CSeq: 3

Session: 12345678

Date: 05 Jun 1997 18:59:15 GMT

RTP-Info: url=rtsp://foo.com/bar.file;

seq=232433;rtptime=972948234

S->C: $\000{2 byte length}{"length" bytes data, w/RTP header}

S->C: $\000{2 byte length}{"length" bytes data, w/RTP header}

S->C: $\001{2 byte length}{"length" bytes RTCP packet}

11 状态码定义

当HTTP状态码适用时，重用HTTP状态码。这里不再给出意义与HTTP相同的状态码。表一给出了哪个请求会返回哪个状态码的列表。

11.1 成功 2xx

11.1.1 250 存储空间不足

服务器在收到一个由于存储空间不足而无法完全满足的RECORD请求时，返回这个警告。如果可能，服务器应该使用范围（Range）头部来指示仍然能够录制的时间范围。因为服务器的其他进程可能会同时消耗存储空间，客户端应该仅把这当作一个估计值。

11.2 重定向 3xx

见[H10.3].

在RTSP里，重定向被用于平衡附在或者把流请求重定向到拓扑上距离客户端更近的服务器。判断拓扑距离远近的机制不在本规范中给出。

11.3 客户端错误 4xx

11.3.1 405 不允许该方法

请求URI所指示的资源不允许使用当前方法所述的请求。响应【必须】包含一个允许（Allow）头部，该头部包含一个对于当前所请求的资源有效的方法的列表。该当请求试图使用在SETUP期间没有列出的方法时，也可以用此状态码，例如：虽然传输（Transport）头部的模式（mode）参数只列出了PLAY方法，但却发送了一个RECORD请求。

11.3.2 451 不理解此参数

请求的接收方不支持请求所包含的一个或多个参数。

11.3.3 452 找不到会议

媒体服务器找不到会议（Conference）头部域所指示的会议

11.3.4 453 带宽不足

由于带宽不足，请求被拒绝。这可能，例如，是由于资源预约失败。

11.3.5 454 找不到

会话（Session）头部中的RTSP会话标识符丢失、失效，或已超时。

11.3.6 455 此状态下此方法无效

客户端或服务器端无法在它的当前状态下处理此请求。该响应【应该】包含一个允许（Allow）头部以方便修正错误。

11.3.7 456 此头部域对该资源无效

服务器无法对请求头部域做出反应。例如，当一个PLAY请求包含范围（Range）头部，但流不允许搜索。

11.3.8 457 无效范围

给出的范围值越界，例如，超过了表示的末端。

11.3.9 458 参数是只读的

SET_PARAMETER想要设置的参数只能读取不能修改。

11.3.10 459 不允许合控制

由于是一个合（表示）URL，所请求的方法不会被应用在该URL上。该方法可能可以用在单个的流URL上。

11.3.11 460 只允许合控制

由于不是一个合（表示）URL,所请求的方法不会被应用在该URL上。该方法可能可以用在表示URL上。

11.3.12 461 传输方法不支持

传输（Transport）域不包含能够支持的传输协议。

11.3.13 462 无法到达目的地址

由于无法到达客户端地址，无法建立传输通道。最可能的产生此错误的情况是，客户端试图把无效的目的地址参数放到传输（Transport）域。

11.3.14 551 不支持该选项

不支持需求（Require）或代理需求（Proxy-Require）域所给的某个选项。应当返回不被支持的头部，列出不支持的选项。

12 头部域定义

HTTP/1.1 [2]以及其他，现未列于此处的应该被接收方忽略的没有确切定义的非标准头部域。

表三总结了RTSP使用的头部域。"g"型别表示是请求和响应通用的请求头部；"R"型别表示是请求头部；"r"型别表示是响应头部；"e"表示是实体头部域。列栏中标记为"req."的域【必须】被接收方针对某些特定的方法实现出来；而标记为"opt."是可选的。注意：不是所有的标记为"req."的域都得在每一个该类请求中发送。"req."意味着仅只客户端（针对响应头部）和服务器端（针对请求头部）【必须】实现此头部。最后一列列出了头部域对哪些方法有意义；所有返回消息主体（message body）的方法都用"entity"标识。在本规范中，DESCRIBE 和GET_PARAMETER处于此列。

头部	型 别	支 持	方 法
Accept	请求	可选.	entity
Accept-Encoding	请求	可选	entity
Accept-Language	请求	可选	all
Allow	响应	可选	all
Authorization	请求	可选	all
Bandwidth	请求	可选	all
Blocksize	请求	可选	all除了OPTIONS, TEARDOWN
Cache-Control	通用	可选	SETUP
Conference	请求	可选	SETUP
Connection	通用	必须	all
Content-Base	实体	可选	entity
Content-Encoding	实体	必须	SET_PARAMETER
Content-Encoding	实体	必须	DESCRIBE, ANNOUNCE
Content-Language	实体	必须	DESCRIBE, ANNOUNCE
Content-Length	实体	必须	SET_PARAMETER, ANNOUNCE
Content-Length	实体	必须	entity
Content-Location	实体	可选	entity
Content-Type	实体	必须	SET_PARAMETER, ANNOUNCE
Content-Type	响应	必须	entity
CSeq	通用	必须	all
Date	通用	可选	all
Expires	实体	可选	DESCRIBE, ANNOUNCE
From	请求	可选	all
If-Modified-Since	请求	可选	DESCRIBE, SETUP
Last-Modified	实体	可选	entity
Proxy-Authenticate			
Proxy-Require	请求	必须	all
Public	响应	可选	all
Range	请求	可选	PLAY, PAUSE, RECORD
Range	响应	可选	PLAY, PAUSE, RECORD
Referer	请求	可选	all
Require	请求	必须	all

Retry-After 响应 可选 all
RTP-Info 响应 必须 PLAY
Scale 请求响应 可选 PLAY, RECORD
Session 请求响应 必须 all除了SETUP, OPTIONS
Server 响应 可选 all
Speed 请求响应 可选 PLAY
Transport 请求响应 必须 SETUP
Unsupported 响应 必须 all
User-Agent 请求 可选 all
Via 通用 可选 all
WWW-Authenticate 响应 可选 all

RTSP头部域一览

12.1 接受（Accept）

接受（Accept）请求-头部域可以用来描述确切的响应可接受的表示描述的内容类型。

表述描述的等级（level）参数被适当地定义为MIME类型注册中的一部分，，而不是在这里。

见[H14.1]中的语法。

例子：

Accept: application/rtsl, application/sdp;level=2

12.2 接受-编码（Accept-Encoding）

见[H14.3]

12.3 接受-语言（Accept-Language）

见[H14.4]。注意所描述的语言是应用于描述表述和原因短语，而不是媒体内容。

12.4 允许（Allow）

允许（Allow）响应头部域列举出请求-URI所指向的资源的所有支持的方法。该域的目的在于通知接收方，具体哪些方法对该资源有效。允许（Allow）头部域必须在405（不允许该方法）中出现。

用例：

Allow: SETUP, PLAY, RECORD, SET_PARAMETER

12.5 授权（Authorization）

见[H14.8]

12.6 带宽（Bandwidth）

带宽（Bandwidth）请求头部域用于描述可分配给客户端的带宽估计值，以单位为bit/秒的正整数表示。在RTSP会话期间，客户端可用带宽可能会改变，例如，由于modem调整。

Bandwidth = "Bandwidth" ":" 1*DIGIT

例子：

Bandwidth: 4000

12.7 块大小（Blocksize）

该请求头部域从客户端发向媒体服务器，以询问服务器某种媒体包的大小。这里的包大小不包括低层包头，如IP, UDP, 或RTP。服务器可以随意使用比所要求的包大小更小的包。服务器【可能】会以最接近此块大小的媒体指定的包大小的整数倍作为实际块大小，在必要情况下也可能忽略此包大小而使用媒体指定的大小。块大小【必须】是一个正十进制数字，以字节为单位。如果值的语法不正确，服务器只返回416错误。

12.8 缓存控制（Cache-Control）

缓存控制（Cache-Control）通用头部域用于给出所有缓存机制在请求/响应连接路径上【必须】遵循的指令。

因为缓存指令可能对请求/响应连接路径沿线的所有接收方都有用，所以缓存指令必须能通过代理或网关程序，不管它们对这些程序是否有意义。不太可能为一个具体的缓存指定缓存指令。

缓存控制（Cache-Control）应该只在SETUP请求和它的响应中给出。注意：缓存控制不像HTTP那样管理着响应缓存，而是管理着SETUP请求所标识的流。除了DESCRIBE的响应外，RTSP请求的响应都是不缓存的。

```
Cache-Control          =  "Cache-Control" ":" 1#cache-directive
cache-directive        =  cache-request-directive
                       |  cache-response-directive
cache-request-directive =  "no-cache"
                       |  "max-stale"
                       |  "min-fresh"
                       |  "only-if-cached"
                       |  cache-extension
cache-response-directive =  "public"
                       |  "private"
                       |  "no-cache"
                       |  "no-transform"
```

| "must-revalidate"
| "proxy-revalidate"
| "max-age" "=" delta-seconds
| cache-extension

cache-extension = token ["=" (token | quoted-string)]

不缓存（no-cache）：

指示媒体流【必须不】在任何地方缓存。这允许原始服务器阻止缓存动作，哪怕是配置为需要向客户端返回“陈旧”（stale）响应的缓存。

公有（public）：

指示任何缓存都可缓存此媒体流。

私有（private）：

指示媒体流只供某单个用户使用，共享缓存【必须不】缓存它。私有（非共享的）缓存可能会缓存该媒体流。

非传输（no-transform）：

可用于中间缓存（代理）转换某种流的媒体类型。例如，代理可以转换视频格式以节省缓存空间，或减少慢速链接之间的传输。但当这种转换被应用在特殊程序所用的流上时，可能会导致严重的运行问题。例如，医学图像的传输程序，科学数据分析程序，以及使用端对端鉴权 ，它们全依赖于接收到的流和原始实体的比特数据一一对应。因此，如果响应当中包含no-transform指令，中间缓冲或代理【必须不】改变流的编码。和HTTP不同，RTSP在这一点上不提供局部转换，例如，允许翻译成另一种语言。

仅若已缓存（only-if-cached）：

某些情况下，如在延时非常严重的网络环境中，客户端可能想让缓存只给出刚刚所缓存的媒体流，而不是从原始服务器接收它们。为此客户端可能会在请求中包含only-if-cached指令。如果它接收到了该指令，缓存【应该】要么用已缓存的符合请求中其他要求的媒体流作为应答，要么响应一个504（网关超时）状态。然而，如果一组缓存像有很好内部连通性的统一系统那样被操控着，该请求【可能】会在这组缓存内发送。

最大-陈旧（max-stale）：

指示客户端愿意接收已经超过了有效生命期 （freshness lifetime）的媒体流。如果max-stale被赋值，那么该客户端愿意接收超过了截止有效期（expiration time ），但超出值不超过该值所标示秒数的响应。如果max-stale没有赋值，则客户端愿意接收任意年龄（age）的陈旧响应。

最小-新鲜（min-fresh）：

指示客户端愿意接收有效生命期（freshness lifetime）不少于当前年龄加上所给出秒数的媒体流。即客户端想要响应在未来的至少所给的的秒数内仍然有效。

必须-重验证（must-revalidate）：

当SETUP响应中出现的must-revalidate指令被缓存接收后，缓存【必须不】在向原始服务器重新验证它前，使用过期的条目来响应下一个请求。即，如果仅基于原始服务器的过期时间，缓存的响应是过期的，则缓存必须每次都进行端对端的重验证。

12.9 会议（Conference）

该请求头部域建立了一个预建立的会议到RTSP流之间的逻辑连接。对于同一个RTSP会话，conference-id必须不能改变。

Conference = "Conference" ":" conference-id Example:

Conference: 199702170042.SAA08642@obiwan.arl.wustl.edu%20Starr

若conference-id无效，返回一个452（452 找不到会议）响应码。

12.10 连接（Connection）

见[H14.10]

12.11 内容-基础（Content-Base）

见[H14.11]

12.12 内容-编码（Content-Encoding）

见[H14.12]

12.13 内容-语言（Content-Language）

见[H14.13]

12.14 内容-长度（Content-Length）

该域包含方法的内容的长度（即是说，从最后一个头部后面的两个CRLF之后）。和HTTP不同的是，它【必须】包含在所有头部后有内容（content）的消息内。如果它丢失了，则假设其值为0。根据[H14.14]给出解释。

12.15 内容-位置（Content-Location）

见[H14.15]

12.16 内容-类型（Content-Type）

见[H14.18]。注意适用于RTSP的Content-Type倾向于限制在表示描述的惯例和parameter-value类型范围内。

12.17 应答序列号（CSeq）

CSeq域指示了RTSP请求-响应对的序列号。该域【必须】出现在所有的请求和响应中。对于每一个包含了序列号的RTSP请求，它对应的响应会有相同的序列号。任何重传的请求必须包含和原来一样的序列号（即是说，重传同一个请求时，序列号不增加）。

12.18 日期（Data）

见[H14.19]

12.19 过期（Expires）

Expires实体-头部域给出了一个日期和时间，超过该时间的描述或者媒体流就不再被视为有效的。具体涵义视方法而定：

DESCRIBE响应：

Expires头部指明了描述的最后有效日期和时间。

缓存（包括代理缓存和用户程序缓存）一般不会返回过期的条目，除非先向原始服务器（或者缓存有该条目的有效拷贝的中间服务器）验证过。关于过期模式的进一步解释见章节13.

出现Expires域并不暗示原始资源会在所述时间内，或之前、之后改变或被删除。

该格式是一种绝对时间，和[H3.3]中HTTP-date的相同，它【必须】是RFC1123-date格式：

Expires = "Expires" ":" HTTP-date

用例：

Expires: Thu, 01 Dec 1994 16:00:00 GMT

RTSP/1.0的客户端和缓存【必须】把其他无效的数据格式尤其是"0"值，当作是过去产生的（也就是“已经过期的”）。

要把一个响应标记为“已经过期”，原始服务器应该使用等于时间头部值的过期时间（Expires date）。要把响应标记为“永不过期”，则原始服务器应该用从响应发送时起大约一年后的时间作为过期时间。RTSP/1.0服务器不应该发送超过当前时间一年之久的过期时间。

除非另有缓存-控制（Cache-Control）头部域指示，否则对于一个默认是不可缓存的媒体流，如果出现了数据值为将来某个时间的Expires头部域，则为指示该媒体流是可缓存的。

12.20 来自（From）

见[H14.22].

12.21 主机（Host）

RTSP不需要该HTTP请求头部域。如果发送了该头部，接收方应该无动作地直接忽略它。

12.22 如果-符合(If-Match)

见[H14.25]。

无论是通过RTSP以外的途径（例如HTTP）取得表示描述，还是服务器实现需要保证从DESCRIBE消息到SETUP消息时间段内描述（description）的完整性，该域在保障表示描述的完整性方面都尤其实用。

由于标识符是种不透明的标识符，因此并不局限于某种特定的会话描述语言。

12.23 如果-被修改-自从（If-Modified-Since）

If-Modified-Since请求头部域是用来和DESCRIBE和SETUP方法一起来形成条件判断。如果被请求的变量从该域所指定的时间以来没有被修改过，则服务器不会返回描述（DESCRIBE） 或 流不会被建立（SETUP）。作为代替，会返回一个没有任何消息体的304（没有修改）响应。

If-Modified-Since = "If-Modified-Since" ":" HTTP-date

该域的一个例子：

If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT

12.24 最后-修改（Last-Modified）

Last-Modified头部域指示原始服务器认为表示描述或媒体流最后被修改的日期和时间。见[H14.29]。对于DESCRIBE或ANNOUNCE方法，该头部域指示最后修改描述的日期和时间，对于SETUP则是修改媒体流的日期和时间。

12.25 位置（Location）

见[H14.30]。

12.26 代理-鉴权（Proxy-Authenticate）

见[H14.33]。

12.27 代理-要求（Proxy-Require）

Proxy-Require头部是用于指出代理【必须】支持的 代理-敏感 的特性。任何代理不支持的Proxy-Require头部特性【必须】由代理向客户端作出否定确认，如果它不支持的话。服务器应该和处理Require域一样地处理该域。

更多该消息的机制和用例的细节见12.32节。

12.28 公布（Public）

见[H14.35]。

12.29 范围（Range）

该请求及响应头部域定义了一个时间范围。该范围可用多种单位给出。该规范定义了smpte (3.5节), npt (3.6节), 和 clock (3.7节) 范围单位。在RTSP中，字节范围[H14.36.1] 是没有意义的且【必须不】使用。头部也可能包含UTC形式的时间参数，指出操作将在什么时候启动。支持范围头部的服务器【必须】理解NPT范围格式，且【应该】理解SMPTE范围格式。范围响应头部指出实际播放或录制的时间范围。如果范围头部用无法理解的时间格式给出，接收方应该返回"501 未实现"。

范围是个半开区间，包括下边界点，但不包括上边界点。换句话说，一个a-b范围精确地从时间a开始，但是b之前停止。只有如视频或音频帧这样的媒体单元的开始时间是相关的 。例如，假设视频是每40毫秒生成一帧。范围10.0-10.1将包括从10.0或稍后开始的视频帧，也会包括10.08开始的视频帧--尽管它的持续期跨过了范围区间。换句话说，范围10.0-10.08，将把10.08的帧排除在外。

```
Range          = "Range" ":" 1\#ranges-说明符
                 [ ";" "time" "=" utc-time ]
```

ranges-说明符 = npt-range | utc-range | smpte-range

示例：

Range: clock=19960213T143205Z-;time=19970123T143720Z

该符号的使用类似于HTTP/1.1 [2] byte-range头部。这允许客户端从媒体对象中选取剪辑，和从给定时间点播放到末尾，以及从当前位置播放到给定时间点。可以把开始回放的时间排到将来的任何时间，尽管服务器可能会拒绝为额外的空闲期保持服务器资源。

12.30 提交方（Referer）

见[H14.37]。指提及表示描述的URL，典型地是从HTTP中取得。

12.31 重试-自从（Retry-After）

见[H14.38]

12.32 要求（Require）

要求头部被客户端用于向服务器询问其能够或不能够支持的选项。服务器【必须】使用不支持（Unsupported）头部来响应该头部，以对不支持的选项作出否定确认。

这意在确保客户端-服务器交互将在所有选项头能被双方理解的情况下无延迟地进行，并仅在不能理解选项时（如上文情况）降低速度。对于一对配合得很好的客户端-服务器，交互将进行得很快，节约了协商机制经常需要的来回时间。另外，这也消除了当客户端需要服务器无法理解的特性时产生的不明确性。

Require = "Require" ":" 1#选项-标签

示例:

C->S: SETUP rtsp://server.com/foo/bar/baz.rm RTSP/1.0

CSeq: 302

Require: funky-feature

Funky-Parameter: funkystuff

S->C: RTSP/1.0 551 Option not supported

CSeq: 302

Unsupported: funky-feature

C->S: SETUP rtsp://server.com/foo/bar/baz.rm RTSP/1.0

CSeq: 303

S->C: RTSP/1.0 200 OK

CSeq: 303

在该示例中，"funky-feature"是向客户端指示需要虚构的Funky-Parameter域的特性标签。"funky-feature"和Funky-parameter之间的关系不通过RTSP交互来沟通，因为此关系是"funky-feature"的不变的属性，因此不应该在每次RTSP交互中都传输。

代理和其他中间设备【应该】忽略该域中不理解的特性。如果有需要中间设备支持的特殊扩展，此扩展应该用 代理-要求（Proxy-Require）域（见12.27节）标签代替。

12.33 RTP-信息（RTP-info）

该域用于设定PLAY响应中的RTP相关参数。

url：

指示后面的RTP参数跟哪个流UTL相关联。

seq：

指示流的第一个包的序列号。这使得客户端在搜索 时方便地处理包。客户端使用该值来区分搜索位置前生成的包和搜索位置后生成的包。

rtptime：

指示和Range响应头部的时间值对应的RTP时间戳。（注意：对于合控制，某些特定的流可能在返回或暗含的Range时间内实际上不产生包。因此，无法保证用seq指示序列号的包实际上含有用rtptime指示的时间戳。）客户端用此值来实现从RTP时间到NPT的映射计算。

从RTP时间戳映射到NTP时间戳（时钟时间）可以借助RTCP。但是，该信息还不足以完成从RTP时间戳到NTP的映射 。此外，为了保证可以在必要的时候（启动后立即或搜索后）获得该信息，并且可靠地传输，该映射被置于RTSP控制信道内。

为了纠正较长的、不间断的表示的偏移，RTSP客户端还要额外地进行从NPT到NPT的映射，使用初始RTCP发送方报告来实现此映射，并用后续的报告来检查相对映射的偏移。

语法：

RTP-Info = "RTP-Info" ":" 1#stream-url 1*parameter

stream-url = "url" "=" url

parameter = ";" "seq" "=" 1*DIGIT

| ";" "rtptime" "=" 1*DIGIT

例子：

RTP-Info: url=rtsp://foo.com/bar.avi/streamid=0;seq=45102,

url=rtsp://foo.com/bar.avi/streamid=1;seq=30211

12.34 倍速（Scale）

为1的倍速值表示和正常播放速度一样的播放、录制速度。如果不是1，该值表示相对正常播放速度的比值。例如，比值2表示比正常播放速度快两倍（快进）；而比值0.5表示只有正常播放速度的一半。也就是说，比值2使正常播放时间的时钟频率增加了两倍。在时钟时间的每一秒内，放了两秒的内容。负值表示反方向。

除非Speed参数另有要求，否则数据速率【应该】不被改变。倍速改变的实现依赖于服务器和媒体类型。对于视频，服务器可能，例如，只传送关键帧或者所选的关键帧。对于音频，可能在保留完整音频的前提下调整时间倍速，或者作为次选，传输音频的小片段。

服务器应该试着估计播放速率，但可能把倍速值限制在自己支持的范围内。响应【必须】包含服务器实际选择的倍速值。

如果请求中包含Range参数，新的倍速值将影响此时间。

Scale = "Scale" ":" ["-"] 1*DIGIT ["." *DIGIT]

以相对正常的3.5倍的速度快退的例子：

Scale: -3.5

12.35 速度（Speed）

该请求头部要求服务器以特定的速度向客户端发送数据，根据服务器的能力和意愿去用给定的速度来提供媒体流。服务器端对它的实现是【可选的】。默认值是流本身的比特率。

该参数值以十进制小数的一个比率给出，例如，2.0说明数据被以正常的两倍的速度传输。Speed值为0是不允许的。如果请求中包含Range参数，则新的速度值将影响此时间。

Speed = "Speed" ":" 1*DIGIT ["." *DIGIT]

例子:

Speed: 2.5

该域的使用会改变数据传输所用的带宽。它可以用在特殊的环境中，如需要高速或低速预览表示时。开发者应该记住会话所用带宽的协商可能会预先进行（通过RTSP以外的途径），因此有必要重新协商。用UDP传输数据时，高度推荐使用如RTCP这样的方式来监视丢包率。

12.36 服务器（Server）

见[H14.39]

12.37 会话

该请求和响应头部域标识出一个根据表示URL，由媒体服务器的SETUP响应开始，由TEARDOWN终止的会话。会话标识由媒体服务器给出（见3.4节）。一旦客户端收到一个会话标识，它【必须】对每个与该会话关联的请求都返回该标识。如果服务器有其他能唯一标识出一个会话的途径，如动态产生的URL，它并不一定要建立一个会话标识。

Session = "Session" ":" session-id [";" "timeout" "=" delta-seconds]

超时（timeout）参数只允许出现在响应头部。服务器用它向客户端指示，服务器打算在由于缺少反馈信息（见章节A）而关闭会话前等RTSP命令等多久。timeout的单位是秒，默认值为60秒（1分钟）。

注意：会话标识把横跨传输会话或连接的RTSP会话标识出来。一个RTSP会话可能会发送对应多个RTSP URL的控制消息。因此，客户端可以用同一个会话控制一个表示中的多个流，只要这些流来自于同一个服务器。（见章节14的例子）。但是，同一个客户端对于同一个URL的多个“用户”【必须】使用不同的会话标识。

区分来自于同一个客户端针对同一个URL的不同传输请求时需要会话标识。

如果会话标识是非法的，则返回454响应（找不到会话）。

12.38 时间戳（Timestamp）

timestamp通用头部描述客户端何时向服务器发了请求。时间戳的值仅对客户端有意义且可使用任何时间间隔单位。服务器【必须】回显同一个值，且【可能】加上一些小数点后的数字来表示从收到请求后过了多少秒--如果它有相关精确信息的话。时间戳被客户端用以计算到服务器的来回时间，以便为重传调整超时时间。

Timestamp = "Timestamp" ":" *(DIGIT) ["." *(DIGIT)] [delay]

delay = *(DIGIT) ["." *(DIGIT)]

12.39 传输（Transport）

该请求头部指示要用哪个传输协议，并配置如目的地址、压缩、多播时的time-to-live和每个流的目的端口号这类参数。它设置那些表示描述没有给定的值。

传输是用逗号分隔的，按优先级排序。参数可能被加到每个传输上，之间用分号隔开。

传输头部【可能】也会被用于改变一些传输参数。服务器【可能】会拒绝改变已存在的流的参数。

服务器【可能】会返回一个传输响应头部以回应性地指出实际选取的参数。

传输请求头部域可能包含客户端可以接受的传输选项清单。此种情况下，服务器【必须】返回实际选定的某一个选项。

传输说明清单的语法是：

transport/profile/lower-transport.

低层传输参数的默认值是视上层而定的。对于RTP/AVT，默认是UDP.

下面是和transport相关的配置参数：

通用参数：

unicast（单播）| multicast（多播）：

二选一地指定是进行单播还是多播的传输尝试。默认值是多播。单播和多播都能处理的客户端【必须】通过包含两个传输的完整具体参数来指出这样的能力。

destination（目的地）：

流将被发往的地址。客户端用destination参数来给出多播地址。为了避免在不知情的情况下被用于拒绝服务攻击，服务器【应该】对客户端鉴权并【应该】在允许客户端把一个媒体流定向到非服务器选取的地址之前记录此类尝试。这对通过UDP发的RTSP命令来说尤其重要，但是实现不能把TCP当作可靠的客户端认证手段。服务器【应该】不允许客户端把媒体流定向到同命令来源不同的地址上去。

source（来源）：

如果流的源地址不同于可从RTSP末端点地址 （回放中的服务器或录制中的客户端）得到的，【可能】会给出source。

该信息也可以通过SDP得到。但是，因为这更多的是一项传输特性而不是媒体初始化特性，该信息的权威性的source应该放在SETUP响应中。

layers（层）：

将要用于该媒体流的多播的层数。layers是发送到以目的地址起始的连续地址。

mode（模式）：

mode参数指示该会话支持的方法。有效的值有PLAY和RECORD。如果没有提供，默认值是PLAY。

append（追加）：

如果mode参数中包括RECORD，则append参数指示媒体数据应该追加到已有的资源上而不是覆盖它。如果要求了追加而服务器不支持，它【必须】拒绝该请求而不是覆盖URI所标识的资源。如果mode参数中不包含RECORD，则append被忽略。

interleaved（交织）：

interleaved参数意味着不管控制流使用何种协议，都把媒体流和控制流混合在一起，使用10.12节定义的机制。用 $ 语法提供表示信道号的参数。该参数可能会以一个范围的形式提供，例如：interleaved=4-5 在这里表示向媒体流提供的传输选择。

这允许用类似UDP的方式来处理RTP/RTCP，例如，一个信道给RTP,另一个给RTCP。

Multicast specific（多播细节）：

ttl：

多播的time-to-live

RTP Specific（RTP细节）：

port（端口号）：

该参数为多播会话提供RTP/RTCP端口号对。它用范围的形式给出，例如，port=3456-3457。

client_port(客户端端口)：

该参数提供客户端选择的接收媒体数据和控制信息的单播RTP/RTCP端口号对。它用范围的形式给出，例如，client_port=3456-3457。

server_port(服务器端口)：

该参数提供服务器选择的用来接收媒体数据和 控制信息的单播RTP/RTCP端口号对。它用范围的形式给出，例如，server_port=3456-3457。

ssrc:

ssrc参数指示媒体服务器应该用的（请求）或将要用的（响应）RTP SSRC[24,章节3]值.该参数只对单播传输有效。它是和媒体流关联的同步源的标识。

```
 Transport          =   "Transport" ":"
                        1\#transport-spec
 transport-spec     =   transport-protocol/profile[/lower-transport]
                        *parameter
 transport-protocol =   "RTP"
 profile            =   "AVP"
 lower-transport    =   "TCP" | "UDP"
 parameter          =   ( "unicast" | "multicast" )
                    |   ";" "destination" [ "=" address ]
                    |   ";" "interleaved" "=" channel [ "-" channel ]
                    |   ";" "append"
                    |   ";" "ttl" "=" ttl
                    |   ";" "layers" "=" 1*DIGIT
                    |   ";" "port" "=" port [ "-" port ]
                    |   ";" "client_port" "=" port [ "-" port ]
                    |   ";" "server_port" "=" port [ "-" port ]
                    |   ";" "ssrc" "=" ssrc
                    |   ";" "mode" = <"> 1\#mode <">
 ttl                =   1*3(DIGIT)
 port               =   1*5(DIGIT)
 ssrc               =   8*8(HEX)
```

channel = 1*3(DIGIT)
address = host
mode = <"> *Method <"> | Method

例子:

Transport: RTP/AVP;multicast;ttl=127;mode="PLAY",
RTP/AVP;unicast;client_port=3456-3457;mode="PLAY"

Transport头部只能描述单个RTP流。（RTSP也可以把多个流当一个实体进行控制。）把它作为RTSP的一部分而不是依赖它作为最通用的会话描述格式，极大地简化了防火墙的设计。

12.40 Unsupported 不支持的

Unsupported响应头部列出服务器不支持的特性。在特性是通过Proxy-Require域给出的情况下（12.32节），代理【必须】把出错消息“551 选项不支持”插入回应消息。

见12.32节的用例。

12.41 User-Agent（用户-代理）

见[H14.42]。

12.42 Vary（变化）

见 [H14.43]

12.43 Via（通过）

见 [H14.44].

12.44 WWW-Authentica（www-鉴权）

见 [H14.46].

13 Caching

在HTTP中，响应-请求对是被缓存了的。RTSP在这方面有显著的不同。响应不可以缓存，除了DESCRIBE返回的或ANNOUNCE中包含的表示描述以外。（因为除了DESCRIBE和GET_PARAMETER以外任何响应不返回任何数据，缓存对于请求来说不是一个真正的问题。）但是，连续媒体数据--一般是是在RTSP之外传输--和会话描述，期望被缓存。

收到一个SETUP或PLAY请求后，代理要确认它是否有该连续媒体内容的最新拷贝和它的描述。它可以通过发出一个SETUP或DESCRIBE请求来判断拷贝是否是最新的，并跟所缓存的拷贝比较Last-Modified头部。如果拷贝不是最新的，它就修改SETUP transport参数为合适值并向原始服务器转发请求。紧跟的控制命令如PLAY或PAUSE就不修改地经过代理。代理把连续媒体数据传送给客户端，可能同时为之后的重用生成本地拷贝。缓存允许的确切的行为通过12.8节描述的cache-response指令给出。如果缓存正在服务于请求者所请求的流，缓存【必须】回答任何DESCRIBE请求，因为原始服务器上的流描述的低级别细节可能已改变。

注意：RTSP缓存和HTTP缓存不一样，它有“cut-through ”多样性。缓存直接拷贝流数据，就像数据是从它那里经过到达客户端一样。因此它没有引入额外的延时。

对客户端，RTSP代理的缓存表现得像是常规的媒体服务器一样；对媒体原始服务器则像客户端。就和HTTP需要储存所缓存对象的内容类型、内容语言等等一样，媒体缓存需要储存表述描述。典型地，缓存会消除表示描述中的所有transport-引用（即多播信息），因为从缓存到客户端的传送不需要这些。编码信息则保持不变。如果缓存有能力转译 所缓存的媒体数据，它会生成一个新的包括了所有它能提供的编码可能的表示描述。

14 示例

下列示例涉及到非标准的流描述格式，如RTSL。下面的例子不是用作这些格式的参考。

14.1 按需点播（单播）

客户端C向媒体服务器A（audio.example.com）和V（video.example.com）请求一个电影。媒体描述储存在web服务器W。媒体描述包含表示和它所有流的描述，包括有效的编解码器、动态RTP载荷类型、协议栈，以及如语言或拷贝限制这类内容信息。它还可能会给出电影时间信息指示。

在该例子中，客户端只对电影的最后部分感兴趣。

C->W: GET /twister.sdp HTTP/1.1
Host: www.example.com
Accept: application/sdp

W->C: HTTP/1.0 200 OK
Content-Type: application/sdp
v=0
o=- 2890844526 2890842807 IN IP4 192.16.24.202
s=RTSP Session
m=audio 0 RTP/AVP 0
a=control:rtsp://audio.example.com/twister/audio.en
m=video 0 RTP/AVP 31
a=control:rtsp://video.example.com/twister/video

C->A: SETUP rtsp://audio.example.com/twister/audio.en RTSP/1.0
CSeq: 1
Transport: RTP/AVP/UDP;unicast;client_port=3056-3057

A->C: RTSP/1.0 200 OK
CSeq: 1
Session: 12345678
Transport: RTP/AVP/UDP;unicast;client_port=3056-3057;
server_port=5000-5001
C->V: SETUP rtsp://video.example.com/twister/video RTSP/1.0
CSeq: 1
Transport: RTP/AVP/UDP;unicast;client_port=3058-3059
V->C: RTSP/1.0 200 OK
CSeq: 1
Session: 23456789
Transport: RTP/AVP/UDP;unicast;client_port=3058-3059;
server_port=5002-5003
C->V: PLAY rtsp://video.example.com/twister/video RTSP/1.0
CSeq: 2
Session: 23456789
Range: smpte=0:10:00-
V->C: RTSP/1.0 200 OK
CSeq: 2
Session: 23456789
Range: smpte=0:10:00-0:20:00
RTP-Info: url=rtsp://video.example.com/twister/video;
seq=12312232;rtptime=78712811
C->A: PLAY rtsp://audio.example.com/twister/audio.en RTSP/1.0
CSeq: 2
Session: 12345678
Range: smpte=0:10:00-
A->C: RTSP/1.0 200 OK
CSeq: 2
Session: 12345678
Range: smpte=0:10:00-0:20:00
RTP-Info: url=rtsp://audio.example.com/twister/audio.en;
seq=876655;rtptime=1032181
C->A: TEARDOWN rtsp://audio.example.com/twister/audio.en RTSP/1.0
CSeq: 3
Session: 12345678
A->C: RTSP/1.0 200 OK
CSeq: 3
C->V: TEARDOWN rtsp://video.example.com/twister/video RTSP/1.0
CSeq: 3
Session: 23456789
V->C: RTSP/1.0 200 OK
CSeq: 3

尽管音轨和视频轨在两个不同的服务器上，并可能在起始时间上有时间上的微小差别及互相有偏移量，客户端可以用标准RTP方法同步它们，具体就是RTCP发送方包括所包含的时间刻度。

14.2 容器文件的流化

在本示例内，容器文件就是可以容纳服务于同一个终端用户的表示的多个连续媒体类型的存储实体。在实际中，一个容器文件对应一个RTSP表示，它的每一种元素对应一个RTSP流。容器文件被广泛用于存储此类表示。而各元素作为互相独立的流来传输，它意在把一个服务端的多个流维护成一个公用的内容。

这使服务器可以用一个存储句柄方便地做打开操作。它还允许了服务器平等地处理各流，从而避免了对流的任何存在优先权的处理方式。

这还使得表示所有者可以防止别人为了得到该多媒体表示的艺术效果而通过客户端选择性地摘取流。类似地，在这样紧密捆绑的表示中，我们可能希望可以通过一个控制消息，使用一个合URL来控制所有的流。

下面是一个使用一个RTSP会话控制多个流的例子。它也示范了合URL的使用。

客户端C向媒体服务器M请求一个表示。该电影存在一个容器文件中。客户端已经取得了该容器文件的RTSP URL。

C->M: DESCRIBE rtsp://foo/twister RTSP/1.0
CSeq: 1
M->C: RTSP/1.0 200 OK

CSeq: 1
Content-Type: application/sdp
Content-Length: 164
v=0
o=- 2890844256 2890842807 IN IP4 172.16.2.93
s=RTSP Session
i=An Example of RTSP Session Usage
a=control:rtsp://foo/twister
t=0 0
m=audio 0 RTP/AVP 0
a=control:rtsp://foo/twister/audio
m=video 0 RTP/AVP 26
a=control:rtsp://foo/twister/video
C->M: SETUP rtsp://foo/twister/audio RTSP/1.0
CSeq: 2
Transport: RTP/AVP;unicast;client_port=8000-8001
M->C: RTSP/1.0 200 OK
CSeq: 2
Transport: RTP/AVP;unicast;client_port=8000-8001;
server_port=9000-9001
Session: 12345678
C->M: SETUP rtsp://foo/twister/video RTSP/1.0
CSeq: 3
Transport: RTP/AVP;unicast;client_port=8002-8003
Session: 12345678
M->C: RTSP/1.0 200 OK
CSeq: 3
Transport: RTP/AVP;unicast;client_port=8002-8003;
server_port=9004-9005
Session: 12345678
C->M: PLAY rtsp://foo/twister RTSP/1.0
CSeq: 4
Range: npt=0-
Session: 12345678
M->C: RTSP/1.0 200 OK
CSeq: 4
Session: 12345678
RTP-Info: url=rtsp://foo/twister/video;
seq=9810092;rtptime=3450012
C->M: PAUSE rtsp://foo/twister/video RTSP/1.0
CSeq: 5
Session: 12345678
M->C: RTSP/1.0 460 Only aggregate operation allowed
CSeq: 5
C->M: PAUSE rtsp://foo/twister RTSP/1.0
CSeq: 6
Session: 12345678
M->C: RTSP/1.0 200 OK
CSeq: 6
Session: 12345678
C->M: SETUP rtsp://foo/twister RTSP/1.0
CSeq: 7
Transport: RTP/AVP;unicast;client_port=10000
M->C: RTSP/1.0 459 Aggregate operation not allowed
CSeq: 7

第一个失败实例里，客户端试图暂停该表示的一个流（此处是视频）。服务器不允许对该表示这样操作。第二个实例里，合URL不应该被用在SETUP上，每个流需要一个控制消息来设置传输参数。

这保持了Transport头部的语法，使防火墙可以容易地解析transport信息。

14.3 单流容器文件

一些RTSP服务器可能会把所有的文件当作容器文件来处理，而另一些服务器可能不支持这样的观点。因此，客户端【应该】用该请求URL的会话描述中给的规则，而不是假设在所有场合用的都是复合URL。这里是一个多流服务器期望如何提供一个单流文件的示例：

```
        Accept: application/x-rtsp-mh, application/sdp
        CSeq: 1
   S->C  RTSP/1.0 200 OK
        CSeq: 1
        Content-base: rtsp://foo.com/test.wav/
        Content-type: application/sdp
        Content-length: 48
        v=0
        o=- 872653257 872653257 IN IP4 172.16.2.187
        s=mu-law wave file
        i=audio test
        t=0 0
        m=audio 0 RTP/AVP 0
        a=control:streamid=0
   C->S  SETUP rtsp://foo.com/test.wav/streamid=0 RTSP/1.0
        Transport: RTP/AVP/UDP;unicast;
                client_port=6970-6971;mode=play
        CSeq: 2
  S->C  RTSP/1.0 200 OK
        Transport: RTP/AVP/UDP;unicast;client_port=6970-6971;
                server_port=6970-6971;mode=play
        CSeq: 2
        Session: 2034820394
   C->S  PLAY rtsp://foo.com/test.wav RTSP/1.0
        CSeq: 3
        Session: 2034820394
   S->C  RTSP/1.0 200 OK
        CSeq: 3
        Session: 2034820394
        RTP-Info: url=rtsp://foo.com/test.wav/streamid=0;
          seq=981888;rtptime=3781123
```

注意SETUP命令里的不同URL，和之后在PLAY命令里切换回合URL。这在有多个流被合控制的情况下完全合理，但在流的数量为一时就不那么直观了。

这种特殊情况下，推荐的是服务器能够容许实现发送：

```
   C->S  PLAY rtsp://foo.com/test.wav/streamid=0 RTSP/1.0
        CSeq: 3
```

在最坏的情况里，服务器会返回：

```
   S->C  RTSP/1.0 460 Only aggregate operation allowed
        CSeq: 3
```

有些人还希望服务器实现也能容许下面这样的：

```
   C->S  SETUP rtsp://foo.com/test.wav RTSP/1.0
        Transport: rtp/avp/udp;client_port=6970-6971;mode=play
        CSeq: 2
```

因为文件里只有一个流，所以它也不会产生歧义。

14.4 现场媒体表示使用多播

媒体服务器M选择多播地址和端口号。在这里，我们假设web服务器只包含指向完整描述的指针，而媒体服务器M维护着完整描述。

```
    C->W: GET /concert.sdp HTTP/1.1
        Host: www.example.com
    W->C: HTTP/1.1 200 OK
        Content-Type: application/x-rtsl
         <session>
          <track src="rtsp://live.example.com/concert/audio">
        </session>
```

C->M: DESCRIBE rtsp://live.example.com/concert/audio RTSP/1.0
CSeq: 1
M->C: RTSP/1.0 200 OK
CSeq: 1
Content-Type: application/sdp
Content-Length: 44
v=0
o=- 2890844526 2890842807 IN IP4 192.16.24.202
s=RTSP Session
m=audio 3456 RTP/AVP 0
a=control:rtsp://live.example.com/concert/audio
c=IN IP4 224.2.0.1/16
C->M: SETUP rtsp://live.example.com/concert/audio RTSP/1.0
CSeq: 2
Transport: RTP/AVP;multicast
M->C: RTSP/1.0 200 OK
CSeq: 2
Transport: RTP/AVP;multicast;destination=224.2.0.1;
port=3456-3457;ttl=16
Session: 0456804596
C->M: PLAY rtsp://live.example.com/concert/audio RTSP/1.0
CSeq: 3
Session: 0456804596
M->C: RTSP/1.0 200 OK
CSeq: 3
Session: 0456804596

14.5 向已存在的会话播放媒体

会议参与者C想要让媒体服务器M向已存在的会议播放一段演示。C向媒体服务器指出网络地址和密钥已经由会议给出，所以服务器不需要选择它们。该示例忽略了简单的ACK响应。

C->M: DESCRIBE rtsp://server.example.com/demo/548/sound RTSP/1.0
CSeq: 1
Accept: application/sdp
M->C: RTSP/1.0 200 1 OK
Content-type: application/sdp
Content-Length: 44
v=0
o=- 2890844526 2890842807 IN IP4 192.16.24.202
s=RTSP Session
i=See above
t=0 0
m=audio 0 RTP/AVP 0
C->M: SETUP rtsp://server.example.com/demo/548/sound RTSP/1.0
CSeq: 2
Transport: RTP/AVP;multicast;destination=225.219.201.15;
port=7000-7001;ttl=127
Conference: 199702170042.SAA08642@obiwan.arl.wustl.edu%20Starr
M->C: RTSP/1.0 200 OK
CSeq: 2
Transport: RTP/AVP;multicast;destination=225.219.201.15;
port=7000-7001;ttl=127
Session: 91389234234
Conference: 199702170042.SAA08642@obiwan.arl.wustl.edu%20Starr
C->M: PLAY rtsp://server.example.com/demo/548/sound RTSP/1.0
CSeq: 3
Session: 91389234234
M->C: RTSP/1.0 200 OK
CSeq: 3

14.6 录制

会议参与者客户端C让媒体服务器M录制会议的音频和视频部分。客户端使用ANNOUNCE方法来向服务器提供被录制会话的元信息。

```
C->M: ANNOUNCE rtsp://server.example.com/meeting RTSP/1.0
    CSeq: 90
    Content-Type: application/sdp
    Content-Length: 121
    v=0
    o=camera1 3080117314 3080118787 IN IP4 195.27.192.36
    s=IETF Meeting, Munich - 1
    i=The thirty-ninth IETF meeting will be held in Munich, Germany
    u=http://www.ietf.org/meetings/Munich.html
    e=IETF Channel 1 <ietf39-mbone@uni-koeln.de>
    p=IETF Channel 1 +49-172-2312 451
    c=IN IP4 224.0.1.11/127
    t=3080271600 3080703600
    a=tool:sdr v2.4a6
    a=type:test
    m=audio 21010 RTP/AVP 5
    c=IN IP4 224.0.1.11/127
    a=ptime:40
    m=video 61010 RTP/AVP 31
    c=IN IP4 224.0.1.12/127
M->C: RTSP/1.0 200 OK
    CSeq: 90
C->M: SETUP rtsp://server.example.com/meeting/audiotrack RTSP/1.0
    CSeq: 91
    Transport: RTP/AVP;multicast;destination=224.0.1.11;
            port=21010-21011;mode=record;ttl=127
M->C: RTSP/1.0 200 OK
    CSeq: 91
    Session: 50887676
    Transport: RTP/AVP;multicast;destination=224.0.1.11;
            port=21010-21011;mode=record;ttl=127
C->M: SETUP rtsp://server.example.com/meeting/videotrack RTSP/1.0
    CSeq: 92
    Session: 50887676
    Transport: RTP/AVP;multicast;destination=224.0.1.12;
            port=61010-61011;mode=record;ttl=127
M->C: RTSP/1.0 200 OK
    CSeq: 92
    Transport: RTP/AVP;multicast;destination=224.0.1.12;
            port=61010-61011;mode=record;ttl=127
C->M: RECORD rtsp://server.example.com/meeting RTSP/1.0
    CSeq: 93
    Session: 50887676
    Range: clock=19961110T1925-19961110T2015
M->C: RTSP/1.0 200 OK
    CSeq: 93
```

15 语法

RTSP语法和RFC2068[2]一样用增广Backus-Naur form (BNF)来描述。

15.1 基本语法

```
OCTET          =    <任何 8位顺序数据>
CHAR           =    <任何 US-ASCII 字符 (0 - 127的八位组)>
UPALPHA        =    <任何 US-ASCII 大写字母 "A".."Z">
LOALPHA        =    <任何 US-ASCII 小写字母 "a".."z">
ALPHA          =    UPALPHA | LOALPHA
DIGIT          =    <任何 US-ASCII 数字 "0".."9">
CTL            =    <任何 US-ASCII 控制字符
```

(0 - 31的八位组) 和 DEL (127)>
CR = <US-ASCII CR, 回车 (13)>
LF = <US-ASCII LF, 换行 (10)>
SP = <US-ASCII SP, 空格 (32)>
HT = <US-ASCII HT, 水平制表符 (9)>
<"> = <US-ASCII 双引号 (34)>
CRLF = CR LF
LWS = [CRLF] 1*(SP | HT)
TEXT = <任何 OCTET 除了 CTLs>
tspecials = "(" | ")" | "<" | ">" | "@"
| "," | ";" | ":" | "\" | <">
| "/" | "[" | "]" | "?" | "="
| "{" | "}" | SP | HT
token = 1*<任何 CHAR 除了 CTLs or tspecials>
quoted-string = (<"> *(qdtext) <">)
qdtext = <任何 TEXT 除了 <">>
quoted-pair = "\" CHAR
message-header = 域-名称 ":" [域-值] CRLF
field-name = token
field-value = *(域-内容 | LWS)
field-content = <the 构成 field-value，组成
*TEXT 或token、 tspecials、
quoted-string组合成的OCTETs>
safe = "$" | "-" | "_" | "." | "+"
extra = "!" | "*" | "'" | "(" | ")" | ","
hex = DIGIT | "A" | "B" | "C" | "D" | "E" | "F" |
"a" | "b" | "c" | "d" | "e" | "f"
escape = "%" hex hex
reserved = ";" | "/" | "?" | ":" | "@" | "&" | "="
unreserved = alpha | digit | safe | extra
xchar = unreserved | reserved | escape

16 安全考虑

由于RTSP服务器和HTTP服务器在语法和使用上的相似性，应用[H15]所述的安全机制。请特别注意以下内容：

鉴权机制：

RTSP和HTTP分享相同的鉴权方案，因此应该遵循相同的鉴权原则。客户端鉴权课题见[H15.1],多鉴权机制支持的相关课题见[H15.2]。

服务器日志信息的误用：

RTSP和HTTP服务器将有大致类似的日志记录机制，因此应该平等地对待日志内容保护，及用户和服务器隐私的保护。HTTP服务器日志相关的服务器推荐见[H15.3]。

敏感信息的传递：

没有理由认为RTSP上传输的信息会没有一般HTTP上传输的信息那么敏感。因此，所有涉及到保护数据隐私权和用户隐私权的警示，是每个RTSP客户端、服务器、代理的实现者都需要考虑的。更多细节见[H15.4]。

基于文件和路径名的攻击：

尽管RTSP URL是不透明的句柄，没必要有文件系统语义，但还是期望很多实现能够把请求URL的一部分直接转化成文件系统目录。此种情况下，文件系统【应该】遵循[H15.5]给出的警示，例如检查路径元素中的“..”。

个人信息：

一般在HTTP上是隐私的信息（用户名，地址，等等），在RTSP上也是，因此它们应该对等。更多推荐意见见[H15.6]。

Accept头部相关的隐私问题：

由于RTSP中存在很多和HTTP一样的“Accept”头部，因此也要遵循[H15.7]中关于它们的使用的相同的警示。

DNS欺骗：

可以推测，如果给典型的和HTTP会话相关的RTSP会话更长连接时间，RTSP客户端DNS优化应该会少些普遍性。尽管如此，对于任何试图依靠DNS-to-IP的映射来保持该映射的简单使用的实现，[H15.8]提供的推荐意见仍然有普遍意义。

地址头部及欺骗：

如果一个支持多优化的服务器不相信其他服务器，那么它必须检查响应中在上述优化的控制下产生的Location和Content-Location头部的值，以确保它们不会向没有权限（[H15.9]）的无效资源作尝试。

作为现有HTTP规范（本规范编写时的RFC 2068 [2]）的推荐的补充，将来的HTTP规范可能提供安全方面的补充指导。

下面是RTSP实现的额外考虑：

集中的拒绝服务式攻击：

协议提供了用远程控制来进行拒绝服务式攻击的机会。攻击者可能会通过修改SETUP请求的目的地地址，向一个或多个IP地址发起传输流。该情况中可能可以看到攻击者的IP，但对于预防更多的攻击者或确认攻击者的标识这并不总是管用。因此，RTSP服务器【应该】只允许客户端为RTSP发起的传输流指定目的地址，如果服务器验证过客户端标识--通过建立一个已知的使用RTSP鉴权机制的用户的数据库（可能是digest鉴权或者更强），或者其他手段--的话。

会话劫持：

因为传输层连际和RTSP之间没有直接联系，恶意客户端就可能以发起会话标识为随机值的请求来影响正常的客户端。服务器【应该】使用较大的随机无序会话标识来降低此种攻击的机会。

鉴权：

服务器应该同时实现基本的和digest[8]的鉴权。在需要控制消息有较强安全性的环境，RTSP控制流可能会被加密。

流课题：

RTSP仅提供流控制。在此章节及余下的备忘里，不包括流传输课题。RTSP实现最有可能采取的方式是依赖于其他协议如RTP、IP multicast、 RSVP 和 IGMP，并应解决这些或其他适用规范所带来的安全问题。

持续性可疑行为：

RTSP服务器【应该】在收到一个进程的存在安全隐患的行为后返回错误代码403（禁止）。RTSP服务器也【应该】能意识到探测服务器弱点和进口的尝试，并【可能】强行断开连接并忽略认为违反了本地安全策略的客户端的其他请求。

附录A：RTSP 协议状态机

RTSP客户端和服务器状态机给出了协议从RTSP会话初始化到RTSP会话终止之间的表现。

状态是基于每个对象定义的。一个对象由流URL和RTSP会话标识唯一地标识出来。任何使用跟多个流构成的表示的合URL相关的请求/回复，都对所有单个的流的状态有影响。例如，如果一个表示/movie包含两个流，/movie/audio 和 /movie/video，那么下列命令：

```
PLAY rtsp://foo.com/movie RTSP/1.0
CSeq: 559
Session: 12345678
```

将对/movie/audio 和 /movie/video的状态有影响。

该示例没有示范标准URL形式或和文件系统的关系的意思。见3.2节。

OPTIONS, ANNOUNCE, DESCRIBE, GET_PARAMETER,SET_PARAMETER 请求对客户端或服务器的状态没有影响，因此未列入状态表。

A.1 客户端状态机

客户端可以处于如下的状态：

Init（初始）：

已经发出SETUP，等待回应中。

Ready（就绪）：

SETUP回复已收到或在Playing状态中PAUSE回复已收到。

Playing（播放）：

PLAY回复已收到。

Recording（录制）：

RECORD回复已收到。

总之，客户端在收到请求的回复之后改变状态。注意某些请求是在未来的时间或位置起作用的（如PAUSE），而状态的改变也要视情况而定。如果对象（例如多播组有效的）不要求显式的SETUP，状态自Ready开始。在此情况就只有两种状态，Ready和Playing。当到达所请求的range的结尾时，的客户端也会从Playing/Recording状态转到Ready。

“下一状态”栏指示接受到成功响应（2xx）后可能的状态。如果请求所得来的状态码是3xx，那么状态变成Init，而4xx状态码不会导致状态改变。没有在对应状态列出的消息，【必须不】在此状态下被客户端发出--例外是上文列出的对状态没有影响的消息。从服务器收到一个REDIRECT等同于收到3xx重定向状态。

状 态	可发消息	响应后下一状态
Init	SETUP	Ready
	TEARDOWN	Init
Ready	PLAY	Playing
	RECORD	Recording
	TEARDOWN	Init
	SETUP	Ready
Playing	PAUSE	Ready
	TEARDOWN	Init
	PLAY	Playing

	SETUP	Playing (changed transport)
Recording	PAUSE	Ready
	TEARDOWN	Init
	RECORD	Recording
	SETUP	Recording (changed transport)

A.2 服务器状态码

服务器可以处于如下状态：

Init（初始）：

初始化状态，没有收到有效SETUP。

Ready（就绪）：

最近收到的SETUP请求成功，回复已发送；或在播放中，最近收到的PAUSE请求成功，回复已发送。

Playing（播放）：

最近收到的PLAY请求成功，回复已发送。数据开始发送。

Recording（录制）：

服务器正在录制媒体数据。

总之，服务器根据收到的请求改变状态。如果服务器处于Playing或Recording和单播模式下，且如果在给定期间没有从客户端收到“健康的”信息，例如RTCP报告或RTSP命令，服务器【可能】会转到Init状态或关闭RTSP会话。服务器可以在会话响应头部（12.37节）声明另一个超时值。如果服务器状态为Ready，且在一分钟或者更长时间内没有收到RTSP请求，它【可能】转为Init状态。注意某些请求是在未来的时间或位置起作用（如PAUSE），服务器在合适的时间状态的改变。在到达客户端所请求范围的末端后，服务器从Playing或Recording状态转为Ready。

REDIRECT消息一旦发出就立即生效，除非在Range头部中另外指出了重定向何时生效。此种情况中，服务器状态也是在合适的时间才改变。

如果对象不要求有显式的SETUP，则状态从Ready开始，且只有Ready和Playing两个状态。

“下一状态”栏指示接受到成功响应（2xx）后可能的状态。如果请求所得来的状态码是3xx，那么状态变成Init，而4xx状态码不会导致状态改变。

状 态	可收消息	下一状态
Init	SETUP	Ready
	TEARDOWN	Init
Ready	PLAY	Playing
	SETUP	Ready
	TEARDOWN	Init
	RECORD	Recording
Playing	PLAY	Playing
	PAUSE	Ready
	TEARDOWN	Init
	SETUP	Playing
Recording	RECORD	Recording
	PAUSE	Ready
	TEARDOWN	Init
	SETUP	Recording

附录B：同RTP交互

RTSP允许媒体客户端控制媒体表示中选择出的非连续片段，表现为RTP媒体层[24]的流。表现为RTP流的媒体层不应该受NPT时间的跳跃的影响。因此，RTP序列号和RTP时间戳在跨越NPT时间跳跃时都【必须】是连续且单调递增的。

例如，假设时钟频率是8000Hz，包之间的时间间隔是100毫秒，初始序列号和时间戳都是0。我们首先播放NPT 10到NPT 15，然后向前跳播放NPT 18到NPT 20。第一个片段在RTP包上表现为序列号从0到49，时间戳从0到39，200。第二个片段则表现为序列号从50到69，时间戳从40，000到55，200。

我们不能假定RTSP客户端可以与RTP媒体agent通信，因为这可能是两个独立的进程。如果RTP时间戳的时间间隔和NPT一样，媒体agent将假设表示有暂停 。如果NPT时间跳跃足够大，RTP时间戳可能会到达上限而从头开始，这样媒体agent可能会认为之后的包跟之前已播放的包重复了。

对于某些数据类型，RTSP层和RTP层之间的紧密整合是有必要的。这完全不能排除上文所述的限制。RTSP/RTP组合媒体客户端应该用RTP-Info域来判断要来的RTP包是在搜索之前还是之后发的。

对于连续音频，服务器【应该】在服务于新PLAY请求之始就设置RTP标志位。这使客户端可以进行播放延迟调整 。

对于倍速改变（12.34节），RTP时间戳应该和回放计时一致。例如，当以两倍速、speed为1播放30帧/秒的视频记录，服务器应该每两帧丢弃一帧以维持用间隔为3000每帧的正常时间戳传送视频流，但是NPT每一视频帧增加1/15秒。

客户端可以通过通知位置重设后的第一个包的RTP时间戳值，来维护正确NPT的播放。RTP-info（12.33）头部的sequence参数提供下个片段的首个序列号。

附录C：用SDP描述RTSP会话

会话描述协议（SDP, RFC 2327 [6]）可能会RTSP被用于描述流或表示。这类使用被限制在以下几个方面，用于给出访问方式和编码：

合控制：

不能够合控制的由来自于一个或多个服务器多个流组成的表示。这类描述通常由其他途径得来，以HTTP为典型。不它们也可以通过ANNOUNCE方法接收。

非合控制：

能够合控制的由多个来自于一个服务器的流组成的表示。这类描述通常在对一个URL的 DESCRIBE请求的回复中给出，或通过ANNOUNCE方法接收。

该附录描述一个SDP文件，例如通过HTTP获取的,如何决定一个RTSP会话上的操作。它还描述了客户端如何解释DESCRIBE请求的回复所给的SDP内容。SDP不为客户端提供能够不用人工指引，而区分多个要同时提交的媒体流和多个互为替换的流的集合（例如，两个讲不同语言的音频流）。

C.1 定义

该附件使用的词条"会话级别","媒体级别"和其他键/属性名及值的定义在SDP（RFC 2327[6]中：

C.1.1 控制URL

“a=control”属性用来传送控制RUL。该属性可用在会话和媒体描述上。；如果用于单个媒体，它指示用来控制那个特定媒体流的URL。如果用在会话级别，该属性指示合控制的URL。

例子：

a=control:rtsp://example.com/foo

该属性可能包含相对或者绝对URL，遵循RFC 1808 [25]的规则和习惯。实现应该按以下顺序寻找基URL：

1. RTSP Content-Base 域
2. RTSP Content-Location 域
3. RTSP 请求 URL

如果此属性只包含星号（*）,则URL被以空内嵌UL对待，因此继承整个基URL。

C.1.2 媒体- 流

"m="被用来列举流。它期望所有列举出的流都将有合适的同步处理。如果会话是单播，端口号则是服务器对客户端的推荐；客户端在SETUP中还是应该在包括它并有可能忽略此推荐。如果服务器没有偏好，它【应该】把端口号值置零。

例子：

m=audio 0 RTP/AVP 31

C.1.3 载荷类型

载荷类型以"m="域给出。若载荷类型是RFC 1890[1]里的静态载荷类型，则不需要其他信息。若是动态载荷类型，则用媒体属性“rtpmap”说明是什么媒体。“rtpmap”属性中的“encoding name”可能是RFC 1890（章节5和6）中规定的一中，或SDP（RFC 2327 [6]）规定的“X-”为前缀的实践性编码。Codec-specific（编码器- 细节）参数不在此域之中，而在下文描述的“fmtp”属性中。想要注册新编码的实现者应该遵循RFC 1890 [1]中的手续。如果媒体类型不匹配RTP AV规范，那么推新建一个子集并使用合适的名字，以代替"m="域的"RTP/AVP"。

C.1.4 格式- 细节 参数

格式细节参数用“fmtp“媒体参数表达。”fmtp“的语法具体对应于属性所指的编码。注意打包时间间隔用“ptime”属性传递。

C.1.5 表示的范围

"a=range"属性定义所储会话的总时间范围。（现场会话的长度可以由“t”和“r”属性推导出。）除非表示含有不同持续时间的媒体流、否则范围属性是会话级别的属性。首先给出的是单位，后跟值范围。单位和它们的值在3.5、3.6、3.7节中给出定义。

例子：

a=range:npt=0-34.4368

a=range:clock=19971113T2115-19971113T2203

C.1.6

“t=”域【必须】包含合适的合控制或非合控制流的开始和停止时间的值。在合控制里，服务器【应该】指示保证描述有效的停止时间值，以及等于或早于收到DESCRIBE请求时刻的开始时间。它也【可能】把开始和停止时间指示为0，意为会话始终有效。在非合控制里，那些值应该反映会话能有效保持SDP语义的时期，且不需要借助其他途径（如包含描述的web页面的生命期）来达到此目的。

C.1.7 连接信息

在SDP里，“C=”域包含媒体流的目的地址。但是，对于按需点播流和一些多播流，目的地址由客户端通过SETUP请求给出。除非媒体内容有固定的目的地址，否则”C=”域将被置为合适的空值。对于型为“IP4“的地址，该值为"0.0.0.0"。

C.1.8 实体标签

可选的“a=etag”属性标识会话描述的版本。它对客户端不透明。SETUP请求可能在If-Match域（见12.22节）包含此标识，以在当该属性值仍和当前描述中的对应时只允建立会话。该属性值是不透明的且可以包含SDP属性所允许包含的任意值。

例子：

a=etag:158bb3e7c7fd62ce67f12b533f06b83a

有人可能会说“o=”域提供相同的功能。但是，这样做可以对需为同一个媒体内容片段提供多种SDP以外的的会话描述类型支持的服务器进行强制要求。

C.2 合控制无效

如果一个给出了多个媒体部分的表示不支持合控制，【必须】通过"a=control:"属性给出每个部分的控制URL。

例子：

```
v=0
o=- 2890844256 2890842807 IN IP4 204.34.34.32
s=I came from a web page
t=0 0
c=IN IP4 0.0.0.0
m=video 8002 RTP/AVP 31
a=control:rtsp://audio.com/movie.aud
m=audio 8004 RTP/AVP 3
a=control:rtsp://video.com/movie.vid
```

注意：描述中控制URL的位置暗示客户端要分别建立到服务器audio.com 和 video.com的RTSP控制会话。

推荐在SDP文件中包含完整的媒体初始化信息，即使SDP是通过非RTSP途径传送到媒体客户端的。这是有必要的，因为没有机制可以指示客户端应该通过DESCRIBE请求更多媒体流细节信息。

C.3 合控制有效

此种前提下，服务器有可以被当作一个整体控制的多个流。在此情况下，就即有用于给出流URL的媒体级"a=control:"属性，又有用作合控制的请求URL的会话级"a=control:"。如果媒体级URL是相对URL，它将根据上文C1.1节所述被处理为绝对URL。

如果表示只由单个流组成，媒体级"a=control:"将被全部忽略。但是，如果表示包含多于一个流，每个媒体流部分【必须】包含自己的"a=control:"属性。

例子：

```
v=0
o=- 2890844256 2890842807 IN IP4 204.34.34.32
s=I contain
i=<more info>
t=0 0
c=IN IP4 0.0.0.0
a=control:rtsp://example.com/movie/
m=video 8002 RTP/AVP 31
a=control:trackID=1
m=audio 8004 RTP/AVP 3
a=control:trackID=2
```

在此实例里，客户端被要求建立一个到服务器的RTSP会话，并用分别用URL rtsp://example.com/movie/trackID=1 和rtsp://example.com/movie/trackID=2 建立视频和音频流。URL rtsp://example.com/movie/控制整个电影。

附录D：最小RTSP实现

D.1 客户端

客户端实现【必须】能够做到如下几点：

*生成下列请求：SETUP, TEARDOWN, 和 PLAY (意即, 一个最小回放客户端) 或 RECORD (意即, 一个最小录制客户端)。如果RECORD被实现，ANNOUNCE【必须】也被实现。

*在请求中包含下列头部：CSeq, Connection, Session, Transport。如果 ANNOUNCE 被实现，也应该实现包含Content-Language, Content-Encoding, Content-Length, 及 Content-Type 头部的能力。

*解析并理解下列响应中的头部：CSeq,Connection, Session, Transport, Content-Language, Content-Encoding, Content-Length, Content-Type。如果实现了RECORD，则也必须理解Location头部。RTP-compliant（RTP适用的）实现应该再实现RTP-Info。

*如果收到一个型别为4xx 或5xx的错误代码，理解所收到错误代码的型别并通知最终用户。如果最终用户明确指出不想要一个或所有状态码的产生条件，则可以不进行通知。

*期待并回复服务器的异步请求，例如ANNOUNCE。这并不是说它一定要实现ANNOUNCE方法，而只是【必须】对收到的任何服务器请求作出肯定或否定响应。

虽然不是必需，但为了便于实际中同早期实现协同工作且/或成为“好市民”，在发布时仍强烈推荐实现下述功能。

* 实现 RTP/AVP/UDP ，使其成为有效 transport.
* 包含 User-Agent 头部.
* 理解附件C中定义的 SDP 会话描述
* 接受从标准输入、命令行、及其他适合操作环境的效果如同一个程序（如web浏览器）的“帮助程序”的方式等,得来的媒体初始化格式(例如 SDP)。

可能有些RTSP程序跟RTSP规范的创建者的预想不同，对于这些程序，上述要求并无意义。因此，上面的推荐仅作为指导，而非严格的要求。

D.1.1 基本回放

为支持按需点播媒体流，客户端【必须】额外地支持：

*生成PAUSE请求

*实现REDIRECT方法及Location头部

D.1.2 鉴权- 激活

为访问要求鉴权的RTSP服务器上的媒体表示，客户端【必须】额外地支持：

*识别401状态码

*解析并包括WWW-Authenticate头部

*实现基本鉴权和Digest鉴权

D.2 服务器

最小服务器实现【必须】能够做到：

*实现下列方法：SETUP, TEARDOWN, OPTIONS 以及PLAY (对于最小回放服务器) or RECORD (对于最小录制服务器)。如果实现RECORD, ANNOUNCE也应该实现。

*包含下列头部：Connection, Content-Length, Content-Type, Content-Language, Content-Encoding, Transport, Public。如果实现RECORD方法，也应实现包含Location头部的能力。RTP-compliant（RTP适应的）实现还应该实现RTP-info域。

*正确解析及响应请求中的下列头部：Connection, Session, Transport, Require。

虽然不是必需，但为了便于实际中同早期实现协同工作且/或成为“好市民”，在发布时仍强烈推荐实现下述功能。

* 实现 RTP/AVP/UDP ，使其成为有效 transport。

* 包含 Server 头部。

* 实现DESCRIBE方法。

* 按附录C中的定义生成SDP会话描述。

可能有些RTSP程序跟RTSP规范的创建者的预想不同，对于这些程序，上述要求并无意义。因此，上面的推荐仅作为指导，而非严格的要求。

D.2.1 基本回放

为支持媒体流按需点播，服务器【必须】额外地支持：

*识别Range头部，并在不支持搜索时返回error。

*实现PAUSE方法。

此外，为支持普遍接受的用户界面特性，高度推荐按需点播媒体服务器支持以下几点：

*包含并解析Range头部，使用 NPT单位。并推荐实现SMPTE单位。

*在媒体初始化信息中包含媒体表示的长度。

*包含从数据- 相关 时间戳到NPT的映射。使用RTP时，用RTP-info域的rtptime部分来把RTP时间戳映射到NPT。

客户端实现可能会用length信息的出现与否来判断剪辑是否可搜索，并据此屏蔽没有length信息的剪辑的搜索特性。常见的表示长度的应用是实现一个“滑动条”，它即可以指示进度，又可以指示时间位置。

RTP时间戳到NPT的映射是必要的，以保证滑动条位置正确。

D.2.2 鉴权- 激活

为正确处理客户端鉴权，服务器【必须】额外支持：

*当资源需要鉴权时生成401状态码

*解析并包括WWW-Authenticate头部

*实现基本鉴权和Digest鉴权

指此协议的发布方式，与本协议内容无关。

只提供控制层协议，不包括传输层

如视频会议等

言下之意似乎是对于不同的媒体类型，Media parameter可能会不同？望高手指教。

事实上我觉得没有那么简单

指延续使用了http上的一些机制

http服务器是无状态的，而rtsp服务器有

指媒体流的传输和rtsp使用同一个tcp连接

指独立于RTSP协议所用的传输层协议，而HTTP协议的数据传输是和HTTP本身共用同一个传输层协议的。

HTTP协议没有状态机制

啰嗦一句，中文翻译中有的“实现”是动词，有的是“动名词”，但有些是纯粹的名词，指解决方案，请大家根据上下文理解。

指不是基于字符的，不限于某种字符集

这应该是说仅支持US-ASCII的程序无法支持该字符集

许多程式由於各式各样的原因，并未考虑到输入的资料可能是 non-ASCII 码的问题。它往往假设了它所要处理的资料都是 ASCII 码，更糟糕的是，当它遇到 non-ASCII 码时，常常假设它不存在，而将它的第八个位元截去! 这是所谓的 8-bit clean 问题。

unchanged in value following multiplication by itself

指无法理解后两位内容的

round-trip-time，一项衡量网络延迟的指标

RTT(Round-Trip Time): 往返时延，在计算机网络中它也是一个重要的性能指标，它表示从发送端发送数据开始，到发送端收到来自接收端的确认（接收端收到数据后便立即发送确认），总共经历的时延;

其实就是类似flashget等程序的启动方式：一旦点击或拷贝了下载路径，就自动开始工作。（我只是举例说明它的启动方式，不是说要实现那种后台监控功能）

指使服务器端能支持RTSP标准协议的最小功能集
指使客户端能以最简单的功能集支持RTSP标准协议
所以客户端应尽可能把自己支持的传输方式都列上去，以便服务器做出选择：列出的方式越多，传输成功建立的可能性就越大
这当然是指对同一个客户端而言。
一种绝对时间，其内容相当于2007年4月12日3:05:37这种。
应该是意指范围的形式不同，而实际意义相同
即我们可以让图像继续播放，而声音暂停传输。
应是指当前还没结束或者还没开始的一个PLAY请求。
指对参数值的设定
即尽量不把会由于不同原因被拒绝的参数放在一起
原子操作即不可以或者不应该再分割的操作，写过多线程程序的人应该有体会。又例如：智能收费卡的pos机上，"读取当前余额+扣除本次消费+更新余额信息”应该被实现为一次原子操作，要么全部成功，要么全部失败，否则就容易出现学生卡已被扣费食堂却说缴费失败这类问题。
指媒体内的时间而言
常见的情况是网关不允许UDP传输或只打开了HTTP端口
end-to-end authentication：与nod-to-nod authentication不同，中间有可能经过其他节点，但无论其他节点如何，会保证这两端的鉴权
这是一个时间段
这是一个时间点
在我找到的RFC文档中应该是[H14.35.1] 和什么相关?
例如对视频进行拖放
应该是指光RTCP还不是充分条件
endpoint address 端点地址: 对于任何赋予计算机能用作目的地地址的一类地址术语。例如,IP地址是一种端点地 址。
意为“直达，切入式，直通的”
我不太清楚这是指语言翻译呢还是编码格式的转换
不太理解