RTSP 协议中文版

Network Working Group
Request for Comments: 2326
Category: Standards Track

H. Schulzrinne
Columbia U.
A. Rao
Netscape
R. Lanphier
RealNetworks
April 1998

翻译： radium 2005.1

实时流协议(RTSP) (Real Time Streaming Protocol (RTSP))

备忘录的状态：

本文档讲述了一种 Internet 社区的 Internet 标准跟踪协议，它需要进一步进行讨论和建议以得到改进。请参考最新版的“Internet 正式协议标准”(STD1)来获得本协议的标准化程度和状态。本备忘录的发布不受任何限制。

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，使实时数据，如音频与视频的受控、点播成为可能。数据源包括现场数据与存储在剪辑中数据。该协议目的在于控制多个数据发送连接，为选择发送通道，如 UDP、组播 UDP 与 TCP，提供途径，并为选择基于 RTP (RFC1889)上传送机制提供方法。

目　　录：

1 绪论 Introduction

1.1 目的 Purpose

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。尽管连续媒体流与控制流有可能交叉，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远程控制。

表示描述(presentation description)定义了被控流，但本文并没有定义表示描述的格式。

这里没有使用 RTSP 连接的概念，而由 RTSP 会话(session)代替（每次服务由服务器端保持一个带标签的会话）。RTSP 会话没有绑定到传输层连接 （如 TCP 连接）。因为虽然在 RTSP 会话期间，RTSP 客户端可打开或关闭多个对服务器端的可靠传输连接以发出 RTSP 请求。但此外，也可能使用无连接传输协议，比如用 UDP 发送 RTSP 请求。

RTSP 控制的流可能用到 RTP，但 RTSP 操作并不依赖用于携带连续媒体的传输机制。实时流协议在语法和操作上与 HTTP/1.1 类似，因此 HTTP 的扩展机制大都可加入 RTSP。尽管如此，RTSP 在很多方面还是和 HTTP 有很大的不同：

? RTSP 引入了很多新方法并且有不同的协议标识符。
? RTSP 服务器在大多数默认情况下需要维持一个状态，但 HTTP 是无状态协议。
? RTSP 客户机和服务器都可以发出请求。
? 数据由另一个协议传送（有一特例除外）。
? RTSP 使用 ISO 10646(UTF-8) 而不是 ISO 8859-1，以配合当前 HTML 的国际化。
? RTSP 使用 URI 请求时包含绝对 URI。而由于历史原因造成的向后兼容性问题，HTTP/1.1 只在请求中包含绝对路径，把主机名放入单独的报头域中。这使得“虚拟主机”实现更为简便，一个单独 IP 地址的主机可虚拟为几个文件树主机。

协议支持的操作如下：

从媒体服务器上检索媒体：

用户可通过 HTTP 或其它方法请求一个表示描述。如表示是组播，表示描述就包含用于连续媒体的的组播地址和端口。如表示仅通过单播发送给用户，用户为了安全应提供目的地址。

媒体服务器邀请进入会议：

媒体服务器可被邀请参加正进行的会议，或回放媒体，或记录其中一部分，或全部。这种模式在分布式教育应用上很有用，会议中几方可轮流按远程控制按钮。

将媒体加到现成讲座中：

如服务器告诉用户可获得附加媒体内容，对现场讲座显得尤其有用。如 HTTP/1.1 中类似，RTSP 请求可由代理、通道与缓存处理。

1.2 要求 Requirement

在本文档中的关键字“必须”，“一定不能”，“要求”，“会”，“不会”，“应该”，“不应该”，“被推荐的”，“可以”，和“可选择的”都在 RFC2119 中解释。

1.3 术语 Terminology

一些术语原由 HTTP/1.1 采用。在 HTTP/1.1 中定义的术语这里不再列举。

合控制：Aggregate control
服务器使用单条时间线对多个流的控制。对音频/视频回馈来讲，这就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馈。
会议：Conference
多方参与的多媒体表示，这里的多方意味着大于或者等于一方。
客户端：Client
指请求媒体服务器上连续流媒体数据的客户端。
连接：Connection
两个应用程序以通讯为目的在传输层建立虚拟电路。
容器文件：Container file
可以容纳多个共同播放时包含表示(presentation)的媒体流的文件。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是本协议内容。
连续媒体：Continuous media
接受器和数据源之间存在时序关系的数据。也就是说，接受器需要重新产生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（但是不交互的），它们在源和接受器之间是一种紧密的时序关系；或者是流的形式，这种关系就没有那么严格了。
实体：Entity
作为请求或者回应的有效负荷传输的信息。由以实体报头域（entity-header field）形式存在的元信息和以实体主体（entity body）形式存在的内容组成，如第八章所述。
媒体的初始化：Media initialization
数据类型/编码的具体初始化，这些包括时钟输率，颜色表等。用户请求媒体回放的任何独立传输信息，是在创建流时初始化媒体流相位时产生的。
媒体参数：Media parameter
针对回放前或回放过程中有可能改变的媒体类型而专门设定的参数。
媒体服务器：Media server
可对一个或多个媒体流提供回放和录制服务的服务器。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。媒体服务器可以建立在作为传送请求表示(presentation)的 Web 服务器的主机上，也可以建立在不同的主机上。
媒体服务器重定向：Media server indirection
重新定向媒体客户端到另外一个媒体服务器。
(媒体)流： (Media) stream
单个媒体实例，比如，在应用中共用音频流或视频流。当使用 RTP 时，流包括由 RTP 会话(session)中源所创建的所有 RTP 和 RTCP 包。这和定义 DSM-CC 流时相同。
消息：Message
RTSP 通讯的基本单元。由 15 章语法定义的一串八位位组组成，并通过连接或者无连接协

议传送。

参与者：Participant

一个会议成员。参与者可以是机器，比如是媒体记录或回放服务器。

表示(presentation)：(我感觉翻译成"演出"更合适一些--gwj)

对用户请求所回馈的一组流，其使用下面的表示描述(presentation description)形式。在本文中的多数情况下，其意味着对流进行总体控制，但这并不是必须的。

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。另外，其他 IETF 协议，如 SDP 协议使 用会话(session)代替现场 presentation。表示描述可以采用包括会话描述(session description)SDP 在内的多种格式。

回应：Response

RTSP 回应。如果能理解 HTTP 回应，就能清楚的理解 RTSP 回应。

请求；Request

RTSP 请求。如果能理解 HTTP 请求，就能清楚的理解 RTSP 请求。

RTSP 会话(session)：

RTSP 交互的全过程。比如，一个电影的观看过程。会话(session)包括由客户端建立连续媒体流传输机制(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传输信息（端口号，传输协议等）的交互。

1.4 协议特点 Protocol Properties

RTSP 特性如下：

可扩展性：

RTSP 中很容易加入新方法和参数。

易解析：

RTSP 可由标准 HTTP 或 MIME 解析器解析。

安全：

RTSP 使用网页安全机制。所有 HTTP 授权机制如 basic 和 digest 授权都可直接使用。也可以传输层或网络层安全机制。

独立于传输：

RTSP 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，可使用可靠流协议。

多服务器支持：

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

记录设备控制：

协议可控制记录和回放设备，也可控制可在记录和回放切换的设备。

流控与会议开始分离：

流控与邀请媒体服务器入会分离；仅要求会议初始化协议提供，或可用来创建唯一会议标识号。特殊情况下， 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 支持帧级精度，允许远程数字编辑。

表示描述中立：

协议没强加特殊表示或元文件，可传达所用格式类型；然而，表示描述至少必须包含一个RTSP URI。

代理与防火墙友好：

协议可由应用和传输层防火墙处理。防火墙需要理解 SETUP 方法，为 UDP 媒体流打开一个\"缺口\"。

HTTP 友好：

此处，RTSP 明智的采用 HTTP 观念，使现在结构都可重用。结构包括 Internet 内容选择平台（PICS）。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。

适当的服务器控制：

如用户能启动一个流，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。

传输协调：

实际处理连续媒体流前，用户可协调传输方法。

性能协调：

如基本特征无效，必须有一些清理机制让用户决定那种方法没生效。这允许用户提出适合的用户界面。 例如，如果不允许寻找，用户界面必定能禁止位置条滑动。

以前要求 RTSP 必须能支持多用户，但现在得出一个更好的方法就是保证 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”远程通过”成为可能。协议不涉及到多个客户端如何协调入口，其由下层“社会协议”或其他下层控制机制提供。

1.5 RTSP 扩展 Extending 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请求集。例如：

? 服务器可能只支持回放而不支持录制功能。

? 对于只支持现场播放的服务器可能不支持播放定位功能。

? 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。

但服务器应该实现 12 章中所有要求的报头域。

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 中[H19.6]描述方法不支持 across server 的情形一致。

RTSP 可以如下三种方式扩展，这里以改变大小排序：

? **以新参数扩展**。如用户需要拒绝通知，而方法扩展不支持，相应标记就加入要求的段中。

? **加入新方法**。如信息接收者不理解请求，返回 501 错误代码（还未实现），发送者不应再次尝试这种方法。用户可使用 OPTIONS 方法查询服务器支持的方法。服务器使用公共回应报头列出支持的方法。

? **定义新版本协议，允许改变所有部分**。（除了协议版本号位置）

1.6 操作模式 Overall Operation

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，表示描述格式不在本协议中定义。使用 HTTP 或其它途径用户可获得这个文件，它没有必要保存在媒体服务器上。

为了说明，假设表示描述(presentation description)描述了多个表示(presentation)，其中每个表示(presentation)维持了一个公共时间轴。为简化说明，且不失一般性，假定表示描述(presentation description)的确包含这样一个表示(presentation)。表示(presentation)可包含多个媒体流。

表示描述(presentation description)即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，被 RTSP 分别控制的媒体流由 RTSP URL 表示。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负载共享。描述（description)还列出了服务器传输可使用的方法。

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 请求源。

组播，服务器选择地址：

媒体服务器选择组播地址和端口，这是现场直播或准点播常用的方式。

组播，用户选择地址：

如服务器加入正在进行的组播会议，组播地址、端口和密匙由会议描述给出。

1.7 RTSP 状态　RTSP States

RTSP 控制通过单独协议发送的流，与控制通道无关。例如，RTSP 控制可通过 TCP 连接，而数据流通过 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 连接顺序发出请求来控制。所以，服务器需要维持能联系流与 RTSP 请求的会话状态。

RTSP 中很多方法与状态无关，但下列方法在定义服务器流资源的分配与应用上起着重要的作用：

SETUP：

让服务器给流分配资源，启动 RTSP 会话。

PLAY 与 RECORD：

启动 SETUP 分配流的数据传输。

PAUSE：

临时停止流，而不释放服务器资源。

TEARDOWN：

释放流的资源，RTSP 会话停止。

标识状态的 RTSP 方法使用会话(session)报头域识别 RTSP 会话，为回应 SETUP 请求，服务器生成会话标识。

1.8 与其他协议关系 Relationship with Other Protocols

RTSP 在功能上与 HTTP 有重叠，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在网页服务器与实现 RTSP 媒体服务器之间存在不同传递点。例如，表示描述(presentation description)可通过 HTTP 和 RTSP 检索，这降低了浏览器的往返传递，也允许独立 RTSP 服务器与用户不全依靠 HTTP。

但是，RTSP 与 HTTP 的本质差别在于数据发送以不同协议进行。HTTP 是不对称协议，用户发出请求，服务器作出回应。RTSP 中，媒体用户和服务器都可发出请求，且其请求都是无状态的；在请求确认后很长时间内，仍可设置参数，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。

当大多数实时媒体使用 RTP 作为传输协议时，RTSP 没有绑定到 RTP。RTSP 假设存在表示描述格式可表示包含几个媒体流的表示的静态与临时属性。

2 符号协定 Notational Conventions

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）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散文和补充反馈的方式来描述的。除 RTSP 中以”1#”代替”,”为分隔符不同外，其余在 RFC 2234 中有详细的描述。简单说明补充反馈方式如下：

补充反馈方式（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，除非特别指定，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选其一。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-linear whitespace）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，线性空格（LWS）可以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动机和背景资料。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 Protocol Parameter

3.1 RTSP 版本　RTSP Version

同[H3.1]定义，仅用 RTSP 代替 HTTP 即可。

如下：
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）或回应（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，并回应一个错误。而低版本的请求也应在被推送前升级。代理或网关回应请求时必须和请求的版本相同。

3.2 RTSP URL

“rtsp”和“rtspu”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]中定义如下：

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]：段(fragment)和询问(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 要求使用可靠协议（Internet 的 TCP 协议）发出命令，而 rtspu 则使用不可靠协议（Internet 的 UDP 协议）。

如是端口为空或没指定，则缺省为 80 端口。对于 rtsp_URI 来说，拥有被请求的
资源的服务器主机通过侦听该端口的 TCP 连接(rtsp)或 UDP 包(rtspu)来接收该 URI 请求。

只要可能，应尽量避免的在 URL 中直接使用 IP 地址。（请参考 RFC1924）

文 本媒体标识符使用 URL 中的字符集及转义规则（参考 RFC1738）来标识一个表示(presentation)与单个流(stream)。URL 可以用 于单个流或者多个流的集合，比如表示(presentation)。因此，在第十章所描述的请求(request)可以用于整个表示 (presentation)或表示中的单个流。注意，有些请求方法仅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 twister 表示(presentation)中，可以通过 media.example.com554 端口的 TCP 连接发送 RTSP 请求来控制的音频流。

也可以是这样 RTSP URL：
rtsp://media.example.com:554/twister
标识了由音频和视频流组成的 twister 表示(presentation)。

这并没有给出 URL 中相关流的标准方法。表示描述定义了表示中的层次关系以及单独流的 URL。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来说不可见并且也并没有给出服务器的具体文件系统结构。

只需进行简单替换后，表示描述同样可以用于非 RTSP 媒体控制协议。

3.3 会议标识 Conference Identifiers

会议标识采用 URI 标准编码方法编码，并对 RTSP 来说是不可见的。它们能包含任一八位位组值。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

会议标识允许 RTSP 会话从媒体服务器参与的多媒体会议中获取参数。比如，可以要求媒体服务器用会议描述中的标识值来代替 RTSP 客户端以提供详细的传输信息。多媒体会议的建立不属于本协议内容，具体请参见 H.323 或 SIP 协议。

3.4 会话标识 Session Identifiers

会话标识符是不可见的任意长度的字符串。线性空格必须是 URL-escaped。会话标识符必须随机产生并且至少应有 8 个八位位组长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 时间编码形式表示而可以达到帧级量级的精度。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，其帧速是 29.97 帧每秒。可通过选择使用不同“SMPTE time”来选择其他 SMPTE 编码格式（如“SMPTE 25”格式）。帧域的时间值在 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每第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"
; 还可以加入其他时间编码
smpte-time = 1*2DIGIT ":" 1*2DIGIT ":" 1*2DIGIT [":" 1*2DIGIT]
["." 1*2DIGIT]

比如：
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 Normal Play Time

正常播放时间(NPT)指出了流相对于表示(presentation)开始时的绝对位置。时间戳由一个十进制小数组成，以秒为单位，小数点左边可以直接以秒表示或者以小时：分：秒的形式表示。

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。它经常作为数字显示在 VCR 上。当处于普通播放模式 (scale = 1)时，NPT 正常前进。当处于快进扫描模式时(scale 率为大于 1 的正数)，NPT 快速前进。当处于反向扫描模式(scale 率小于-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

比如：
npt=123.45-125
npt=12:05:35.3-
npt=now-

语法遵循 ISO 8601 规则。npt-sec 标志法便于自动产生， ntp-hhmmss 标志法便于人工使用。“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，既没有绝对时间，也没有 0 时间，所以需要该参数。

3.7 绝对时间 Absolute Time

绝对时间表示为 ISO 8601 时间戳，使用 UTC(GMT)小数法表示。

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 Options Tags

选择标签是用来指定 RTSP 新选择的唯一标识符。这些标签用于要求(Require)(12.32 节)和代理要求(Proxy Require)(12.27 节)报头域中。

语法：
option-tag = 1*xchar

建立新的 RTSP 选择可以通过在选择前加入相反域名的前缀（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）新的选择。

3.8.1 用 IANA 注册新的选择标签 Registering New Option Tags With IANA

当注册新 RTSP 选择标签的时候，应该提供以下信息：
? 选择的名字和描述。名字长度不限，但是应该不少于 20 字符。名字不得包含任何空格，控制符或句点。
? 指出谁拥有选择的改变控制权（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，RFC，发表论文，专利文档，技术报告，源代码，或者计算机手册。
? 选择的所有权，以及联系地址（邮编及电子信件地址）。

4 RTSP 消息

RTSP 是基于文本的协议，采用 ISO 10646 字符集，使用 UTF-8 编码方案。行以 CRLF 中断，但接收者本身可将 CR 和 LF 解释成行终止符。基于文本的协议使以自描述方式增加可选参数更容易。由于参数的数量和命令的频率出现较低，处理效率没引起注意。如仔细研究，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 字符集避免敏感字符集切换，但对应用来说不可见。RTCP 也采用这种编码方案。带有重要意义位的 ISO 8859-1 字符表示如 100001x 10xxxxxx.。RTSP 信息可通过任何低层传输协议携带。

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。方法也可设计为在服务器端只需要少量或不需要状态维护。当信息体包含在信息中，信息体长度有如下因素决定：

不管实体报头域是否出现在信息中，不包括信息体的的回应信息总以报头域后第一和空行结束。

如出现内容长度报头域，其值以字节计，表示信息体长度。如未出现报头域，其值为零。

服务器关闭连接。

注意：RTSP 目前并不支持 HTTP/1.1\"块\"传输编码，需要有内容长度头。假如返回适度表示描述长度，即使动态产生，使块传输编码没有必要，服务器也应该能决定其长度。如有实体，即使必须有内容长度，且长度没显式给出，规则可确保行为合理。

从用户到服务器端的请求信息在第一行内包括源采用的方法、源标识和所用协议版本。RTSP 定义了附加状态代码，而没有定义任何 HTTP 代码。

4.1 消息类型 Message Type

见[H4.1]。如下：

RTSP 消息由客户端到服务器的请求和由服务器到客户端的回应组成。

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

请求（Request）和回应（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，一个或多个报头域（headers）、一行表示报头域结束的空行（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 消息报头 Message Headers

见[H4.2]。
RTSP 报头域，包括主报头（General-Header,4.3 节）、请求报头（Request-Header ,5.2 节）、回应报头（Response-Header ,6.2 节）及实体报头（Entity-Header,7.1 节），都遵照 RFC822-3.1 节[7]给出的通用格式定义。每个报头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，报头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>
报头域接收的顺序并不重要，但良好的习惯是，先发送主报头，然后是请求报头或回应报头，最后是实体报头。
当且仅当报头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 报头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，最终能将多个报头域结合成“域名：域值”对。

4.3 消息主体 Message Body

见[H4.3]。
RTSP 消息的消息主体（如果有）用来携带请求或回应的主体。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报头域中有详细说明。（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/回应链添加或者删除。

什么时候允许消息带消息体的规则在请求和回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报头域中的传输编码报头域。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 对于回应消息来说，无论消息中是否存在消息主体都与请求方法和回应状态编码无关。所有回应报头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 报头域而使人产生误解。所有 1××（信息），204（无内容），304（未修改）回应都不包含消息主体。而其他回应则都包含主体，尽管其长度有可能长度为零。

4.4 消息长度 Message Length

当消息包含消息主体时，消息主体的长度由以下规则来决定（按优先级高低顺序排列）：
1. 任何回应消息都不包含消息主体（如 1××，204 和 304 回应），并且不管消息中是否存在实体报头域都以消息报头域后的第一行空行表示结束。
2. 如果内容长度报头域存在，它在字节中的值就是消息主体的长度。如果内容报头域不存在，则假设值为零。
3. 服务器关闭连接时。（关闭连接没有用来表明请求主体结束，否则可能导致服务器不能回应。
注意，RTSP 不支持（至少现在）HTTP/1.1 的块传输编码（详见[H3.6]）并且要求有内容长度报头域。
尽管表示描述长度动态产生，但由于可获得了表示描述返回长度，使得服务器总是能决定表示描述长度而不需使用块传输编码方式。只要有实体主体就必须有内容长度项，这些规则保证了即使没有给出明确长度也能做出合理的操作。

5 普通报头域 General Header Fields

有几种报头域是请求与回应都要使用的，但并不用于被传输的实体。这些报头只用于被传输的消息。

General-Header = Date ; Section 10.6
| Pragma ; Section 10.12
普通报头域名称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。实际上，新的或实验中的报头域只要能被通讯各方识别，其语法就可使用，而无法识别的报头域都将被视为实体域。

6 请求 Request

从客户端到服务器端的请求消息包括，消息首行中，对资源的请求方法、资源的标识符及使用的协议。

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 请求队列 Request Line

Request-Line = Method SP Request-URI SP RTSP-Version 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 请求报头域 Request Header Fields

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 而言，RTSP 请求要求绝对路径（并包括 rtsp 或 rtspu 方案，主机，端口号）。

HTTP/1.1 要求服务器理解绝对 URL，但是客户端需要假设为主机请求报头域。这样做完全是为了 HTTP/1.0 服务器端向后兼容性，因此 RTSP 并不需要这样做。

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，并且它只能在使用的方法不要求应用于资源时才能使用。

比如：OPTIONS * RTSP/1.0。

7 回应 Response

7.1 状态行 Status-Line

完整回应消息的第一行就是状态行，它依次由协议版本、数字形式的状态代码、及相应的词语文本组成，各元素间以空格（SP）分隔，除了结尾的 CRLF 外，不允许出现单独的 CR 或 LF 符。

Status-Line = HTTP-Version SP **Status-Code** SP **Reason-Phrase** CRLF

7.1.1 状态代码和原因分析 Status Code and Reason Phrase

状态代码（Status-Code）由 3 位数字组成，表示请求是否被理解或被满足。原因分析是用简短的文字来描述状态代码产生的原因。状态代码用来支持自动操作，原因分析是为人类用户准备的。客户端不需要检查或显示原因分析。状态代码的第一位数字定义了回应的类别，后面两位数字没有具体分类。首位数字有 5 种取值可能：
o 1xx:：保留，将来使用。

o 2xx：成功 — 操作被接收、理解、接受（received, understood, accepted）。

o 3xx：重定向（Redirection）— 要完成请求必须进行进一步操作。

o 4xx：客户端出错 — 请求有语法错误或无法实现。

o 5xx：服务器端出错 — 服务器无法实现合法的请求。

HTTP/1.0 的状态代码、原因解释在下面给出。下面的原因解释只是建议采用，可任意更改，而不会对协议造成影响。完整的代码定义在第 9 节。

Status-Code = "100" ; Continue
| "200" ; OK
| "201" ; Created
| "250" ; Low on Storage Space
| "300" ; Multiple Choices
| "301" ; Moved Permanently
| "302" ; Moved Temporarily
| "303" ; See Other
| "304" ; Not Modified
| "305" ; Use Proxy
| "400" ; Bad Request
| "401" ; Unauthorized

| "402" ; Payment Required
| "403" ; Forbidden
| "404" ; Not Found
| "405" ; Method Not Allowed
| "406" ; Not Acceptable
| "407" ; Proxy Authentication Required
| "408" ; Request Time-out
| "410" ; Gone
| "411" ; Length Required
| "412" ; Precondition Failed
| "413" ; Request Entity Too Large
| "414" ; Request-URI Too Large
| "415" ; Unsupported Media Type
| "451" ; Parameter Not Understood
| "452" ; Conference Not Found
| "453" ; Not Enough Bandwidth
| "454" ; Session Not Found
| "455" ; Method Not Valid in This State
| "456" ; Header Field Not Valid for Resource
| "457" ; Invalid Range
| "458" ; Parameter Is Read-Only
| "459" ; Aggregate operation not allowed
| "460" ; Only aggregate operation allowed
| "461" ; Unsupported transport
| "462" ; Destination unreachable
| "500" ; Internal Server Error
| "501" ; Not Implemented
| "502" ; Bad Gateway
| "503" ; Service Unavailable
| "504" ; Gateway Time-out
| "505" ; RTSP Version not supported
| "551" ; Option not supported
| extension-code
extension-code = 3DIGIT
Reason-Phrase = *<TEXT, excluding CR, LF>

HTTP 状态代码是可扩展的，而只有上述代码才可以被当前全部的应用所识别。HTTP 应用不要求了解全部注册的状态代码，当然，如果了解了更好。实际上，应用程序必须理解任何一种状态代码，如果碰到不识别的情况，可根据其首位数字来判断其类型并处理。另外，不要缓存无法识别的回应。

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，可认为回应的状态码就是 400。在这种情况下，用户代理应当在回应消息的实体中通知用户，因为实体中可以包括一些人类可以识别的非正常状态的描述信息。

Code reason
100 Continue all
200 OK all
201 Created RECORD
250 Low on Storage Space RECORD
300 Multiple Choices all
301 Moved Permanently all
302 Moved Temporarily all
303 See Other all
305 Use Proxy all
400 Bad Request all
401 Unauthorized all
402 Payment Required all
403 Forbidden all
404 Not Found all
405 Method Not Allowed all
406 Not Acceptable all
407 Proxy Authentication Required all
408 Request Timeout all
410 Gone all
411 Length Required all
412 Precondition Failed DESCRIBE, SETUP
413 Request Entity Too Large all
414 Request-URI Too Long all
415 Unsupported Media Type all
451 Invalid parameter SETUP
452 Illegal Conference Identifier SETUP
453 Not Enough Bandwidth SETUP
454 Session Not Found all
455 Method Not Valid In This State all
456 Header Field Not Valid all
457 Invalid Range PLAY
458 Parameter Is Read-Only SET_PARAMETER
459 Aggregate Operation Not Allowed all
460 Only Aggregate Operation Allowed all
461 Unsupported Transport all
462 Destination Unreachable all
500 Internal Server Error all
501 Not Implemented all
502 Bad Gateway all
503 Service Unavailable all
504 Gateway Timeout all
505 RTSP Version Not Supported all

551 Option not support all
Table 1: Status codes and their usage with RTSP methods

7.1.2 回应报头域 Response Header Fields

回应报头域中包括不能放在状态行中的附加回应信息。该域还可以存放与服务器相关的信息，以及在对请求 URI 所指定资源进行访问的下一步信息。

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

回应报头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。实际上，新的或实验中的报头域只要能被通讯各方识别，其语法就可使用，而无法识别的报头域都将被视为实体域。

8 实体 Entity

如不受请求方法或回应状态编码限制，请求和回应消息可传输实体，实体由实体报头域和实体主体组成，有些回应仅包括实体头。在此，根据谁发送实体、谁接收实体，发送者和接收者可分别指用户和服务器。

8.1 实体报头域 Entity Header Fields

实体报头定义实体主体可选元信息，如没有实体主体，指请求标识的资源。

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

扩展头机制允许定义附加实体报头域，而不用改变协议，但这些段不能假定接收者能识别。不可识别报头域应被接收者忽略，而让代理转发。

8.2 实体主体 Entity Body

见[H7.2]

与 RTSP 请求或回应一起发送的实体主体的格式和编码信息都在实体报头域（Entity-Header）中定义。

Entity-Body = *OCTET

实体主体只在请求方法有要求时才会被放在请求消息中。请求消息报头域处的内容长度报头域（Content-Length header field）的标志将指明请求中的实体主体是否存在。包含实体主体的 RTSP/1.0 请求必须包含合法的内容长度报头域。

对回应消息来说，消息中是否包含实体主体取决于请求方法和回应代码。所有的 HEAD 请求方法的回应都不应包括主体，即便是实体报头域中指明有主体也一样。在主体中不应包括这些回应信息，全部 1xx（信息）、204（无内容）和 304（未修改）。而其它的回应必须包括实体主体或其内容长度报头（Content-Length header）域的定义值为 0。

9 连接 Connections

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/回应传输。
2、每个请求/回应传输一个连接。
3、无连接模式。

传输连接类型由 RTSP URI 来定义。对 \"rtsp\" 方案，需要持续连接；而\"rtspu\"方案，调用 RTSP 请求发送，而不用建立连接。

不象 HTTP，RTSP 允许媒体服务器给媒体用户发送请求。然而，这仅在持久连接时才支持，否则媒体服务器没有可靠途径到达用户，这也是请求通过防火墙从媒体服务器传到用户的唯一途径。

9.1 流水线操作 Pipelining

支持持久连接或无连接的客户端可能给其请求排队。服务器必须以收到请求的同样顺序发出回应。

9.2 可靠性及确认 Reliability and Acknowledgements

如果请求不是发送给组播组，接收者就确认请求，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。应用缓存最后所测量的 RTT，作为将来连接的初始值。如使用一个可靠传输协议传输 RTSP，请求不允许重发，RTSP 应用反过来依赖低层传输提供可靠性。
如两个低层可靠传输（如 TCP 和 RTSP）应用重发请求，有可能每个包损失导致两次重传。由于传输栈在第一次尝试到达接收着者前不会发送应用层重传，接收者也不能充分利用应用层重传。如包损失由阻塞引起，不同层的重发将使阻塞进一步恶化。
时标头用来避免重发模糊性问题，避免对圆锥算法的依赖。
每个请求在 CSeq 头中携带一个系列号，每发送一个不同请求，它就加一。如由于没有确认而重发请求，请求必须携带初始系列号。

实现 RTSP 的系统必须支持通过 TCP 传输 RTSP ，并支持 UDP。对 UDP 和 TCP，RTSP 服务器的缺省端口都是 554。许多目的一致的 RTSP 包被打包成单个低层 PDU 或 TCP 流。RTSP 数据可与 RTP 和 RTCP 包交叉。不象 HTTP，RTSP 信息必须包含一个内容长度头，无论信息何时包含负载。否则，RTSP 包以空行结束，后跟最后一个信息头。

2009-12-25 Add By Gao.wenjie

10 方法定义 Method Definitions

The method token indicates the method to be performed on the resource identified by the Request-URI. The method is case-sensitive. New methods may be defined in the future. Method names may not start with a $ character (decimal 24) and must be a token. Methods are summarized in Table 2.
方法是指在请求 URI 中定义的、在资源中被执行的动作。方法是大小写敏感的。新的方法可能在将来被定义。方法的命名不能以$字符开头而且必须是一个符号，表 2 总结了常用的方法命令字。

method	direction	object	requirement
DESCRIBE	C->S	P,S	recommended
ANNOUNCE	C->S, S->C	P,S	optional
GET_PARAMETER	C->S, S->C	P,S	optional
OPTIONS	C->S, S->C	P,S	required (S->C: optional)
PAUSE	C->S	P,S	recommended

PLAY	C->S	P,S	required
RECORD	C->S	P,S	optional
REDIRECT	S->C	P,S	optional
SETUP	C->S	S	required
SET_PARAMETER	C->S, S->C	P,S	optional
TEARDOWN	C->S	P,S	required

Table 2: Overview of RTSP methods, their direction, and what objects (P: presentation, S: stream) they operate on Notes on Table 2: PAUSE is recommended, but not required in that a fully functional server can be built that does not support this method, for example, for live feeds. If a server does not support a particular method, it MUST return "501 Not Implemented" and a client SHOULD not try this method again for this server.

表 2：对于 RTSP 的方法来说，他们的流向和他们的操作目的（P：描述 S：流）已经在表 2 中标记出来：PAUSE 命令被推荐使用，但是在一台不支持该命令的服务器被建立时，该命令就不是必须的，比如，现场直播服务器。如果一个服务器不支持一个特殊的命令，它将返回“501 没有被执行”错误，同时客户端将不再尝试向此服务器发送同样的命令。

10.1 选择 OPTIONS：

The behavior is equivalent to that described in [H9.2]. An OPTIONS request may be issued at any time, e.g., if the client is about to try a nonstandard request. It does not influence server state.

该命令字的行为与[H9.2]中描述的相同。一个 OPTIONS 请求可以在任何时间发出，比如一个客户端尝试发出一个非标准请求。OPTIONS 命令不影响服务器状态。

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

Note that these are necessarily fictional features (one would hope that we would not purposefully overlook a truly useful feature just so that we could have a strong example in this section).

注意：这些都是必要的假设条件（在这一部分我们有意忽略一些真实有用的特征以使我们得到一个更有针对性的例子）

10.2 描述 DESCRIBE

The DESCRIBE method retrieves the description of a presentation or media object identified by the request URL from a server. It may use the Accept header to specify the description formats that the client understands. The server responds with a description of the requested resource. The DESCRIBE reply-response pair constitutes the media initialization phase of RTSP.

DESCRIBE 方法通过请求 URL 请求从服务器检索一个表示的描述或者媒体目标。DESCRIBE 请求通常使用 Accept 报头来说明可以被客户端理解的描述的格式。服务器选择其中的一种描述应答给客户端。一对 DESCRIBE 响应-应答组成 RTSP 命令的初始化过程。

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

10.3 通告 ANNOUNCE

The ANNOUNCE method serves two purposes:

When sent from client to server, ANNOUNCE posts the description of a presentation or media object identified by the request URL to a server. When sent from server to client, ANNOUNCE updates the session description in real-time.

If a new media stream is added to a presentation (e.g., during a live presentation), the whole presentation description should be sent again, rather than just the additional components, so that components can be deleted.

ANNOUNCE 方法有两种用途：

当从客户端发往服务器时，ANNOUNCE 方法向服务器发送一个表示（presentation ）的描述或者一个通过请求 RUL 识别的媒体目标。当从服务器发往客户端时，ANNOUNCE 方法实时更新一个会话的描述。

如果一个新的媒体流被加入进一个表示（比如，在现场直播期间），整个表示描述宁愿被重新发送，也不会只是增加额外的组成成分，所以那些组成可以被删除。

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

The SETUP request for a URI specifies the transport mechanism to be used for the streamed media. A client can issue a SETUP request for a stream that is already playing to change transport parameters, which a server MAY allow. If it does not allow this, it MUST respond with error "455 Method Not Valid In This State".

For the benefit of any intervening firewalls, a client must indicate the transport parameters even if it has no influence over these parameters, for example, where the server advertises a fixed multicast address.

Since SETUP includes all transport initialization information, firewalls and other intermediate network devices (which need this information) are spared the more arduous task of parsing the DESCRIBE response, which has been reserved for media initialization.

The Transport header specifies the transport parameters acceptable to the client for data transmission; the response will contain the transport parameters selected by the server.

SETUP 方法请求一个指定了被流媒体使用的传输机制的 URI。客户端可以为一个已经播放的流发送 SETUP 方法来改变传输机制，服务器可以同意。如果该请求没有被同意，服务器必须回应一个"455 该状态下方法无效"的错误。

为了适应各种防火墙，一个客户端必须指明传输参数，即使防火墙还没有对这些参数产生影响，例如，广播一个固定的组播地址的服务器地址。

既然 SETUP 方法包含了所有的传输初始化信息，防火墙和其他的网络中介设备就可以省略复杂的 DESCRIBE 应答的解析任务，解析任务由媒体的初始化负责。

传输头定义了客户端为了适应数据发送而定义的传输参数；应答中将包含服务器选择出来的传输参数。

例子：

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

The server generates session identifiers in response to SETUP requests. If a SETUP request to a server includes a session identifier, the server MUST bundle this setup request into the existing session or return error "459 Aggregate Operation Not Allowed" (see Section 11.3.10).

服务器为应答 SETUP 请求而产生一个会话（session）标示符。如果一个发送给服务器的 SETUP 请求中带有一个会话标示符，服务器必须将这个 setup 请求打包进已经存在的会话中，或者应答一个"459 Aggregate Operation Not Allowed" 错误。

10.5 播放 PLAY

The PLAY method tells the server to start sending data via the mechanism specified in SETUP. A client MUST NOT issue a PLAY request until any outstanding SETUP requests have been acknowledged as successful.

PLAY 方法告诉服务器使用 SETUP 方法中定义的传输机制开始发送数据。客户端在 SETUP 请求被成功应答之前，禁止发送 PLAY 请求。

The PLAY request positions the normal play time to the beginning of the range specified and delivers stream data until the end of the range is reached. PLAY requests may be pipelined (queued); a server MUST queue PLAY requests to be executed

in order. That is, a PLAY request arriving while a previous PLAY request is still active is delayed until the first has been completed. This allows precise editing.

PLAY 请求定位到正常播放的指定范围的开始位置，然后开始传输流数据，直到结束。

PLAY 请求可能是管道；服务器必须将 PLAY 请求放入命令执行队列。也就是说，当前一个 PLAY 请求依然有效时，再次接收到的 PLAY 请求是将被延时到第一个 PLAY 命令完成。这需要精确编辑。

For example, regardless of how closely spaced the two PLAY requests in the example below arrive, the server will first play seconds 10 through 15, then, immediately following, seconds 20 to 25, and finally seconds 30 through the end.

例如，不管第二个 PLAY 命令到来的多快，服务器都将播放 10 到 15 秒，然后，立即播放后面的 20~25 秒，最后是从 30 秒直到最后。

```
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-
```

See the description of the PAUSE request for further examples.

更多的描述见 PAUSE 请求的例子。

A PLAY request without a Range header is legal. It starts playing a stream from the beginning unless the stream has been paused. If a stream has been paused via PAUSE, stream delivery resumes at the pause point. If a stream is playing, such a PLAY request causes no further action and can be used by the client to test server liveness.

一个没有 Range 报头的 PLAY 请求也是合法的。如果流不是被暂停的，则从起始位置开始播放。如果一个流已经通过 PAUSE 方法暂停了，则流从暂停的位置重新开始。如果一个流正在播放，那么 PLAY 请求不会产生任何动作，因此也经常被客户端用来测试服务器是否还在活动。

The Range header may also contain a time parameter. This parameter specifies a time in UTC at which the playback should start. If the message is received after the specified time, playback is started immediately. The time parameter may be used to aid in synchronization of streams obtained from different sources.

Range 报头也可以包含一个时间参数。这个时间参数以 UTC 的方式定义了回放的起始位置。如果接收到消息时在指定的时间已经过去，就会立即开始回放。时间参数经常用来同步从不同源所获取的流。

For a on-demand stream, the server replies with the actual range that will be played back. This may differ from the requested range if alignment of the requested range to valid frame boundaries is required for the media source. If no range is specified in the request, the current position is returned in the reply. The unit of the range in the reply is the same as that in the request.

对于实时数据流，服务器应答将要播放的实际值。如果请求有效帧边界范围的队列对媒体源是必须的，则实时数据流与请求队列是不同的。如果在请求中没有指定范围，应答将返回当前位置。应答中的范围单位与请求中的单位相同。

After playing the desired range, the presentation is automatically paused, as if a PAUSE request had been issued.

播放完要求播放的范围之后，presentation 自动暂停，就好像 PAUSE 方法被发送一样。

The following example plays the whole presentation starting at SMPTE time code 0:10:20 until the end of the clip. The playback is to start at 15:36 on 23 Jan 1997.

下面的例子以 SMPTE 时间方式在 0：10：20 播放整个 presentation，直到片段结束。回放在 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

For playing back a recording of a live presentation, it may be desirable to use clock units:

对于现场直播，使用时间为单位可能更合适一些。。

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

A media server only supporting playback MUST support the npt format and MAY support the clock and smpte formats.

支持回放的媒体服务器必须支持 NPT 格式同时应该支持时钟和 SMPTE 格式。

10.6 暂停 PAUSE

The PAUSE request causes the stream delivery to be interrupted (halted) temporarily. If the request URL names a stream, only playback and recording of that stream is halted. For example, for audio, this is equivalent to muting. If the request URL names a presentation or group of streams, delivery of all currently active streams within the presentation or group is halted. After resuming playback or recording, synchronization of the tracks MUST be maintained. Any server resources are kept, though servers MAY close the session and free resources after being paused for the duration specified with the timeout parameter of the Session header in the SETUP message.

PAUSE 请求造成正在传输的流被临时停止。如果请求 URL 指定了一个流，则只有该流的回放和录像暂停。例如，对于音频来讲，暂停动作就等同于静音。如果请求 URL 指定了一个

描述或者一组流，则在描述或者组之内的活动流都将被暂停。在回放或者录像重新开始之后，同步点必须被保留。虽然服务器在暂停超时之后可以关闭 session 和释放资源，但是每一个服务器资源都将被保留。该超时参数在 SETUP 消息的 Session 头中定义。

Example:

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

The PAUSE request may contain a Range header specifying when the stream or presentation is to be halted. We refer to this point as the "pause point". The header must contain exactly one value rather than a time range. The normal play time for the stream is set to the pause point.

PAUSE 请求可能包含一个指定流或者表示在什么时间被暂停的 Range 报头。我们称这个点为“暂停点”。报头中必须包含一个正确的值而不是时间范围。流的正常播放时间被设定给暂停点。

The pause request becomes effective the first time the server is encountering the time point specified in any of the currently pending PLAY requests. If the Range header specifies a time outside any currently pending PLAY requests, the error "457 Invalid Range" is returned. *If a media unit (such as an audio or video frame) starts presentation at exactly the pause point, it is not played or recorded.* If the Range header is missing, stream delivery is interrupted immediately on receipt of the message and the pause point is set to the current normal play time.

暂停请求第一次有效服务器正遇到指定的时间点在任何一个当前挂起的 PLAY 请求。

如果 Range 报头指定的时间超出了任何一个当前挂起的 Play 请求，就会返回一个“457 非法范围”的错误。*如果一个媒体单位（例如一个音频或者视频帧）在实际的暂停点开始演出，它不被播放或者录像。*如果收到一个没有 Range 报头的消息，流的传输立即被中断，同时暂停点被设置为正常播放时的当前时间。

A PAUSE request discards all queued PLAY requests. However, the pause point in the media stream MUST be maintained. A subsequent PLAY request without Range header resumes from the pause point.

PAUSE 请求放弃全部的 PLAY 请求队列。可是，媒体流中的暂停点必需被维护。因为暂停之后，在接收到一个没有范围报头的 Play 请求时将从暂停点重新开始。

For example, if the server has play requests for ranges 10 to 15 and 20 to 29 pending and then receives a pause request for NPT 21, it would start playing the second range and stop at NPT 21. If the pause request is for NPT 12 and the server is playing at NPT 13 serving the first play request, the server stops immediately. If the pause request is for NPT 16, the server stops after completing the first play request and discards the second play request.

例如，如果服务器已经在播放 10--15,20-29 正被挂起，这时接收到一个要求在 21 暂停的请求，这将开始播放第二个范围并在 NPT 21 停止。如果是要求在 NPT12 暂停，而此时服务器正在播放第一个 play 请求的 NPT13 位置，则服务器立即停止。如果要求在 NPT16 暂停，则服务器在播放完第一个 Play 请求之后暂停，并放弃第二个 Play 请求。

As another example, if a server has received requests to play ranges 10 to 15 and then 13 to 20 (that is, overlapping ranges), the PAUSE request for NPT=14 would take effect while the server plays the first range, with the second PLAY request effectively being ignored, assuming the PAUSE request arrives before the server has started playing the second, overlapping range. Regardless of when the PAUSE request arrives, it sets the NPT to 14. If the server has already sent data beyond the time specified in the Range header, a PLAY would still resume at that point in time, as it is assumed that the client has discarded data after that point. This ensures continuous pause/play cycling without gaps.

10.7 断开 TEARDOWN

The TEARDOWN request stops the stream delivery for the given URI, freeing the resources associated with it. If the URI is the presentation URI for this presentation, any RTSP session identifier associated with the session is no longer valid. Unless all transport parameters are defined by the session description, a SETUP request has to be issued before the session can be played again.

TEARDOWN 方法根据给定的 URI 资源将流传输停止，同时释放这些资源。如果 URI 是一个针对该描述的 URI 描述，任何一个带有该 session 的 RTSP 描述符都不在有效。除非所有的传输参数在 session 描述中重新定义，一个 SETUP 请求在 session 重新播放之前被再次发送。

Example:

```
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

The GET_PARAMETER request retrieves the value of a parameter of a presentation or stream specified in the URI. The content of the reply and response is left to the implementation. GET_PARAMETER with no entity body may be used to test client or server liveness ("ping").

Example:

```
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

The "text/parameters" section is only an example type for parameter. This method is intentionally loosely defined with the intention that the reply content and response content will be defined after further experimentation.

10.9 设置参数 SET_PARAMETER

This method requests to set the value of a parameter for a presentation or stream specified by the URI.

A request SHOULD only contain a single parameter to allow the client to determine why a particular request failed. If the request contains several parameters, the server MUST only act on the request if all of the parameters can be set successfully. A server MUST allow a parameter to be set repeatedly to the same value, but it MAY disallow changing parameter values.

Note: transport parameters for the media stream MUST only be set with the SETUP command.

Restricting setting transport parameters to SETUP is for the benefit of firewalls.

The parameters are split in a fine-grained fashion so that there can be more meaningful error indications. However, it may make sense to allow the setting of several parameters if an atomic setting is desirable. Imagine device control where the client does not want the camera to pan unless it can also tilt to the right angle at the same time.

Example:

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
```

The "text/parameters" section is only an example type for parameter. This method is intentionally loosely defined with the intention that the reply content and response content will be defined after further experimentation.

10.10 重定向 REDIRECT

A redirect request informs the client that it must connect to another server location. It contains the mandatory header Location, which indicates that the client should issue requests for that URL. It may contain the parameter Range, which indicates when the redirection takes effect. If the client wants to continue to send or receive media for this URI, the client MUST issue a TEARDOWN request for the current session and a SETUP for the new session at the designated host.

This example request redirects traffic for this URI to the new server at the given play time:

```
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

This method initiates recording a range of media data according to the presentation description. The timestamp reflects start and end time (UTC). If no time range is given, use the start or end time provided in the presentation description. If the session has already started, commence recording immediately.

The server decides whether to store the recorded data under the request-URI or another URI. If the server does not use the request-URI, the response SHOULD be 201 (Created) and contain an entity which describes the status of the request and refers to the new resource, and a Location header.

A media server supporting recording of live presentations MUST support the clock range format; the smpte format does not make sense.

In this example, the media server was previously invited to the conference indicated.

```
C->S: RECORD rtsp://example.com/meeting/audio.en RTSP/1.0
      CSeq: 954
      Session: 12345678
      Conference: 128.16.64.19/32492374
```

10.12 嵌入二进制数据 Embedded(interleaved) Binary Data

Certain firewall designs and other circumstances may force a server to interleave RTSP methods and stream data. This interleaving should generally be avoided unless necessary since it complicates client and server operation and imposes additional overhead. Interleaved binary data SHOULD only be used if RTSP is carried over TCP.

Stream data such as RTP packets is encapsulated by an ASCII dollar sign (24 hexadecimal), followed by a one-byte channel identifier, followed by the length of the encapsulated binary data as a binary, two-byte integer in network byte order. The stream data follows immediately afterwards, without a CRLF, but including the

upper-layer protocol headers. Each $ block contains exactly one upper-layer protocol data unit, e.g., one RTP packet.

The channel identifier is defined in the Transport header with the interleaved parameter(Section 12.39).

When the transport choice is RTP, RTCP messages are also interleaved by the server over the TCP connection. As a default, RTCP packets are sent on the first available channel higher than the RTP channel. The client MAY explicitly request RTCP packets on another channel. This is done by specifying two channels in the interleaved parameter of the Transport header(Section 12.39).

RTCP is needed for synchronization when two or more streams are interleaved in such a fashion. Also, this provides a convenient way to tunnel RTP/RTCP packets through the TCP control connection when required by the network configuration and transfer them onto UDP when possible.

```
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 状态代码定义（Status Code Definitions）

Where applicable, HTTP status [H10] codes are reused. Status codes that have the same meaning are not repeated here. See Table 1 for a listing of which status codes may be returned by which requests.

在合适的地方，HTTP 状态码被重用。含义相同的状态码没有被重复定义。表一列出了哪一个请求返回哪一个相应的状态码。

11.1 成功 2xx（Success 2xx）

11.1.1 存储空间低 250　Low on Storage Space

The server returns this warning after receiving a RECORD request that it may not be able to fulfill completely due to insufficient storage space. If possible, the server should use the Range header to indicate what time period it may still be able to record. Since other processes on the server may be consuming storage space simultaneously, a client should take this only as an estimate.

服务器接收到 RECORD 请求并且发现没有足够的存储空间完成该命令时返回该警告。如果可能，服务器应该使用 Range 头指出录像的是周期。因为服务器上的其他的进程可能也在同时录像，因此客户端应该把该周期仅仅作为一个参考。

11.2 重定向（Redirection 3xx）

See [H10.3].

Within RTSP, redirection may be used for load balancing or redirecting stream requests to a server topologically closer to the client. Mechanisms to determine topological proximity are beyond the scope of this specification.

11.3 客户端错误（Client Error ）4xx

11.3.1 方法不允许 (405 Method Not Allowed)

The method specified in the request is not allowed for the resource identified by the request URI. The response MUST include an Allow header containing a list of valid methods for the requested resource. This status code is also to be used if a request attempts to use a method not indicated during SETUP, e.g., if a RECORD request is issued even though the mode parameter in the Transport header only specified PLAY.

请求 URI 中的资源标示不允许请求中定义的方法。应答必须包含一个对于请求资源有效的方法列表的头区域。这个状态码也经常表示一个请求尝试使用一个没有在 SETUP 中指明的方法，例如，在参数头中只定义了 PLAY 参数却发出了一个 RECORD 请求。

11.3.2 参数不能理解 (451 Parameter Not Understood)

The recipient of the request does not support one or more parameters contained in the request.

请求的接收者不支持请求中包含的一个或者多个参数。

11.3.3 会议未找到 (452 Conference Not Found)

The conference indicated by a Conference header field is unknown to the media server.

Conference 头中定义的会议对于媒体服务是未知的。

11.3.4 带宽不足 (453 Not Enough Bandwidth)

The request was refused because there was insufficient bandwidth.This may, for example, be the result of a resource reservation failure.
请求因为带宽不足被拒绝。

11.3.5 会话未找到 (454 Session Not Found)

The RTSP session identifier in the Session header is missing,invalid, or has timed out.
在会话头区域定义的 RTSP 会话标识没有找到，或者是无效的，或者是超时。

11.3.6 本状态下该方法无效 (455 Method Not Valid in This State)

The client or server cannot process this request in its current state. The response SHOULD contain an Allow header to make error recovery easier.
客户端或者服务器无法在当前状态下执行该请求。应答应该包含一个容易使错误恢复的 Allow 头区域。

11.3.7 报头域对资源无效(456 Header Field Not Valid for Resource)

The server could not act on a required request header. For example,if PLAY contains the Range header field but the stream does not allow seeking.
服务器对请求报头无效。例如，一个 PLAY 方法包含一个 Range 报头域但是流不允许定位操作。

11.3.8 无效范围 (457 Invalid Range)

The Range value given is out of bounds, e.g., beyond the end of the presentation.
给出的范围值超出界限，比如，超过了 presentation 的结尾。

11.3.9 参数只读 (458 Parameter Is Read-Only)

The parameter to be set by SET_PARAMETER can be read but not modified.
SET_PARAMETER 方法设定的参数只能读，无法修改。

11.3.10 不允许合操作(459 Aggregate Operation Not Allowed)

The requested method may not be applied on the URL in question since it is an aggregate (presentation) URL. The method may be applied on a stream URL.

11.3.11 只允许合操作 (460 Only Aggregate Operation Allowed)

The requested method may not be applied on the URL in question since it is not an aggregate (presentation) URL. The method may be applied on the presentation URL.

11.3.12 不支持的传输 (461 Unsupported Transport)

The Transport field did not contain a supported transport specification.

11.3.13 目标不可达 (462 Destination Unreachable)

The data transmission channel could not be established because the client address could not be reached. This error will most likely be the result of a client attempt to place an invalid Destination parameter in the Transport field.

11.3.14 选择不被支持 (551 Option not Supported)

An option given in the Require or the Proxy-Require fields was not supported. The Unsupported header should be returned stating the option for which there is no support.

12 报头域定义（Header Field Definitions）

HTTP/1.1 [2] or other, non-standard header fields not listed here currently have no well-defined meaning and SHOULD be ignored by the recipient.

Table 3 summarizes the header fields used by RTSP. Type "g" designates general request headers to be found in both requests and responses, type "R" designates request headers, type "r" designates response headers, and type "e" designates entity header fields.
表三是 RTSP 报头域的摘要。‘g’表示可以在请求和应答中使用的通用请求报头。‘R’表示请求报头，‘r’表示应答报头，‘e’表示实体报头域。

Fields marked with "req." in the column labeled "support" MUST be implemented by the recipient for a particular method, while fields marked "opt." are optional. Note that not all fields marked "req." will be sent in every request of this type. The "req." means only that client (for response headers) and server (for request headers) MUST implement the fields. The last column lists the method for which this header field is meaningful; the designation "entity" refers to all methods that return a message body. Within this specification, DESCRIBE and GET_PARAMETER fall into this class.
Support 列中标记的‘reg’标志必须接收者必须执行，而‘opt.’标记是可选择的。注意：不是所有标记有‘reg.’的报头域在每一个请求中都必须被发送。‘reg.’标志的含义是只有客户端和服务器必须执行的域。最后一列列出这些报头域的目的。‘entity’代表所有返回方法的消息实体。

Header	type	support	methods
Accept	R	opt.	entity
Accept-Encoding	R	opt.	entity

Accept-Language	R	opt.	all
Allow	r	opt.	all
Authorization	R	opt.	all
Bandwidth	R	opt.	all
Blocksize	R	opt.	all but OPTIONS, TEARDOWN
Cache-Control	g	opt.	SETUP
Conference	R	opt.	SETUP
Connection	g	req.	all
Content-Base	e	opt.	entity
Content-Encoding	e	req.	SET_PARAMETER
Content-Encoding	e	req.	DESCRIBE, ANNOUNCE
Content-Language	e	req.	DESCRIBE, ANNOUNCE
Content-Length	e	req.	SET_PARAMETER, ANNOUNCE
Content-Length	e	req.	entity
Content-Location	e	opt.	entity
Content-Type	e	req.	SET_PARAMETER, ANNOUNCE
Content-Type	r	req.	entity
CSeq	g	req.	all
Date	g	opt.	all
Expires	e	opt.	DESCRIBE, ANNOUNCE
From	R	opt.	all
If-Modified-Since	R	opt.	DESCRIBE, SETUP
Last-Modified	e	opt.	entity
Proxy-Authenticate			
Proxy-Require	R	req.	all
Public	r	opt.	all
Range	R	opt.	PLAY, PAUSE, RECORD
Range	r	opt.	PLAY, PAUSE, RECORD
Referer	R	opt.	all
Require	R	req.	all
Retry-After	r	opt.	all
RTP-Info	r	req.	PLAY
Scale	Rr	opt.	PLAY, RECORD
Session	Rr	req.	all but SETUP, OPTIONS
Server	r	opt.	all

Speed	Rr	opt.	PLAY
Transport	Rr	req.	SETUP
Unsupported	r	req.	all
User-Agent	R	opt.	all
Via	g	opt.	all
WWW-Authenticate	r	opt.	all

Overview of RTSP header fields

12.1 接受 Accept

The Accept request-header field can be used to specify certain presentation description content types which are acceptable for the response.

Accept 请求报文头经常被用来指定可以被应答接受的明确的表示描述内容类型

The "level" parameter for presentation descriptions is properly defined as part of the MIME type registration, not here. See [H14.1] for syntax.

表示描述中的 Level 参数作为 MIME 注册类型的一部分的适当补充。

Example of use:

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

12.2 接受编码 Accept-Encoding

See [H14.3]

12.3 接受语言 Accept-Langue

See [H14.4]. Note that the language specified applies to the presentation description and any reason phrases, not the media content.

指定的语言适用于表示描述和任何一个短语，而不是媒体目录。

12.4 允许（Allow）

The Allow response header field lists the methods supported by the resource identified by the request-URI. The purpose of this field is to strictly inform the recipient of valid methods associated with the resource. An Allow header field must be present in a 405 (Method not allowed) response.

Example of use:
Allow: SETUP, PLAY, RECORD, SET_PARAMETER
Allow 的应答报头域列出了可以被请求 URI 中的资源标识所支持的方法。这个域的作用就是明确的告诉接收者资源中可用的方法集。一个 Allow 报头域必须在 405 应答中存在。

12.5 授权（Authorization）

See [H14.8]

12.6 带宽 Bandwidth

The Bandwidth request header field describes the estimated bandwidth available to the client, expressed as a positive integer and measured in bits per second. The bandwidth available to the client may change during an RTSP session, e.g., due to modem retraining.

Bandwidth 请求报头域说明对客户端可用的可能的带宽，用一个正整数和标准的 bit/秒表示.客户端可用的带宽可以通过 RTSP 会话修改，比如，在调制解调器重连期间。

Bandwidth = "Bandwidth" ":" 1*DIGIT

Example:

Bandwidth: 4000

12.7 块大小 Blocksize

This request header field is sent from the client to the media server asking the server for a particular media packet size. This packet size does not include lower-layer headers such as IP, UDP, or RTP. The server is free to use a blocksize which is lower than the one requested. The server MAY truncate this packet size to the closest multiple of the minimum, media-specific block size, or override it with

the media-specific size if necessary. The block size MUST be a positive decimal number, measured in octets. The server only returns an error (416) if the value is syntactically invalid.

这个请求报头域由客户端发送给媒体服务器的，想服务器询问具体的媒体数据包的大小。这个包大小不包括底层协议的头，比如 IP，UDP 或者 RTP。服务器可以自由使用低于一个请求的块大小。

12.8 缓存控制 Cache-Control

The Cache-Control general header field is used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain.

Cache directives must be passed through by a proxy or gateway application, regardless of their significance to that application, since the directives may be applicable to all recipients along the request/response chain. It is not possible to specify a cache-directive for a specific cache.

Cache-Control should only be specified in a SETUP request and its response. Note: Cache-Control does not govern the caching of responses as for HTTP, but rather of the stream identified by the SETUP request. Responses to RTSP requests are not cacheable, except for responses to DESCRIBE.

```
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 ) ]
```

no-cache:

Indicates that the media stream MUST NOT be cached anywhere. This allows an origin server to prevent caching even by caches that have been configured to return stale responses to client requests.

public:

Indicates that the media stream is cacheable by any cache.

private:

Indicates that the media stream is intended for a single user and MUST NOT be cached by a shared cache. A private (non-shared) cache may cache the media stream.

no-transform:

An intermediate cache (proxy) may find it useful to convert the media type of a certain stream. A proxy might, for example, convert between video formats to save cache space or to reduce the amount of traffic on a slow link. Serious operational problems may occur, however, when these transformations have been applied to streams intended for certain kinds of applications. For example, applications for medical imaging, scientific data analysis and those using end-to-end authentication all depend on receiving a stream that is bit-for-bit identical to the original entity-body. Therefore, if a response includes the no-transform directive, an intermediate cache or proxy MUST NOT change the encoding of the stream. Unlike HTTP, RTSP does not provide for partial transformation at this point, e.g., allowing translation into

a different language.

only-if-cached:

In some cases, such as times of extremely poor network connectivity, a client may want a cache to return only those media streams that it currently has stored, and not to receive these from the origin server. To do this, the client may include the only-if-cached directive in a request. If it receives this directive, a cache SHOULD either respond using a cached media stream that is consistent with the other constraints of the request, or respond with a 504 (Gateway Timeout) status. However, if a group of caches is being operated as a unified system with good internal connectivity, such a request MAY be forwarded within that group of caches.

max-stale:

Indicates that the client is willing to accept a media stream that has exceeded its expiration time. If max-stale is assigned a value, then the client is willing to accept a response that has exceeded its expiration time by no more than the specified number of seconds. If no value is assigned to max-stale, then the client is willing to accept a stale response of any age.

min-fresh:

Indicates that the client is willing to accept a media stream whose freshness lifetime is no less than its current age plus the specified time in seconds. That is, the client wants a response that will still be fresh for at least the specified number of seconds.

must-revalidate:

When the must-revalidate directive is present in a SETUP response received by a cache, that cache MUST NOT use the entry after it becomes stale to respond to a subsequent

request without first revalidating it with the origin server. That is, the cache must do an end-to-end revalidation every time, if, based solely on the origin server's Expires, the cached response is stale.)

12.9 会议 Conference

This request header field establishes a logical connection between a pre-established conference and an RTSP stream. The conference-id must not be changed for the same RTSP session.

Conference = "Conference" ":" conference-id Example:
Conference: 199702170042.SAA08642@obiwan.arl.wustl.edu%20Starr

A response code of 452 (452 Conference Not Found) is returned if the conference-id is not valid.

12.10 连接 Connection

See [H14.10]

12.11 基本内容 Content-Base

See [H14.11]

12.12 内容编码（Content-Encoding）

See [H14.12]

12.13 内容语言 Content-Language

See [H14.13]

12.14 内容长度（Content-Length）

This field contains the length of the content of the method (i.e.after the double CRLF following the last header). Unlike HTTP, it MUST be included in all messages that carry content beyond the header portion of the message. If it is missing, a default value of zero is assumed. It is interpreted according to [H14.14].

12.15 内容位置 Content-Location

See [H14.15]

12.16 内容类型（Content-Type）

See [H14.18]. Note that the content types suitable for RTSP are likely to be restricted in practice to presentation descriptions and parameter-value types.

12.17 序列号 CSeq

The CSeq field specifies the sequence number for an RTSP request-response pair. This field MUST be present in all requests and responses. For every RTSP request containing the given sequence number, there will be a corresponding response having the same number. Any retransmitted request must contain the same sequence number as the original (i.e. the sequence number is not incremented for retransmissions of the same request).

12.18 日期（Date）

See [H14.19].

12.19 过期 Expires

The Expires entity-header field gives a date and time after which the description or media-stream should be considered stale. The interpretation depends on the method:

DESCRIBE response: The Expires header indicates a date and time after which the description should be considered stale.

A stale cache entry may not normally be returned by a cache (either a proxy cache or an user agent cache) unless it is first validated with the origin server (or with an intermediate cache that has a fresh copy of the entity). See section 13 for further discussion of the expiration model.

The presence of an Expires field does not imply that the original resource will change or cease to exist at, before, or after that time.

The format is an absolute date and time as defined by HTTP-date in [H3.3]; it MUST be in RFC1123-date format:

Expires = "Expires" ":" HTTP-date

An example of its use is

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

RTSP/1.0 clients and caches MUST treat other invalid date formats, especially including the value "0", as having occurred in the past (i.e., "already expired").

To mark a response as "already expired," an origin server should use an Expires date that is equal to the Date header value. To mark a response as "never expires," an origin server should use an Expires date approximately one year from the time the response is sent.

RTSP/1.0 servers should not send Expires dates more than one year in the future.

The presence of an Expires header field with a date value of some time in the future on a media stream that otherwise would by default be non-cacheable indicates

that the media stream is cacheable, unless indicated otherwise by a Cache-Control header field (Section 12.8).

12.20 来自 From

See [H14.22].

12.21 主机 Host

This HTTP request header field is not needed for RTSP. It should be silently ignored if sent.

12.22 如果匹配 If-Match

See [H14.25].

This field is especially useful for ensuring the integrity of the presentation description, in both the case where it is fetched via means external to RTSP (such as HTTP), or in the case where the server implementation is guaranteeing the integrity of the description between the time of the DESCRIBE message and the SETUP message.

The identifier is an opaque identifier, and thus is not specific to any particular session description language.

12.23 从何时更改（If-Modified-Since）

The If-Modified-Since request-header field is used with the DESCRIBE and SETUP methods to make them conditional. If the requested variant has not been modified since the time specified in this field, a description will not be returned from the server (DESCRIBE) or a stream will not be set up (SETUP). Instead, a 304 (not modified) response will be returned without any message-body.

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

An example of the field is:

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

12.24 最近更改（Last-Modified）

The Last-Modified entity-header field indicates the date and time at which the origin server believes the presentation description or media stream was last modified. See [H14.29]. For the methods DESCRIBE or ANNOUNCE, the header field indicates the last modification date and time of the description, for SETUP that of the media stream.

12.25 位置（Location）

See [H14.30].

12.26 代理授权 Proxy-Authenticate

See [H14.33].

12.27 代理要求 Proxy-Require

The Proxy-Require header is used to indicate proxy-sensitive features that MUST be supported by the proxy. Any Proxy-Require header features that are not supported by the proxy MUST be negatively acknowledged by the proxy to the client if not supported. Servers should treat this field identically to the Require field.

See Section 12.32 for more details on the mechanics of this message and a usage example.

12.28 公用性 public

See [H14.35].

12.29 范围 Range

This request and response header field specifies a range of time.The range can be specified in a number of units. This specification defines the smpte (Section 3.5), npt (Section 3.6), and clock (Section 3.7) range units. Within RTSP, byte ranges [H14.36.1] are not meaningful and MUST NOT be used. The header may also contain a time parameter in UTC, specifying the time at which the operation is to be made effective. Servers supporting the Range header MUST understand the NPT range format and SHOULD understand the SMPTE range format. The Range response header indicates what range of time is actually being played or recorded. If the Range header is given in a time format that is not understood, the recipient should return "501 Not Implemented".

Ranges are half-open intervals, including the lower point, but excluding the upper point. In other words, a range of a-b starts exactly at time a, but stops just before b. Only the start time of a media unit such as a video or audio frame is relevant. As an example, assume that video frames are generated every 40 ms. A range of 10.0-10.1 would include a video frame starting at 10.0 or later time and would include a video frame starting at 10.08, even though it lasted beyond the interval. A range of 10.0-10.08, on the other hand, would

exclude the frame at 10.08.

Range = "Range" ":" 1\#ranges-specifier[";" "time" "=" utc-time]

ranges-specifier = npt-range | utc-range | smpte-range

Example:

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

The notation is similar to that used for the HTTP/1.1 [2] byte-range header. It allows clients to select an excerpt from the media object, and to play from a given point to the end as well as from the current location to a given point. The start of playback can be scheduled for any time in the future, although a server may refuse to keep server resources for extended idle periods.

12.30 提交方（Referer）

See [H14.37]. The URL refers to that of the presentation description, typically retrieved via HTTP.

12.31 稍后再试 Retry-After

See [H14.38].

12.32 要求 Require

The Require header is used by clients to query the server about options that it may or may not support. The server MUST respond to this header by using the Unsupported header to negatively acknowledge those options which are NOT supported.

This is to make sure that the client-server interaction will proceed without delay when all options are understood by both sides, and only slow down if options are not understood (as in the case above). For a well-matched client-server pair, the interaction proceeds quickly, saving a round-trip often required by negotiation mechanisms. In addition, it also removes state ambiguity when the client requires features that the server does not understand.

Require = "Require" ":" 1#option-tag

Example:

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

In this example, "funky-feature" is the feature tag which indicates to the client that the fictional Funky-Parameter field is required. The relationship between "funky-feature" and Funky-Parameter is not communicated via the RTSP exchange, since that relationship is an immutable property of "funky-feature" and thus should not be transmitted with every exchange.

Proxies and other intermediary devices SHOULD ignore features that are not understood in this field. If a particular extension requires that intermediate devices support it, the extension should be tagged in the Proxy-Require field instead (see Section 12.27).

12.33 RTP 信息 RTP-Info

This field is used to set RTP-specific parameters in the PLAY response.

url:

Indicates the stream URL which for which the following RTP parameters correspond.

seq:

Indicates the sequence number of the first packet of the stream. This allows clients to gracefully deal with packets when seeking. The client uses this value to differentiate packets that originated before the seek from packets that originated after the seek.

rtptime:

Indicates the RTP timestamp corresponding to the time value in the Range response header. (Note: For aggregate control, a particular stream may not actually generate a packet for the Range time value returned or implied. Thus, there is no guarantee that the packet with the sequence number indicated by seq actually has the timestamp indicated by rtptime.) The client uses this value to calculate the mapping of RTP time to NPT.

A mapping from RTP timestamps to NTP timestamps (wall clock) is available via RTCP. However, this information is not sufficient to generate a mapping from RTP timestamps to NPT. Furthermore, in order to ensure that this information is available at the necessary time (immediately at startup or after a seek), and that it is delivered reliably, this mapping is placed in the RTSP control channel.

In order to compensate for drift for long, uninterrupted presentations, RTSP clients should additionally map NPT to NTP, using initial RTCP sender reports to do the mapping, and later reports to check drift against the mapping.

Syntax:

```
RTP-Info        = "RTP-Info" ":" 1#stream-url 1*parameter
stream-url      = "url" "=" url
parameter       = ";" "seq" "=" 1*DIGIT
                | ";" "rtptime" "=" 1*DIGIT
```

Example:

```
  RTP-Info: url=rtsp://foo.com/bar.avi/streamid=0;seq=45102,
            url=rtsp://foo.com/bar.avi/streamid=1;seq=30211
```

12.34 比例 Scale--播放速度

A scale value of 1 indicates normal play or record at the normal forward viewing rate. If not 1, the value corresponds to the rate with respect to normal viewing rate. For example, a ratio of 2 indicates twice the normal viewing rate ("fast forward") and a ratio of 0.5 indicates half the normal viewing rate. In other words, a ratio of 2 has normal play time increase at twice the wallclock rate.

For every second of elapsed (wallclock) time, 2 seconds of content will be delivered. A negative value indicates reverse direction. Unless requested otherwise by the Speed parameter, the data rate SHOULD not be changed. Implementation of scale changes depends on the server and media type. For video, a server may, for example, deliver only key frames or selected key frames. For audio, it may time-scale the audio while preserving pitch or, less desirably, deliver fragments of audio.

The server should try to approximate the viewing rate, but may restrict the range of scale values that it supports. The response MUST contain the actual scale value chosen by the server.

If the request contains a Range parameter, the new scale value will take effect at that time.

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

Example of playing in reverse at 3.5 times normal rate:

Scale: -3.5

12.35 速度 Speed---码流速度。

This request header fields parameter requests the server to deliver data to the client at a particular speed, contingent on the server's ability and desire to serve the media stream at the given speed. Implementation by the server is OPTIONAL. The default is the bit rate of the stream.

The parameter value is expressed as a decimal ratio, e.g., a value of 2.0 indicates that data is to be delivered twice as fast as normal. A speed of zero is invalid. If the request contains a Range parameter, the new speed value will take effect at that time.

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

Example:

Speed: 2.5

Use of this field changes the bandwidth used for data delivery. It is meant for use in specific circumstances where preview of the presentation at a higher or lower rate is necessary. Implementors should keep in mind that bandwidth for the session may be negotiated beforehand (by means other than RTSP), and therefore re-negotiation may be necessary. When data is delivered over UDP, it is highly recommended that means such as RTCP be used to track packet loss rates.

12.36 服务器（Server）

See [H14.39]

12.37 会话 Session

This request and response header field identifies an RTSP session started by the media server in a SETUP response and concluded by TEARDOWN on the presentation URL. The session identifier is chosen by the media server (see Section 3.4). Once a client receives a Session identifier, it MUST return it for any request related to that session. A server does not have to set up a session identifier if it has other means of identifying a session, such as dynamically generated URLs.

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

The timeout parameter is only allowed in a response header. The server uses it to indicate to the client how long the server is prepared to wait between RTSP commands before closing the session due to lack of activity (see Section A). The timeout is measured in seconds, with a default of 60 seconds (1 minute).

Note that a session identifier identifies a RTSP session across transport sessions or connections. Control messages for more than one RTSP URL may be sent within a single RTSP session. Hence, it is possible that clients use the same session for controlling many streams constituting a presentation, as long as all the streams come from the same server. (See example in Section 14). However, multiple "user" sessions for the same URL from the same client MUST use different session identifiers.

The session identifier is needed to distinguish several delivery requests for the same URL coming from the same client.

The response 454 (Session Not Found) is returned if the session Identifier is invalid.

12.38 时间戳 Timestamp

The timestamp general header describes when the client sent the request to the server. The value of the timestamp is of significance only to the client and may use any timescale. The server MUST echo the exact same value and MAY, if it has accurate information about this, add a floating point number indicating the number of seconds that has elapsed since it has received the request. The timestamp is used

by the client to compute the round-trip time to the server so that it can adjust the timeout value for retransmissions.

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

12.39 传输 Transport

This request header indicates which transport protocol is to be used and configures its parameters such as destination address, compression, multicast time-to-live and destination port for a single stream. It sets those values not already determined by a presentation description.

Transports are comma separated, listed in order of preference. Parameters may be added to each transport, separated by a semicolon.

The Transport header MAY also be used to change certain transport parameters. A server MAY refuse to change parameters of an existing stream.

The server MAY return a Transport response header in the response to indicate the values actually chosen.

A Transport request header field may contain a list of transport options acceptable to the client. In that case, the server MUST return a single option which was actually chosen.

The syntax for the transport specifier is transport/profile/lower-transport.

The default value for the "lower-transport" parameters is specific to the profile. For RTP/AVP, the default is UDP.

Below are the configuration parameters associated with transport:

General parameters:

unicast | multicast:

mutually exclusive indication of whether unicast or multicast delivery will be attempted. Default value is multicast. Clients that are capable of handling both unicast and multicast transmission MUST indicate such capability by including two full transport-specs with separate parameters for each.

destination:

The address to which a stream will be sent. The client may specify the multicast address with the destination parameter. To avoid becoming the unwitting perpetrator of a remote-controlled denial-of-service attack, a server SHOULD authenticate the client and SHOULD log such attempts before allowing the client to direct a media stream to an address not chosen by the server. This is particularly important if RTSP commands are issued via UDP, but implementations cannot rely on TCP as reliable means of client identification by itself. A server SHOULD not allow a client to direct media streams to an address that differs from the address commands are coming from.

source:

If the source address for the stream is different than can be derived from the RTSP endpoint address (the server in playback or the client in recording), the source MAY be specified. This information may also be available through SDP. However, since this is more a feature of transport than media initialization, the authoritative source for this information should be in the SETUP response.

layers:

The number of multicast layers to be used for this media stream. The layers are sent to consecutive addresses starting at the destination address.

mode:

The mode parameter indicates the methods to be supported for this session. Valid values are PLAY and RECORD. If not provided, the default is PLAY.

append:

If the mode parameter includes RECORD, the append parameter indicates that the media data should append to the existing resource rather than overwrite it. If appending is requested and the server does not support this, it MUST refuse the request rather than overwrite the resource identified by the URI. The append parameter is ignored if the mode parameter

does not contain RECORD.

interleaved:

The interleaved parameter implies mixing the media stream with the control stream in whatever protocol is being used by the control stream, using the mechanism defined in Section 10.12. The argument provides the channel number to be used in the $ statement. This parameter may be specified as a range, e.g., interleaved=4-5 in cases where the transport choice for the media stream requires it.

This allows RTP/RTCP to be handled similarly to the way that it is done with UDP, i.e., one channel for RTP and the other for RTCP.

Multicast specific:

ttl:

multicast time-to-live

RTP Specific:

port:

This parameter provides the RTP/RTCP port pair for a multicast session. It is specified as a range, e.g., port=3456-3457.

client_port:

This parameter provides the unicast RTP/RTCP port pair on which the client has chosen to receive media data and control information. It is specified as a range, e.g., client_port=3456-3457.

server_port:

This parameter provides the unicast RTP/RTCP port pair on which the server has chosen to receive media data and control information. It is specified as a range, e.g., server_port=3456-3457.

ssrc:

The ssrc parameter indicates the RTP SSRC [24, Sec. 3] value that should be (request) or will be (response) used by the media server. This parameter is only valid for unicast transmission. It identifies the synchronization source to be

associated with the media stream.

```
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
```

Example:

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

The Transport header is restricted to describing a single RTP stream. (RTSP can also control multiple streams as a single entity.) Making it part of RTSP rather than relying on a multitude of session description formats greatly simplifies designs of firewalls.

12.40 不支持 Unsupported

The Unsupported response header lists the features not supported by the server. In the case where the feature was specified via the Proxy-Require field (Section 12.32), if there is a proxy on the path between the client and the server, the proxy MUST insert a message reply with an error message "551 Option Not Supported".

See Section 12.32 for a usage example.

12.41 用户代理（User-Agent）

See [H14.42]

12.42 变化

See [H14.43]

12.43 通过

See [H14.44]

12.44 WWW-授权（WWW-Authenticate）

See [H14.46]

13 缓存 50

In HTTP, response-request pairs are cached. RTSP differs significantly in that respect. Responses are not cacheable, with the exception of the presentation description returned by DESCRIBE or included with ANNOUNCE. (Since the responses for anything but DESCRIBE and GET_PARAMETER do not return any data, caching is not really an issue for these requests.) However, it is desirable for the continuous

media data, typically delivered out-of-band with respect to RTSP, to be cached, as well as the session description.

HTTP 协议会将应答-请求都缓存起来。RTSP 不同。除了 DESCRIBE 应答的表示描述或者包含有 ANNOUNCE 的应答，其他的应答不会被缓存。（因为除了 DESCRIBE 和 GET_PARAMETER，其他的应答都不带数据，对这些没有应答数据的命令分配缓存是没有意义的。）然而，为连续的媒体数据分配缓存还是有必要的，特别是用 RTSP 协议之外的协议传输数据时，就好像 session 描述一样被缓存。

On receiving a SETUP or PLAY request, a proxy ascertains whether it has an up-to-date copy of the continuous media content and its description. It can determine whether the copy is up-to-date by issuing a SETUP or DESCRIBE request, respectively, and comparing the Last-Modified header with that of the cached copy. If the copy is not up-to-date, it modifies the SETUP transport parameters as appropriate and forwards the request to the origin server. Subsequent control commands such as PLAY or PAUSE then pass the proxy unmodified. The proxy delivers the continuous media data to the client, while possibly making a local copy for later reuse. The exact behavior allowed to the cache is given by the cache-response directives described in Section 12.8. A cache MUST answer any DESCRIBE requests if it is currently serving the stream to the requestor, as it is possible that low-level details of the stream description may have changed on the origin-server.

当收到一个 SETUP 或者 PLAY 请求时，代理服务器检验连续的媒体内容和描述是否是最新版本。它能确定由 SETUP 或者 DESCRIBE 发出的请求是否是最新的，然后，将最后修改的头与缓存的数据进行比较。如果版本不是最新的，则向起始服务器修改 SETUP 传输参数。后面的控制命令如 PLAY 或者 PAUSE 将直接穿过代理。代理在向客户端传送连续媒体数据的同时，可能为了后面的重用产生本地版本。在 12.8 节已经给出了实际的允许缓存行为的应答命令描述。

Note that an RTSP cache, unlike the HTTP cache, is of the "cut-through" variety. Rather than retrieving the whole resource from the origin server, the cache simply copies the streaming data as it passes by on its way to the client. Thus, it does not introduce additional latency.

注意：RTSP 缓存与 HTTP 缓存不同的地方是穿透（cut-through）变化。相对于从源服务器检索整个资源的过程，缓存简单的将经过这里传给客户端的的流数据缓存。因此，不会引起额外的等待时间。

To the client, an RTSP proxy cache appears like a regular media server, to the media origin server like a client. Just as an HTTP cache has to store the content type, content language, and so on for the objects it caches, a media cache has to

store the presentation description. Typically, a cache eliminates all transport-references (that is, multicast information) from the presentation description, since these are independent of the data delivery from the cache to the client. Information on the encodings remains the same. If the cache is able to translate the cached media data, it would create a new presentation description with all the encoding possibilities it can offer.

14 实例 Examples

The following examples refer to stream description formats that are not standards, such as RTSL. The following examples are not to be used as a reference for those formats.

14.1 要求媒体（单播） Media on Demand (Unicast)

Client C requests a movie from media servers A (audio.example.com) and V (video.example.com). The media description is stored on a web server W . The media description contains descriptions of the presentation and all its streams,including the codecs that are available, dynamic RTP payload types, the protocol stack, and content information such as language or copyright restrictions. It may also give an indication about the timeline of the movie. In this example, the client is only interested in the last part of the movie.

```
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
```

Even though the audio and video track are on two different servers, and may start at slightly different times and may drift with respect to each other, the client can synchronize the two using standard RTP methods, in particular the time scale contained in the RTCP sender reports.

14.2 容器文件的流 Streaming of a Container file

For purposes of this example, a container file is a storage entity in which multiple continuous media types pertaining to the same end-user presentation are present. In effect, the container file represents an RTSP presentation, with each of its components being RTSP streams. Container files are a widely used means to store such presentations. While the components are transported as independent streams, it is desirable to maintain a common context for those streams at the server end.

This enables the server to keep a single storage handle open easily. It also allows treating all the streams equally in case of any prioritization of streams by the server.

It is also possible that the presentation author may wish to prevent selective retrieval of the streams by the client in order to preserve the artistic effect of the combined media presentation. Similarly, in such a tightly bound presentation, it is desirable to be able to control all the streams via a single control message using an aggregate URL.

The following is an example of using a single RTSP session to control multiple streams. It also illustrates the use of aggregate URLs.

Client C requests a presentation from media server M . The movie is stored in a container file. The client has obtained an RTSP URL to the container file.

```
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
```

In the first instance of failure, the client tries to pause one stream (in this case video) of the presentation. This is disallowed for that presentation by the server. In the second instance, the aggregate URL may not be used for SETUP and one control message is required per stream to set up transport parameters.

This keeps the syntax of the Transport header simple and allows easy parsing of transport information by firewalls.

14.3 单个流容器文件 Single Stream Container Files

Some RTSP servers may treat all files as though they are "container files", yet other servers may not support such a concept. Because of this, clients SHOULD use the rules set forth in the session description for request URLs, rather than assuming that a consistent URL may always be used throughout. Here's an example of how a multi-stream server might expect a single-stream file to be served:

```
        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
```

Note the different URL in the SETUP command, and then the switch back to the aggregate URL in the PLAY command. This makes complete sense when there are multiple streams with aggregate control, but is less than intuitive in the special case where the number of streams is one.

In this special case, it is recommended that servers be forgiving of implementations that send:

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

In the worst case, servers should send back:

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

One would also hope that server implementations are also forgiving of the following:

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

Since there is only a single stream in this file, it's not ambiguous what this means.

14.4 组播现场媒体表示　Live Media Presentation Using Multicast

The media server M chooses the multicast address and port. Here, we assume that the web server only contains a pointer to the full description, while the media server M maintains the full description.

```
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 在存在的会话中播放媒体 Playing media into an existing session

A conference participant C wants to have the media server M play back a demo tape into an existing conference. C indicates to the media server that the network

addresses and encryption keys are already given by the conference, so they should not be chosen by the server.

The example omits the simple ACK responses.

```
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 Recording

The conference participant client C asks the media server M to record the audio and video portions of a meeting. The client uses the ANNOUNCE method to provide meta-information about the recorded session to the server.

```
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 Syntax

The RTSP syntax is described in an augmented Backus-Naur form (BNF) as used in RFC 2068 [2].

15.1 基本语法 Base Syntax

```
OCTET          =    <any 8-bit sequence of data>
CHAR           =    <any US-ASCII character (octets 0 - 127)>
UPALPHA        =    <any US-ASCII uppercase letter "A".."Z">
LOALPHA        =    <any US-ASCII lowercase letter "a".."z">
ALPHA          =    UPALPHA | LOALPHA

DIGIT          =    <any US-ASCII digit "0".."9">
CTL            =    <any US-ASCII control character
                     (octets 0 - 31) and DEL (127)>
CR             =    <US-ASCII CR, carriage return (13)>
LF             =    <US-ASCII LF, linefeed (10)>

SP             =    <US-ASCII SP, space (32)>
HT             =    <US-ASCII HT, horizontal-tab (9)>
<">            =    <US-ASCII double-quote mark (34)>
CRLF           =    CR LF

LWS            =    [CRLF] 1*( SP | HT )
TEXT           =    <any OCTET except CTLs>
tspecials      =    "(" | ")" | "<" | ">" | "@"
                |    "," | ";" | ":" | "\" | <">
                |    "/" | "[" | "]" | "?" | "="
                |    "{" | "}" | SP | HT

token          =    1*<any CHAR except CTLs or tspecials>
```

```
quoted-string      =      ( <"> *(qdtext) <"> )
qdtext             =      <any TEXT except <">>
quoted-pair        =      "\" CHAR

message-header     =      field-name ":" [ field-value ] CRLF
field-name         =      token
field-value        =      *( field-content | LWS )
field-content      =      <the OCTETs making up the field-value and
                           consisting of either *TEXT or
                           combinations of token, tspecials, and
                           quoted-string>

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（Security Considerations）

Because of the similarity in syntax and usage between RTSP servers and HTTP servers, the security considerations outlined in [H15] apply. Specifically, please note the following:

Authentication Mechanisms:

RTSP and HTTP share common authentication schemes, and thus should follow the same prescriptions with regards to authentication. See [H15.1] for client authentication issues,

and [H15.2] for issues regarding support for multiple authentication mechanisms.

Abuse of Server Log Information:

RTSP and HTTP servers will presumably have similar logging mechanisms, and thus should be equally guarded in protecting the contents of those logs, thus protecting the privacy of the

users of the servers. See [H15.3] for HTTP server recommendations regarding server logs.

Transfer of Sensitive Information:

There is no reason to believe that information transferred via RTSP may be any less sensitive than that normally transmitted via HTTP. Therefore, all of the precautions regarding the protection of data privacy and user privacy apply to implementors of RTSP clients, servers, and proxies. See [H15.4] for further details.

Attacks Based On File and Path Names:

Though RTSP URLs are opaque handles that do not necessarily have file system semantics, it is anticipated that many implementations will translate portions of the request URLs directly to file system calls. In such cases, file systems SHOULD follow the precautions outlined in [H15.5], such as checking for ".." in path components.

Personal Information:

RTSP clients are often privy to the same information that HTTP clients are (user name, location, etc.) and thus should be equally. See [H15.6] for further recommendations.

Privacy Issues Connected to Accept Headers:

Since may of the same "Accept" headers exist in RTSP as in HTTP, the same caveats outlined in [H15.7] with regards to

their use should be followed.

DNS Spoofing:

Presumably, given the longer connection times typically associated to RTSP sessions relative to HTTP sessions, RTSP client DNS optimizations should be less prevalent. Nonetheless, the recommendations provided in [H15.8] are still relevant to any implementation which attempts to rely on a DNS-to-IP mapping to hold beyond a single use of the mapping.

Location Headers and Spoofing:

If a single server supports multiple organizations that do not trust one another, then it must check the values of Location and Content-Location headers in responses that are generated under control of said organizations to make sure that they do not attempt to invalidate resources over which they have no authority. ([H15.9])

In addition to the recommendations in the current HTTP specification (RFC 2068 [2], as of this writing), future HTTP specifications may provide additional guidance on security issues.

The following are added considerations for RTSP implementations.

Concentrated denial-of-service attack:

The protocol offers the opportunity for a remote-controlled denial-of-service attack. The attacker may initiate traffic flows to one or more IP addresses by specifying them as the destination in SETUP requests. While the attacker's IP address may be known in this case, this is not always useful in prevention of more attacks or ascertaining the attackers identity. Thus, an RTSP server SHOULD only allow client-specified destinations for RTSP-initiated traffic flows if the server has verified the client's identity, either against a database of known users using RTSP authentication mechanisms

(preferably digest authentication or stronger), or other secure means.

Session hijacking:

Since there is no relation between a transport layer connection and an RTSP session, it is possible for a malicious client to issue requests with random session identifiers which would affect unsuspecting clients. The server SHOULD use a large, random and non-sequential session identifier to minimize the possibility of this kind of attack.

Authentication:

Servers SHOULD implement both basic and digest [8] authentication. In environments requiring tighter security for the control messages, the RTSP control stream may be encrypted.

Stream issues:

RTSP only provides for stream control. Stream delivery issues are not covered in this section, nor in the rest of this memo. RTSP implementations will most likely rely on other protocols such as RTP, IP multicast, RSVP and IGMP, and should address security considerations brought up in those and other applicable specifications.

Persistently suspicious behavior:

RTSP servers SHOULD return error code 403 (Forbidden) upon receiving a single instance of behavior which is deemed a security risk. RTSP servers SHOULD also be aware of attempts to probe the server for weaknesses and entry points and MAY arbitrarily disconnect and ignore further requests clients which are deemed to be in violation of local security policy.

附录 A: RTSP 协议状态机 RTSP Protocol State Machines

The RTSP client and server state machines describe the behavior of the protocol from RTSP session initialization through RTSP session termination.

State is defined on a per object basis. An object is uniquely identified by the stream URL and the RTSP session identifier. Any request/reply using aggregate URLs denoting RTSP presentations composed of multiple streams will have an effect on the individual states of all the streams. For example, if the presentation /movie contains two streams, /movie/audio and /movie/video, then the following command:

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

will have an effect on the states of movie/audio and movie/video. This example does not imply a standard way to represent streams in URLs or a relation to the filesystem. See Section 3.2.

The requests OPTIONS, ANNOUNCE, DESCRIBE, GET_PARAMETER, SET_PARAMETER do not have any effect on client or server state and are therefore not listed in the state tables.

A.1 客户端状态机 Client State Machine

The client can assume the following states:

Init:

SETUP has been sent, waiting for reply.

Ready:

SETUP reply received or PAUSE reply received while in Playing state.

Playing:

PLAY reply received

Recording:

RECORD reply received

In general, the client changes state on receipt of replies to requests. Note that some requests are effective at a future time or position (such as a PAUSE), and state also changes accordingly. If no explicit SETUP is required for the object (for example, it is available via a multicast group), state begins at Ready. In this case, there are only two states, Ready and Playing. The client also changes state from Playing/Recording to Ready when the end of the requested range is reached.

The "next state" column indicates the state assumed after receiving a success response (2xx). If a request yields a status code of 3xx, the state becomes Init, and a status code of 4xx yields no change in state. Messages not listed for each state MUST NOT be issued by the client in that state, with the exception of messages not affecting state, as listed above. Receiving a REDIRECT from the server is equivalent to receiving a 3xx redirect status from the server.

state	message sent	next state after response
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 服务器端状态机 Server State Machine

The server can assume the following states:

Init:

The initial state, no valid SETUP has been received yet.

Ready:

Last SETUP received was successful, reply sent or after playing, last PAUSE received was successful, reply sent.

Playing:

Last PLAY received was successful, reply sent. Data is being sent.

Recording:

The server is recording media data.

In general, the server changes state on receiving requests. If the server is in state Playing or Recording and in unicast mode, it MAY revert to Init and tear down the RTSP session if it has not received "wellness" information, such as RTCP reports or RTSP commands, from the client for a defined interval, with a default of one minute. The server can declare another timeout value in the Session response header (Section 12.37). If the server is in state Ready, it MAY revert to Init if it does not receive an RTSP request for an interval of more than one minute. Note that some requests (such as PAUSE) may

be effective at a future time or position, and server state changes at the appropriate time. The server reverts from state Playing or Recording to state Ready at the end of the range requested by the client.

The REDIRECT message, when sent, is effective immediately unless it has a Range header specifying when the redirect is effective. In such a case, server state will also change at the appropriate time.

If no explicit SETUP is required for the object, the state starts at Ready and there are only two states, Ready and Playing.

The "next state" column indicates the state assumed after sending a success response (2xx). If a request results in a status code of 3xx, the state becomes Init. A status code of 4xx results in no change.

state	message received	next state
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 协议的交互 Interaction with RTP

RTSP allows media clients to control selected, non-contiguous sections of media presentations, rendering those streams with an RTP media layer[24]. The media layer rendering the RTP stream should not be affected by jumps in NPT. Thus, both RTP sequence numbers and RTP timestamps MUST be continuous and monotonic across jumps of NPT.

As an example, assume a clock frequency of 8000 Hz, a packetization interval of 100 ms and an initial sequence number and timestamp of zero. First we play NPT 10 through 15, then skip ahead and play NPT 18 through 20. The first segment is presented as RTP packets with sequence numbers 0 through 49 and timestamp 0 through 39,200. The second segment consists of RTP packets with sequence number 50 through 69, with timestamps 40,000 through 55,200.

We cannot assume that the RTSP client can communicate with the RTP media agent, as the two may be independent processes. If the RTP timestamp shows the same gap as the NPT, the media agent will assume that there is a pause in the presentation. If the jump in NPT is large enough, the RTP timestamp may roll over and the media agent may believe later packets to be duplicates of packets just played out.

For certain datatypes, tight integration between the RTSP layer and the RTP layer will be necessary. This by no means precludes the above restriction. Combined RTSP/RTP media clients should use the RTP-Info field to determine whether incoming RTP packets were sent before or after a seek.

For continuous audio, the server SHOULD set the RTP marker bit at the beginning of serving a new PLAY request. This allows the client to perform playout delay adaptation.

For scaling (see Section 12.34), RTP timestamps should correspond to the playback timing. For example, when playing video recorded at 30 frames/second at a scale of

two and speed (Section 12.35) of one, the server would drop every second frame to maintain and deliver video packets with the normal timestamp spacing of 3,000 per frame, but NPT would increase by 1/15 second for each video frame.

The client can maintain a correct display of NPT by noting the RTP timestamp value of the first packet arriving after repositioning. The sequence parameter of the RTP-Info (Section 12.33) header provides the first sequence number of the next segment.

附录 C 使用 SDP 进行 RTSP 会话描述 Use of SDP for RTSP Session Descriptions

The Session Description Protocol (SDP, RFC 2327 [6]) may be used to describe streams or presentations in RTSP. Such usage is limited to specifying means of access and encoding(s) for:

aggregate control:

A presentation composed of streams from one or more servers that are not available for aggregate control. Such a description is typically retrieved by HTTP or other non-RTSP means. However, they may be received with ANNOUNCE methods.

non-aggregate control:

A presentation composed of multiple streams from a single server that are available for aggregate control. Such a description is typically returned in reply to a DESCRIBE request on a URL, or received in an ANNOUNCE method.

This appendix describes how an SDP file, retrieved, for example, through HTTP, determines the operation of an RTSP session. It also describes how a client should interpret SDP content returned in reply to a DESCRIBE request. SDP provides no mechanism by which a client

can distinguish, without human guidance, between several media streams to be rendered simultaneously and a set of alternatives (e.g., two audio streams spoken in different languages).

C.1 定义 Definitions

The terms "session-level", "media-level" and other key/attribute names and values used in this appendix are to be used as defined in SDP (RFC 2327 [6]):

C.1.1 控制 URL Control URL

The "a=control:" attribute is used to convey the control URL. This attribute is used both for the session and media descriptions. If used for individual media, it indicates the URL to be used for controlling that particular media stream. If found at the session level, the attribute indicates the URL for aggregate control.

Example:

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

This attribute may contain either relative and absolute URLs, following the rules and conventions set out in RFC 1808 [25]. Implementations should look for a base URL in the following order:

1. The RTSP Content-Base field
2. The RTSP Content-Location field
3. The RTSP request URL

If this attribute contains only an asterisk (*), then the URL is treated as if it were an empty embedded URL, and thus inherits the entire base URL.

C.1.2 媒体流 Media streams

The "m=" field is used to enumerate the streams. It is expected that all the specified streams will be rendered with appropriate synchronization. If the session is unicast, the port number serves as a recommendation from the server to the client; the client still has to include it in its SETUP request and may ignore this recommendation. If the server has no preference, it SHOULD set the port number value to zero.

Example:
  m=audio 0 RTP/AVP 31

C.1.3 有效载荷类型 Payload type(s)

The payload type(s) are specified in the "m=" field. In case the payload type is a static payload type from RFC 1890 [1], no other information is required. In case it is a dynamic payload type, the media attribute "rtpmap" is used to specify what the media is. The "encoding name" within the "rtpmap" attribute may be one of those specified in RFC 1890 (Sections 5 and 6), or an experimental encoding with a "X-" prefix as specified in SDP (RFC 2327 [6]). Codec-specific parameters are not specified in this field, but rather in the "fmtp" attribute described below. Implementors seeking to register new encodings should follow the procedure in RFC 1890 [1]. If the media type is not suited to the RTP AV profile, then it is recommended that a new profile be created and the appropriate profile name be used in lieu of "RTP/AVP" in the "m=" field.

C.1.4 详细格式参数 Format-specific parameters

Format-specific parameters are conveyed using the "fmtp" media

attribute. The syntax of the "fmtp" attribute is specific to the encoding(s) that the attribute refers to. Note that the packetization interval is conveyed using the "ptime" attribute.

C.1.5 表示的范围 Range of presentation

The "a=range" attribute defines the total time range of the stored session. (The length of live sessions can be deduced from the "t" and "r" parameters.) Unless the presentation contains media streams of different durations, the range attribute is a session-level attribute. The unit is specified first, followed by the value range. The units and their values are as defined in Section 3.5, 3.6 and 3.7.

Examples:

```
a=range:npt=0-34.4368
a=range:clock=19971113T2115-19971113T2203
```

C.1.6 有效时间 Time of availability

The "t=" field MUST contain suitable values for the start and stop times for both aggregate and non-aggregate stream control. With aggregate control, the server SHOULD indicate a stop time value for which it guarantees the description to be valid, and a start time that is equal to or before the time at which the DESCRIBE request was received. It MAY also indicate start and stop times of 0, meaning that the session is always available. With non-aggregate control, the values should reflect the actual period for which the session is available in keeping with SDP semantics, and not depend on other means (such as the life of the web page containing the description) for this purpose.

C.1.7 连接信息 Connection Information

In SDP, the "c=" field contains the destination address for the media stream. However, for on-demand unicast streams and some multicast

streams, the destination address is specified by the client via the SETUP request. Unless the media content has a fixed destination address, the "c=" field is to be set to a suitable null value. For addresses of type "IP4", this value is "0.0.0.0".

C.1.8 实体标签 Entity Tag

The optional "a=etag" attribute identifies a version of the session description. It is opaque to the client. SETUP requests may include this identifier in the If-Match field (see section 12.22) to only allow session establishment if this attribute value still corresponds to that of the current description. The attribute value is opaque and may contain any character allowed within SDP attribute values.

Example:

```
a=etag:158bb3e7c7fd62ce67f12b533f06b83a
```

One could argue that the "o=" field provides identical functionality. However, it does so in a manner that would put constraints on servers that need to support multiple session description types other than SDP for the same piece of media content.

C.2 合控制不可用 Aggregate Control Not Available

If a presentation does not support aggregate control and multiple media sections are specified, each section MUST have the control URL specified via the "a=control:" attribute.

Example:

```
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
```

Note that the position of the control URL in the description implies that the client establishes separate RTSP control sessions to the servers audio.com and video.com.

It is recommended that an SDP file contains the complete media initialization information even if it is delivered to the media client through non-RTSP means. This is necessary as there is no mechanism to indicate that the client should request more detailed media stream information via DESCRIBE.

C.3 合控制可用 Aggregate Control Available

In this scenario, the server has multiple streams that can be controlled as a whole. In this case, there are both media-level "a=control:" attributes, which are used to specify the stream URLs, and a session-level "a=control:" attribute which is used as the request URL for aggregate control. If the media-level URL is relative, it is resolved to absolute URLs according to Section C.1.1 above.

If the presentation comprises only a single stream, the media-level "a=control:" attribute may be omitted altogether. However, if the presentation contains more than one stream, each media stream section MUST contain its own "a=control" attribute.

Example:

```
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
```

In this example, the client is required to establish a single RTSP session to the server, and uses the URLs rtsp://example.com/movie/trackID=1 and rtsp://example.com/movie/trackID=2 to set up the video and audio streams, respectively. The URL rtsp://example.com/movie/ controls the whole movie.

附录 D 最简单的 RTSP 实现 Appendix D: Minimal RTSP implementation

D.1 客户端 Client

A client implementation MUST be able to do the following :

* Generate the following requests: SETUP, TEARDOWN, and one of PLAY (i.e., a minimal playback client) or RECORD (i.e., a minimal

recording client). If RECORD is implemented, ANNOUNCE must be implemented as well.

* Include the following headers in requests: CSeq, Connection, Session, Transport. If ANNOUNCE is implemented, the capability to include headers Content-Language, Content-Encoding, Content-Length, and Content-Type should be as well.
* Parse and understand the following headers in responses: CSeq, Connection, Session, Transport, Content-Language, Content-Encoding, Content-Length, Content-Type. If RECORD is implemented, the Location header must be understood as well. RTP-compliant implementations should also implement RTP-Info.
* Understand the class of each error code received and notify the end-user, if one is present, of error codes in classes 4xx and 5xx. The notification requirement may be relaxed if the end-user explicitly does not want it for one or all status codes.
* Expect and respond to asynchronous requests from the server, such as ANNOUNCE. This does not necessarily mean that it should implement the ANNOUNCE method, merely that it MUST respond positively or negatively to any request received from the server.

Though not required, the following are highly recommended at the time of publication for practical interoperability with initial implementations and/or to be a "good citizen".

* Implement RTP/AVP/UDP as a valid transport.
* Inclusion of the User-Agent header.
* Understand SDP session descriptions as defined in Appendix C
* Accept media initialization formats (such as SDP) from standard input, command line, or other means appropriate to the operating environment to act as a "helper application" for other applications (such as web browsers).

There may be RTSP applications different from those initially envisioned by the contributors to the RTSP specification for which the requirements above do not make sense. Therefore, the

recommendations above serve only as guidelines instead of strict requirements.

D.1.1 回放 Basic Playback

To support on-demand playback of media streams, the client MUST additionally be able to do the following:

* generate the PAUSE request;
* implement the REDIRECT method, and the Location header.

D.1.2 授权 Authentication-enabled

In order to access media presentations from RTSP servers that require authentication, the client MUST additionally be able to do the following:

* recognize the 401 status code;
* parse and include the WWW-Authenticate header;
* implement Basic Authentication and Digest Authentication.

D.2 服务器 Server

A minimal server implementation MUST be able to do the following:

* Implement the following methods: SETUP, TEARDOWN, OPTIONS and either PLAY (for a minimal playback server) or RECORD (for a minimal recording server). If RECORD is implemented, ANNOUNCE should be implemented as well.
* Include the following headers in responses: Connection, Content-Length, Content-Type, Content-Language, Content-Encoding, Transport, Public. The capability to include the Location header should be implemented if the RECORD method is. RTP-compliant

implementations should also implement the RTP-Info field.

* Parse and respond appropriately to the following headers in requests: Connection, Session, Transport, Require.

Though not required, the following are highly recommended at the time of publication for practical interoperability with initial implementations and/or to be a "good citizen".

* Implement RTP/AVP/UDP as a valid transport.
* Inclusion of the Server header.
* Implement the DESCRIBE method.
* Generate SDP session descriptions as defined in Appendix C

There may be RTSP applications different from those initially envisioned by the contributors to the RTSP specification for which the requirements above do not make sense. Therefore, the recommendations above serve only as guidelines instead of strict requirements.

D.2.1 回放 Basic Playback

To support on-demand playback of media streams, the server MUST additionally be able to do the following:

* Recognize the Range header, and return an error if seeking is not supported.
* Implement the PAUSE method.

In addition, in order to support commonly-accepted user interface features, the following are highly recommended for on-demand media servers:

* Include and parse the Range header, with NPT units.

Implementation of SMPTE units is recommended.

* Include the length of the media presentation in the media initialization information.
* Include mappings from data-specific timestamps to NPT. When RTP is used, the rtptime portion of the RTP-Info field may be used to map RTP timestamps to NPT.

Client implementations may use the presence of length information to determine if the clip is seekable, and visibly disable seeking features for clips for which the length information is unavailable. A common use of the presentation length is to implement a "slider bar" which serves as both a progress indicator and a timeline positioning tool.

Mappings from RTP timestamps to NPT are necessary to ensure correct positioning of the slider bar.

D.2.2 授权 Authentication-enabled

In order to correctly handle client authentication, the server MUST additionally be able to do the following:

* Generate the 401 status code when authentication is required for the resource.
* Parse and include the WWW-Authenticate header
* Implement Basic Authentication and Digest Authentication

附录 E 作者地址 Authors' Addresses

Henning Schulzrinne

Dept. of Computer Science

Columbia University
1214 Amsterdam Avenue
New York, NY 10027
USA

EMail: schulzrinne@cs.columbia.edu

Anup Rao
Netscape Communications Corp.
501 E. Middlefield Road
Mountain View, CA 94043
USA

EMail: anup@netscape.com

Robert Lanphier
RealNetworks
1111 Third Avenue Suite 2900
Seattle, WA 98101
USA

EMail: robla@real.com

附录 F 致谢 Acknowledgements

This memo is based on the functionality of the original RTSP document submitted in October 96. It also borrows format and descriptions from HTTP/1.1.

This document has benefited greatly from the comments of all those participating in the MMUSIC-WG. In addition to those already mentioned, the following individuals have contributed to this specification:

Rahul Agarwal, Torsten Braun, Brent Browning, Bruce Butterfield, Steve Casner, Francisco Cortes, Kelly Djahandari, Martin Dunsmuir, Eric Fleischman, Jay Geagan, Andy Grignon, V. Guruprasad, Peter Haight, Mark Handley, Brad Hefta-Gaub, John K. Ho, Philipp Hoschka, Anne Jones, Anders Klemets, Ruth Lang, Stephanie Leif, Jonathan Lennox, Eduardo F. Llach, Rob McCool, David Oran, Maria Papadopouli, Sujal Patel, Ema Patki, Alagu Periyannan, Igor Plotnikov, Pinaki Shah, David Singer, Jeff Smith, Alexander Sokolsky, Dale Stammen, and John Francis Stracke.

参考书目 References

1 Schulzrinne, H., "RTP profile for audio and video conferences with minimal control", RFC 1890, January 1996.

2 Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and T. Berners-Lee, "Hypertext transfer protocol - HTTP/1.1", RFC 2068, January 1997.

3 Yergeau, F., Nicol, G., Adams, G., and M. Duerst, "Internationalization of the hypertext markup language", RFC 2070, January 1997.

4 Bradner, S., "Key words for use in RFCs to indicate requirement levels", BCP 14, RFC 2119, March 1997.

5 ISO/IEC, "Information technology - generic coding of moving pictures and associated audio information - part 6: extension for digital storage media and control," Draft International Standard ISO 13818-6, International Organization for Standardization ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, Nov. 1995.

6 Handley, M., and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

7 Franks, J., Hallam-Baker, P., and J. Hostetler, "An extension to HTTP: digest access authentication", RFC 2069, January 1997.

8 Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

9 Hinden, B. and C. Partridge, "Version 2 of the reliable data protocol (RDP)", RFC 1151, April 1990.

10 Postel, J., "Transmission control protocol", STD 7, RFC 793, September 1981.

11 H. Schulzrinne, "A comprehensive multimedia control architecture for the Internet," in Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), (St. Louis, Missouri), May 1997.

12 International Telecommunication Union, "Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service," Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1996.

13 McMahon, P., "GSS-API authentication method for SOCKS version 5", RFC 1961, June 1996.

14 J. Miller, P. Resnick, and D. Singer, "Rating services and rating systems (and their machine readable descriptions)," Recommendation REC-PICS-services-961031, W3C (World Wide Web Consortium), Boston, Massachusetts, Oct. 1996.

15 J. Miller, T. Krauskopf, P. Resnick, and W. Treese, "PICS label distribution label syntax and communication protocols," Recommendation REC-PICS-labels-961031, W3C (World Wide Web Consortium), Boston, Massachusetts, Oct. 1996.

16 Crocker, D. and P. Overell, "Augmented BNF for syntax specifications: ABNF", RFC 2234, November 1997.

17 Braden, B., "Requirements for internet hosts - application and support", STD 3, RFC 1123, October 1989.

18 Elz, R., "A compact representation of IPv6 addresses", RFC 1924, April 1996.

19 Berners-Lee, T., Masinter, L. and M. McCahill, "Uniform resource locators (URL)", RFC 1738, December 1994.

20 Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

22 Braden, B., "T/TCP - TCP extensions for transactions functional specification", RFC 1644, July 1994.

22 W. R. Stevens, TCP/IP illustrated: the implementation, vol. 2. Reading, Massachusetts: Addison-Wesley, 1994.

23 Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: a transport protocol for real-time applications", RFC 1889, January 1996.

24 Fielding, R., "Relative uniform resource locators", RFC 1808, June 1995.

版权申明 Full Copyright Statement