Adobe 的实时消息协议

H. Parmar, Ed.
M. Thornburgh, Ed.
Adobe
December 21, 2012
nie950 at gmail.com 翻译

目录

绪论

关于 Adobe 的实时消息协议(Real Time Messaging Protocol,RTMP)，是一种多媒体的复用和分组的应用层协议，通过某种可靠的传输协议（例如 TCP）传输数据流(例如音频，视频和交互数据)。

简介

Adobe 的实时消息协议(Real Time Messaging Protocol，RTMP)在两个对等的通信端之间通过可靠的传输协议，例如 TCP[RFC0793]，提供双向的消息多路服务，用来传输带有时间信息的并行的视频，音频和数据。通常的协议的实现会给不同类型的消息赋予不同的优先级，当传输能力受到限制时它会影响消息下层流发送的队列顺序。

这个备忘录描述了实时消息协议的语法规则和操作方法。

1.1 术语

在备忘录中的关键字“一定（MUST）”，“一定不（MUST NOT）”，“要求（REQUIRED）”，“将是（SHALL）”，“将不是（SHALL NOT）”，“应该（SHOULD）”，“不应该（SHOULD NOT）”，“推荐（RECOMMENDED）”，“不推荐（NOT RECOMMENDED）”，“可能（MAY）”，和“可选（OPTIONAL）”表示的意义在[RFC2119]中描述。

1.贡献者

Rajesh Mallipenddi，前 Adobe Systems 员工，本规范的最早编辑者，并提供了本规范的大部分原始文稿。

Mohit Srivastava，Adobe Systems 员工，为本规范的发展作出了贡献。

2.定义

载荷(Payload)：数据包中的数据，比如音频采样或者压缩后的视频数据。载荷的格式和表示的意义不在本文的讨论的范围。

数据包(Packet)：一个数据包由固定的头和载荷数据组成。一些下层的协议可能需要定义数据包的封包。

端口(Port)：“传输协议在给定的主机上用来区分不同目的地的抽象。TCP/IP 使用较小的正整数来标识端口。”OSI 传输层使用的传输选择器（TSEL）相当于端口。

传输地址(Transport address)：网络地址和端口的组合标识了传输层的端点，比如 IP 地址和 TCP 端口。数据包从源传输地址传送到目的传输地址。

消息流(Message stream)：在通信过程中消息流动的一个逻辑通道。

消息流 ID(Message stream ID)：每一个消息都有一个 ID，这个 ID 用来区分这个消息传送使用的是哪一个消息流。

块(Chunk)：消息的一个分片。消息在通过网络发送前被划分成更小的部分并且交叉的读取。块保证即使横跨不同的流所有消息的都是有序的端对端的传送。

块流(Chunk stream)：通信的逻辑通道，它允许块在特定的方向流动。块流的方向可以是从客户端到服务器或者相反方向。

块流 ID(Chunk stream ID)：每一个块都有一个 ID，它用来标识块传送使用的是哪一个块流。

混流(Multiplexing)：将单独的音/视频数据混合成一个音/视频流的过程，它使得可以同时传送多个音视频数据。

分流(DeMultiplexing)：混流的反过程，它是将交错的音视频数据组装还原成原始的音频和视频数据。

原程过程调用(Remote Procedure Call，RPC)：一种请求，客户端或服务器用它去调用在对端上的子过程或程序。

元数据(Metadata)：数据的描述。电影的元数据包括电影标题，时长和创建时间等等。

应用程序实例(Application Instance)：服务器上的应用程序实例，客户端通过向这个应用程序实例发送连接请求来连接服务器。

动作消息格式(Action Message Format，AMF)：一种二进制紧凑格式，用来序列化 Action Script 对象图。AFM 有两个版本：AFM 0[AMF0]和 AMF 3[AMF3]。

3.字节序，对齐，时间格式

所有整数字段的表示都使用网络字节序，零字节在最前面显示，同时零位是字或字段的最重要的位（译者注：零位是符号位）。这种字节序通常被称为大端(big-endian)。传输顺序在网际协议（Internet Protocol）[RFC0791]中有具体的描述。除非另有说明，本文所有常量数值均为 10 进制（10 为基数）。

除了其它指定外，所有的 RTMP 协议都是单字节对齐的；比如说，一个 16 位的字段起始位置可能是一个奇数偏移。如果需要填充，填充字节应该（SHOULD）是 0。

RTMP 中给定的时间戳是从一个未指定的时间点开始的相对时间的整数毫秒数。通常，每个流都将从时间戳为 0 为起始时间，但这不是必须的，只要两端的能在时间点上达成一致。注意这意味着不同流（尤其是来自不同的主机）的同步需要 RTMP 之外的同步机制。

因为时间戳是 32 位长，它们将在 49 天，17 小时，2 分 47.296 秒后重新开始。由于流被要求持续不断的传输，可能要运行几年后才结束，所以 RTMP 应用程序在处理时间戳时应该（SHOULD）使用序列号算法（Serial Number Arithmetic）[RFC1982]，应该（SHOULD）能处理这个环绕的情形。比如说，应用程序应该认为所有相邻的时间戳之差都在 2^31-1 之内（译者注：2^31 表示 2 的 31 次方，32 位的无符号数最大能表示 2^32 - 1=4294967295），所 1000 应该在 4000000000 之后，而 3000000000 在 4000000000 之前。

时间戳增量同样被定义成相对于前一个时间戳的无符号相对毫秒数。时间戳增量可能是

24 或者 32 位长。

4.RTMP 块流

本节讨论实时消息协议的块流(RTMP Chunk Stream)。它为高层的多媒体协议提供了混流和组包服务。

尽管 RTMP 的块流工作成 RTMP 协议上（第 6 节），它也可以使用任何协议来发送消息数据。第一个消息都包含了时间戳和载荷的类型识别。RTMP 块流和 RTMP 协议协同工作很适合于各种和样的音视频程序，从一对一和一对多的直播到视频点播（video-on-demand）服务再到互动会议程序。

当我们使用可靠的传输协议比如 TCP[RFC0793]，RTMP 块流保证横跨多个流的所有消息端到端传输的时序。RTMP 流不提供优先级或者类似的控制机制，但是上层协议可以使用块流来提供这种优先级机制。比如说，一个视频直播服务器根据发送时间或者每一个消息的响应可能会选择丢弃一个缓慢的客户端的视频消息来保证客户端能及时接收到音频消息。

RTMP 块流包括它的带内协议控制消息并且提供了一种机制供高层协议加入用户控制消息。

5.1. 消息格式

将消息分割成块用来支持混流的消息格式取决于高层协议。如果我们创建一个块，消息格式应该（SHOULD）包含下面的字段。

时间戳：消息的时间戳。这个字段包括 4 个字节。

长度：消息载荷的长度。如果消息头不能忽略，它应该包括消息头的长度。这个字段在块头中占 3 个字节。

类型 ID：一些类型 ID 预留给协议控制消息使用。这些传送信息的消息由 RTMP 块流和上传协议共同处理。其它类型 ID 都供高层协议使用，这些类型 ID 对 RTMP 块流是不透明的。实事上，RTMP 块流任何地方都不会需要将这些值当作类型使用；所有（非协议的）消息都可以是同一种类型，或者应用程序可以使用这个字段来区分同步轨道而不是使用类型。这个字段占用块头的 1 字节。

消息流 ID：消息流 ID 可以是任意值。不同的消息流混流成相同的块流，块流基于它们的消息流 ID 分流。除此之外，对于块流而言，这个值是一个不透明的值。这个字段以小端面的方式占用块头 4 个字节。

5.2. 握手

RTMP 的连接开始于握手。握手内容不同于协议的其它部分；它包括三个固定大小的块而不是带头信息的变长块。

客户端（发起连接的端点）和服务器各自发送相同的三个块。为了演示，这三个块客户端发送的被记做 C0，C1，C2；服务发送的被记做 S0，S1，S2。

5.2.1. 握手顺序

客户端发送 C0 和 C1 块开始握手。

客户端必须（MUST）等接收到 S1 后才能发送 C2。

客户端必须（MUST）等接收到 S2 后才能发送其它数据。

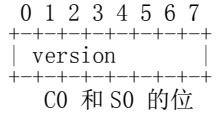
服务器必须（MUST）等接收到 C0 才能发送 S0 和 S1，也可以（MAY）等接到 C1 一起之后。

服务器必须（MUST）等到 C1 才能发送 S2。

服务器必须（MUST）等到 C2 才能发送其它数据。

5.2.2. C0 和 S0 的格式

C0 和 S0 包只有八个位，可以看成一个 8 位的整数。



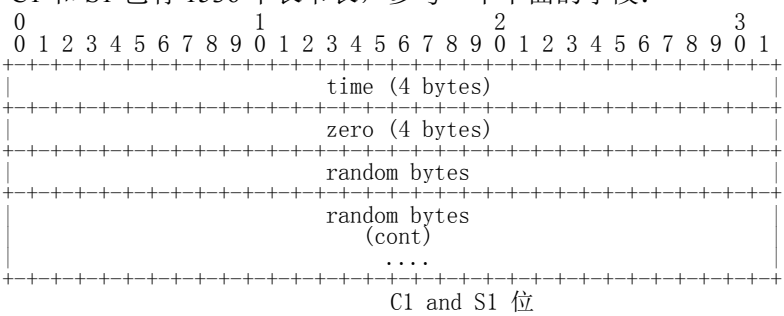
C0 和 S0 的位

下面 C0/S0 包的各个字段：

版本（8 bits）：在 C0 中，该字段标识了客户端请求的 RTMP 版本。在 S0 中，这个字段是服务器的 RTMP 版本。这个版本被定义成 3。0-2 用于早期的专利产品已经废弃了；4-31 被保留给未来的实现；32-255 是被禁止的（为了区分 RTMP 和基于文本的协议，基于文本的协议经常是以可打印字符开头）。如果不能识别客户端请求的版本，服务器应该（SHOULD）回复 3。客户端降低版本到 3 或者放弃连接。

5.2.3. C1 和 S1 的格式

C1 和 S1 包有 1536 个长节长，参考一下下面的字段：



C1 and S1 位

时间（4 字节）：这个字段包含了一个时间戳，它可能（SHOULD）作为所有本端的后续块的起始时间点。它可以是 0，也可以是任意值。为了同步多个块流，终端可能希望发送其它块流的当前时间戳的值。

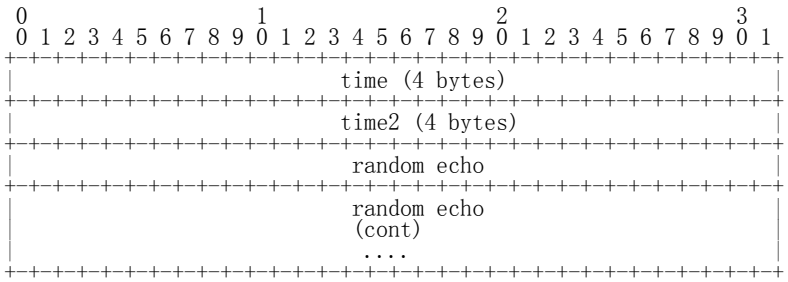
零（4 字节）：这个字段必需是全 0。

随机数（1528 字节）：这个字段包含了任意值。因为每一个终端都区分该数据是回复它自己发起的握手的还是对端发起的握手，这些发送的数据应该（SHOULD）足够的随机。

也不必是加密的随机数，或者是动态数据。

5.2.4. C2 和 S2 的格式

C2 和 S2 包有 1536 字节长，分别是 S1 和 C1 的回复，包含下面的域：



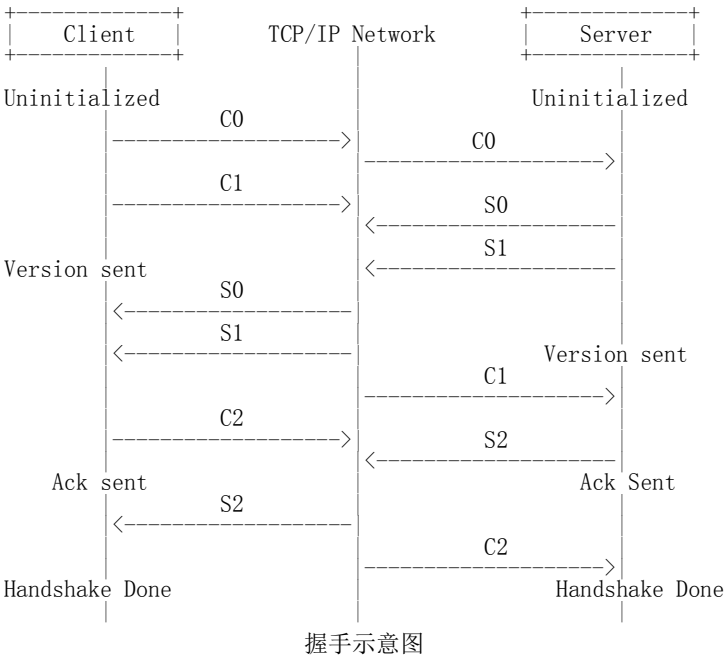
C2 和 S2 的位

时间（4 字节）：这个字段必须（MUST）包含了一个时间戳，它是由对端发送过来。对于 C2 来说是 S1，对于 S2 来说是 C1。

时间 2（4 字节）：这个字段必须（MUST）包含前一个对端发送过来的并被读取的包（S1 或者 C1）。

随机数（1528 字节）：这个字段必须（MUST）包含对端发送过来的随机数对 S1 来说是 C2，对于 S2 来说是 C1。两端都可以时间字段和时间 2 字段结合当前的时间来快速宽带和（或）连接的延迟，但这个方法不太可能很有用处。

5.2.5. 握手示意图



握手示意图

下面是握手图示中提到的各个阶段具体内容：

未初始化（Uninitialized）：协议的版本发送出去的这个阶段。客户端与服务器都是未初始化态。客户端在 C0 中发送协议版本。如果服务器支持这个版本，它将回应 S0 和 S1。如

果不支持，服务器将会采取适当措施的回应。在 RTMP 中，这个措施是终止连接。

版本已发送（Version Sent）：客户端和服务器在未初始化态后是版本已发送态。客户端等待 S1，服务器在等待 C1。在接收到响应包后，客户端发送 C2，服务器发送 S2。状态就变成了确认已发送。

确认已发送（Ack Send）：客户端和服务器分别在等 S2 和 C2。

握手完成（Handshake Done）：客户端与服务器可以交换消息了。

5.3. 分块

握手结束后，连接对一个或者多个块流进行合并。每个块流承载了一个消息流中的一种类型的消息。每一个块在创建时都有一个唯一的 ID，称为块流 ID。块通过网络进行传输。传输过程中，每一个块必须完全发送才能发送下一个块。在接收端，通过块流 ID 块被组装成消息。

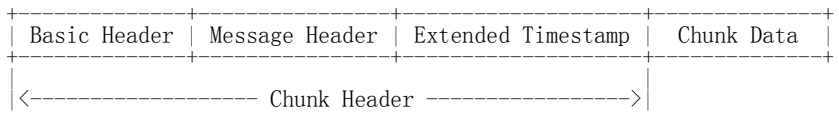
分块允许高层协议的大消息拆分成小消息，比方说为了防止大的低优先级消息（比如视频）阻塞小的高优先级的消息（比如说音频和控制）。

分块也允许用更低的开销发送小消息，例如原本位于消息内部的信息被在压缩在块头中。

块的大小是可配置的。它可以通过设置块大小（Set Chunk Size）的控制消息（5.4.1 节）。更大的块减少 CPU 的使用，但是会导致更多的写操作，这会使得在较借低的宽带下延迟它的内容。小的块对于高码率的流是不适合的。块流大小在每一个方向都是独立维护的。

5.3.1. 块格式

每一个块由块头和数据组成。块头由三个部分组成。



块格式

基本头（Basic Header，1 到 3 字节）：这个字段将对块流 ID 和块类型进行了编码。块类型标明了消息头的编码格式。字段的长度取决于块流 ID，因为块流是一个变长的字段。

消息头（Message Header，0、3、7 和 11 字节）：这个字段对发送的消息（部分或和全部）的信息进行了编码。这个字段的长度取决于块头的块类型。

扩展时间戳（Extended Timestamp，0 和 4 字节）：这个字段是否出现取决于块消息头中的时间戳（timestamp）或时间戳增量（timestamp delta）。更多信息见 5.3.1.3 节。

块数据（Chunk Data，可变大小）：这个块的有效负载，最大为配置的最大的块大小。

5.3.1.1. 块基本头（Basic Header）

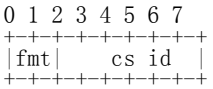
块头将块流 ID 和块类型进行了编码（下图中用 fmt 表示）。块类型决定了消息头的编码格式。块基本头字段可能是 1、2 或 3 字节，取决于块流 ID。

协议的实现应该（SHOULD）使用能够容纳 ID 的最小表示。

协议支持 ID 是 3-65599 之间的 65597 个流。ID 是 0、1 和 2 都是保留值。值为 0 表明了块基本头是 2 字节的形式并且 ID 是在 64-319 范围内（第二个字节+64）。值为 1 表明块基本是 3 字节的形式并且 ID 的范围在 64-65599 内（(第三个字节)*256+第二个字节+64）。值在 3-63 范围的块基本头则表示了完整的块流 ID。块流 ID 为 2 保留用做低层协议的控制消息和命令。

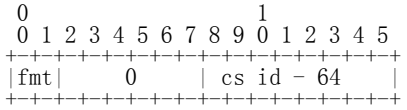
块基本头的 0-5 位（最低有效位）表示块流 ID。

块流 ID 在 2-63 之间编进 1 字节版本的基本头中的块流 ID 字段。



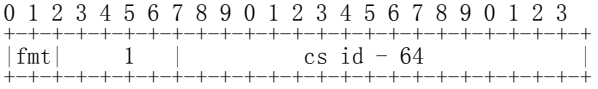
块基本头 1

块流 ID 在 62-319 之间的可以编码成块基本头的 2 字节的形式。ID 的计算方法是（第二个字节+64）。



块基本头 2

块流 ID 在 64-65566 之间的编进块基本头为 3 字节版本的字段。ID 的计算方法是（(第三个字节)*256+第二个字节+64）。



块基本头 3

cs id（6 位）：这个字段包含了块流 ID，值范围是 2-63。值为 0 和 1 用来表示这个字段是 2 或 3 字节的版本。

fmt （2 位）：这个字段标识了“块消息头”4 种格式中的 1 种。这个“块消息头”的每一种块类型将在下一节介绍。

cs id - 64（8 或 16 位）：这个字段包含了块流 ID 减去 64。比如说，ID 是 365 会在 cs id 用 1 表示和一个这里 16 位的 301。

块流 ID 在 64-319 之间可能是用 2 字节或 3 字节形式的头表示。

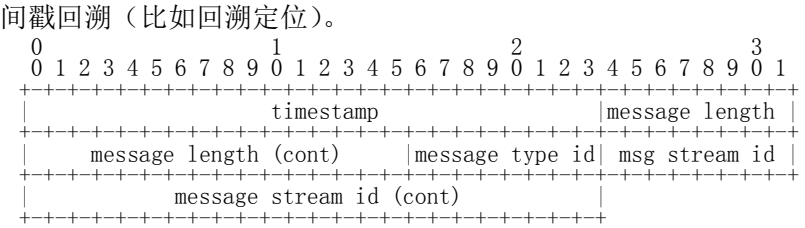
5.3.1.2. 消息头（Message Header）

有四种不能的块消息头，由块基本头中的”fmt”字段表示。

RTMP 的实现应该（SHOULD）尽可能为每一种块消息头使用最紧凑的表示。

5.3.1.2.1. 类型 0

类型为 0 块头有 11 字节长。这种类型必须（MUST）在块流开始的时候使用，不管是不是流的时间戳回溯（比如回溯定位）。

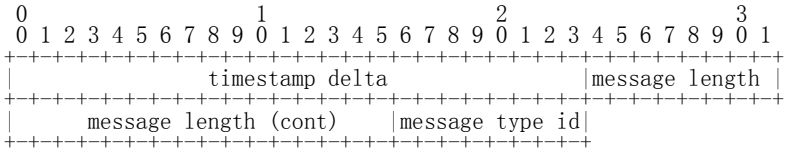


块消息头类型 0

时间戳（timestamp，3 字节）：对于类型为 0 块，这里发送的是绝对时间戳。如是时间戳大于等于 16777215（十六进制是 0xFFFFFF），这个字段必须（MUST）是 16777215，表明扩展时间戳（Extended Timestamp）字段使用全 32 位对时间进行了编码。否则，这个字段应该（SHOULD）表示完整的时间戳。

5.3.1.2.2. 类型 1

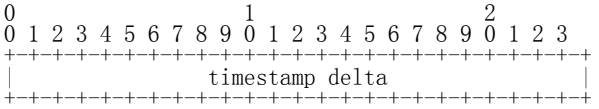
类型为 1 的块头有 7 个字节长。没有包含消息的流 ID，这个块使用了前一个块的流 ID。变长消息的流（比如很多视频格式）应该（SHOULD）在流的第一个块之后每个新消息的首个块之后使用这个格式。



块消息头类型 1

5.3.1.2.3. 类型 2

类型为 2 的块头有 3 个字节长。不包含流 ID 和消息长度，这个块与上一个块有相同的流 ID 和消息长度。消息有固定长度的流（比如说，一些音频和数据格式）应该（SHOULD）在流的第一个块之后每个消息的首个块之后使用这个格式。



块消息头类型 2

5.3.1.2.4. 类型 3

类型为 3 的块没有消息头。不包含流 ID、消息长度和时间增量字段，这种类型的块与前一个相同的流 ID 有相同的值。当单个消息被分解成多个块，所有的除了第一个之外的块应当（SHOULD）使用这种类型。参考例 2（5.3.2.2 节）。组成流的消息有相同的大小，流 ID 和时间应在类型为 2 的块后面该使用这个种类型。参考例 1（5.3.2.1 节）。如果第一个消息与第二消息之间的时间间隔与第一个消息块的时间戳相同，那么类型为 3 的块立即跟在类型为 1 的后面，而不需要类型为 2 的块来注册时间增量了。如果类型为 3 的块跟在类型为 0 的块后面，那么这个类型为 3 的块的时间戳增量与类型为 0 的块的时间戳一样。

5.3.1.2.5. 公共消息头字段

块的消息头的每一个字段的介绍：

时间戳增量（timestamp，3 字节）：对于类型 1 和类型 2 的块，这里发送的是先前块与当前块之间的差值。如果增量大于等于 16777215（十六进制是 0xFFFFFF），这个字段必须（MUST）是 167777215，表明扩展时间戳（Extended Timestamp）字段使用全 32 位的时间增量进行了编码。否则，这个字段应该（SHOULD）表示实际的时间增量。

消息长度（message length，3 字节）：对于类型为 0 或者类型为 1 的块，这里发送的是消息的长度。注意一般情况下，这与块有效负载的长度是不一样的。所有块的有效负载长度除了最后一块外都是块的额定最大长度，最后一块长度是消息的剩余长度（对于长度很短的消息，这可能是消息的全部长度）。

消息类型 ID（message type id，1 字节）：对于类型为 0 和类型为 1 的块，这里发送的是

消息类型。

消息流 ID（message stream id，4 字节）：对于类型为 0 的块，存储的是消息流 ID。消息流 ID 以小端（little-endian）格式存储的。通常情况下，所有块流相同的消息来自相同于的消息流。由于有可能将不同消息混合到相同的块流里，这使得不能有效地实施头部压缩。然而一旦现有的一个消息流关闭了，随后又打开了一个消息流，没有理由不能使用一个现有的块流，通过发送块类型为 0 的块使用复用那个块流。

5.3.1.3. 扩展时间戳（Extended Timestamp）

扩展时间戳字段用来对时间戳的编码或者是对那些时间戳大于 16777215 的时间增量的编码；这些是对于类型为 0，1，2 块的时间戳或者时间增量不能使用 24 位表示的。这个字段完整的表示了 32 位的时间戳或时间增量。类型 0 的时间戳的时间或类型 1，2 的时间戳增量设置为 16777215（0xFFFFFF）表明这个字段是有意义的。当这个类型为 3 的块的最近的相同块流 ID 的类型为 0，1 或 2 的块也使用这个字段，那么它的这个字段也是有意义的。

5.3.2. 实例（Examples）

5.3.2.1. 实例 1

这个实例演示了一个简单的音频流消息。实例展示了信息的冗余。

	Message Stream ID	Message Type ID	Time	Length
Msg # 1	12345	8	1000	32
Msg # 2	12345	8	1020	32
Msg # 3	12345	8	1040	32
Msg # 4	12345	8	1060	32

用于分块的音频消息的实例

接下来的表格展示了这个流产生的块。从消息 3 开始，数据传输被优化了。从这个点开始，每一个消息的开销只有 1 个字节。

	Chunk Stream ID	Chunk Type	Header Data	No. of Bytes After Header	Total No. of Bytes in the Chunk
Chunk#1	3	0	delta: 1000 length: 32, type: 8, stream ID: 12345 (11bytes)	32	44
Chunk#2	3	2	20 (3 bytes)	32	36
Chunk#3	3	3	none (0 bytes)	32	33
Chunk#4	3	3	none (0 bytes)	32	33

音频消息的每一个块的格式

5.3.2.2. 实例 2

这个实例演示了一个消息长度太长不能适应长度为 128 字节的块而被拆分成几个块。

	Message Stream ID	Message TYpe ID	Time	Length
Msg # 1	12346	9 (video)	1000	307

拆分成块的消息的实例

下面是产生的块：

	Chunk Stream ID	Chunk Type	Header Data	No. Of Bytes after Header	Total No. of bytes in the chunk
Chunk#1	4	0	delta: 1000 length: 307 type: 9, stream ID: 12346 (11 bytes)	128	140
Chunk#2	4	3	none (0 bytes)	128	129
Chunk#3	4	3	none (0 bytes)	51	52

每一个块的格式

块 1 的头数据整个消息的长度为是 307 字节。

从两个实例观察到，类型为 3 的块有两种不同的使用途径。第一个用于同一个消息的连续块（译者注：实例 2）。第二种是用来指示新消息的开始，这个消息的头继承于已现有的状态数据（译者注：实例 1）。

5.4. 协议控制消息

RTMP 块流使用类型为 1、2、3、5 和 6 的消息用于协议控制消息。这些消息包含 RTMP 块流协议需要的信息。

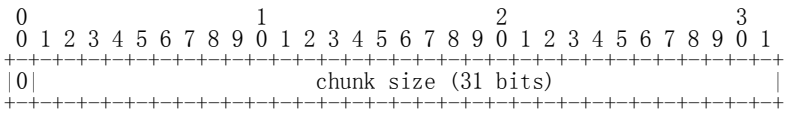
这个协议控制消息必须（MUST）使用 ID 为 0 消息流（称为控制流）并且用 ID 为 2 块流发送。协议控制消息在接收到时尽快处理。它们的时间戳忽略不用。

5.4.1. 设置块格式（Set Chunk Size（1））

协议控制消息 1，即设置块大小，被用来通知对端一个新的最大块大小。

最大块大小默认值为 128 字节，但是客户端和服务器可以改变这个值，并且通过个消息通知对方。比如说，假设客户端需要发送 131 字节大小的音频块并且块大小为 128 字节。这种情况下，客户端可以发送这个消息告诉服务器现在块大小为 131 字节。客户端接下来就可以使用单一个块来发送音频数据了。

块大小的最大值应该（SHOULD）至少 128 字节，必须（MUST）至少 1 个字节。块大小的最大值每一个方向是独立维护的。



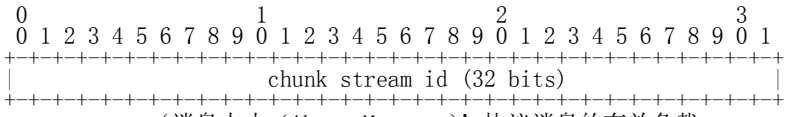
‘设置块大小（Set Chunk Size）’协议消息的有效负载

0：这个位必须（MUST）是 0。

chunk size（31 位）：这个字段包含了新的块大小的最大值，字节为单位，发送者随后的所有的块都使用这个值直到下一个关于设置块大小最大值的通知。合法的大小是 1 到 2147483647（0x7FFFFFFF），包含 1 和 2147483647；然而，所有大于 16777215（0xFFFFF）的大小都是等价的，因为没有块的比它所在的消息还要长，并且没有消息的长度是大于 16777215 字节的。

5.4.2. 消息中止（Abort Message（2））

协议控制消息 2，即中止消息，用来通知对端如果正在等待接收消息的块那么就丢弃已通过块流接收到的消息。对端通过这个控制消息的有效负载接收到块流 ID。应用程序在关闭的时候发送这个消息，表明不需要继续处理消息了。

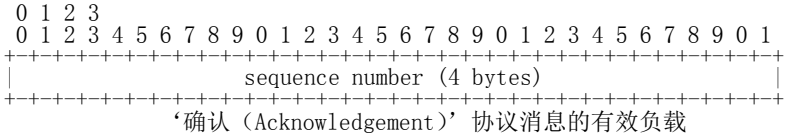


‘消息中止（Abort Message）’协议消息的有效负载

块流 ID（chunk stream ID，32 位）：这个字段包含了当前被丢弃的消息块流 ID。

5.4.3. 确认（Acknowledgement（3））

客户端和服务器必须（MUST）在收到字节数等于窗口大小后向对方向发送确认消息。窗口大小是发送者收到确认前而能发送的最大字节数。消息指定了序列号，这上序号是指到目前为止接收到的字节数。

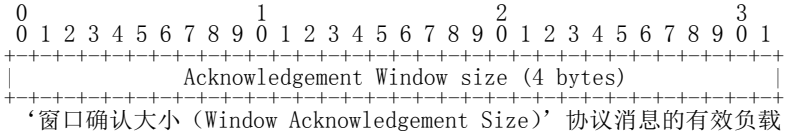


‘确认（Acknowledgement）’协议消息的有效负载

序列号（sequence number，32 位）：这个字段包含了目前为止接收到的字节数。

5.4.4. 窗口确认大小（Windows Acknowledgement Size（5））

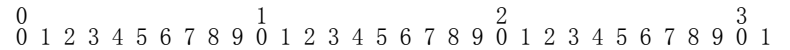
客户端或者服务器发送这个消息来告诉对方两次确认发送间的窗口大小。发送方在发送了窗口大小的字节数后期望收到对方的确认。接收方必须（MUST）在上一次发送确认之后接收到指定大小字节后发送一次确认（第 5.4.3 节），如果之前没有发送过确认，那么就从会话开始之后开始算发送确认。

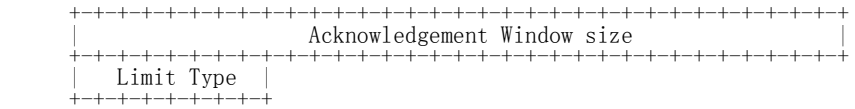


‘窗口确认大小（Window Acknowledgement Size）’协议消息的有效负载

5.4.5. 设置对端带宽（Set Peer Bandwidth（6））

客户端或服务器发送这个消息去限制对端的输出带宽。接收方收到这个消息后限制自己的输出带宽。除了尚未做出确认的消息外，接收方限制它的输出宽带为消息指定的窗口大小。如果上一个窗口大小与最后一收到的窗口确认大小不一样，这个消息的接收者应该（SHOULD）回复一个窗口确认大小的消息。





‘设置对端带宽（Set Peer Bandwidth)’协议消息的有效负载

限制类型取下面的值的其中一个：

0-Hard(硬)：对端应该（SHOULD）用指定的窗口大小限制自己的输出带宽。

1-Soft（软）：对端应该（SHOULD）使用指定的窗口大小限制自己的输出带宽，如果已经限制了的则取二者中小值。

2-Dynamic（动态）：如果先前的限制类型是是硬，那么把这个消息看成硬类型，否则忽略这个消息。

5.RTMP 消息格式

本节讨论 RTMP 消息的格式，使用低层的传输层在网络间传输的 RTMP 消息，比如，RTMP 块流。

由于 RTMP 设计成工作在 RTMP 块流之上的，所以它可以使用任意的传输层协议进行发送消息。RTMP 块流和 RTMP 配合适合各种各样的音视频程序，从一对一和一对多的直播到视频点播服务再到互动会议程序。

6.1. RTMP 消息格式

服务器和客户端通过网络发送 RTMP 消息相互进行通讯。消息可以包括音频，视频，数据，甚至其它任何数据。

RTMP 消息有分成两个部分，一个是头部，一个是有效负载。

6.1.1. 消息头

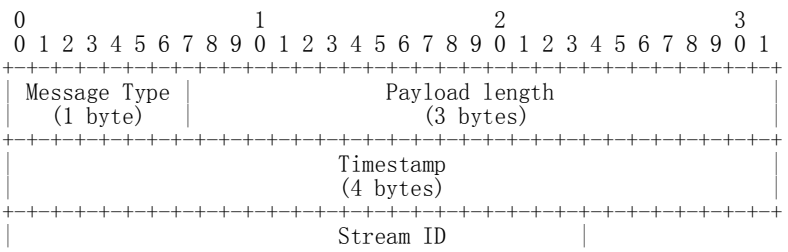
消息头由下面几个字段组成：

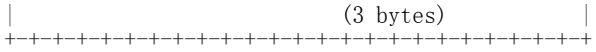
消息类型（Message Type）：字段占用 1 字节来表示消息类型。类型的 ID 范围在（1-6）保留给协议控制消息。

长度（Length）：字段占用 3 字节来表示有效负荷以字节为单位的长度。使用大端格式。

时间戳（Timestamp）：字段占用 4 字节来表示消息的时间戳。使用大端格式。

消息流 ID（Message Stream Id）：字段占用 3 字节来消息流的 ID。使用大端格式。





消息头

6.1.2. 消息有效负载

消息的其它部分就是有效负载，这是包含在消息中的实际数据。举例说，它可能是一些音频采样或是被压缩的视频数据。有效负载的格式和表示内容超出了本文档的范围了。

6.2. 用户控制消息（User Control Messages（4））

RTMP 使用类型 ID 为 4 的消息做为用户控制消息。这些消息包含了 RTMP 流层使用的信息。RTMP 块流协议使用 ID 为 1、2、3、5 和 6 的协议消息（5.4 节）。

用户控制消息应该（SHOULD）消息流 ID 0（称为控制流），并且通过 RTMP 块流发送，使用块流 ID 为 2。当接收方从流中接收到用户控制消息时，它应该马上生效；它们的时间戳忽略不用。

客户端或者服务器发送这个消息来通知对方用户控制事件。这个消息包含了事件的类型和事件数据。

‘用户控制（User Control）’协议消息的有效负载

消息数据的前 2 个字节用来指明事件类型。事件数据跟在事件类型的后面。事件数据的的长度是可变的。然而，这种情况下消息是通过 RTMP 块流层传送的，块的大小应该（SHOULD）足够大使用单个块可以空纳这类消息。

事件类型和事件数据的格式在 7.1.7 节列出。

6.RTMP 控制消息

本节描述了服务器和客户端相互通信进行交换的不同类型的消息和命令。

在客户端与服务器端相互交换的不能消息类型包括用来发送音频数据的音频消息，用来发送视频数据的视频消息，发送用户数据的数据消息，共享对象消息和命令消息。共享对象消息提供了一种通用的方法去管理多个客户端和一个服务器之间的分布式数据。命令消息传递客户端与服务器之间 AMF 编码的命令。客户端或服务器可以通过和对端通信流使用命令消息请求远程过程调用（Remote Procedure Calls，RPC）。

7.1. 消息类型

服务器和客户端通过网络发送消息进行相互通信。消息可以是任意类型的，包括音频消息，视频消息，命令消息，共享对象消息，数据消息和用户控制消息。

7.1.1. 命令消息（Command Message（20，17））

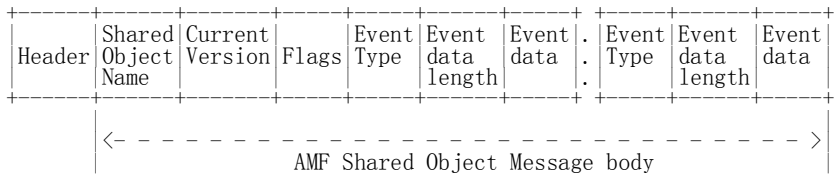
命令消息在客户端与服务器之间传递 AMF 编码的命令。消息类型 ID 为 20 用于表示 AMF0 编码，消息类型 ID 为 17 用于 AMF3 编码。这些消息执行对端上的一些操作比如 connect、createStream、publish、play 和 pause。命令消息如 onstatus、result 等等用来通知发送者请求命令的状态。一个命令消息由命令的名字、事务 ID 和包含相关参数的命令对象组成。一个客户端或服务器可以通过网络使用命令消息向对端请求远程过程调用（RPC）。

7.1.2. 数据消息（Data Message（18，15））

客户端用这个消息向对端发送 Metadata 或者任意的用户数据。Metadata 包函了数据的（音频、视频等）详细信息，像创建时间，时长，主题等等。这些消息使用消息类型 18 表示 AMF0，用消息类型 15 来表示 AMF3。

7.1.3. 共享对象消息（Share Object Message（19，16））

共享对象是一个 Flash 对象（一个键值对的集合），用来同步多个客户端，应用实例等等。消息类型为 19 表示使用 AMF0，16 保留用作 AMF3 编码共享事件。每一个消息都可能包含多个事件。



共享对象消息格式

支持下面类型的事件类型：

Event	Description
Use(=1)	客户端向服务器发送这个事件通知服务器创建指定名字的共享对象。
Release(=2)	当客户端在本端删除共享对象后向服务器发送个事件。
Request Change (=3)	客户端发送这个事件请求改变指定名字参数关联的共享对象的值。
Change (=4)	服务器向除事件创建者之外的所有客户端发送这个事件来通知它们指定名字的值发生了改变。
Success (=5)	如果客户端的 RequestChange 事件被接受了，服务器使用这个事件对请求的客户端做出回应。
SendMessage	客户端向服务器通过发送这个事件给服务器去广播一个

(=6)	消息。在接收到这个消息后，服务器会向所有客户端广播这个消息，包括发送者。
Status (=7)	服务器发送这个事件通知客户端错误状态。
Clear (=8)	服务器发送个消息给和客户端去清除一个共享对象。服务器也会使用这个事件回应客户端连接时的 Use 事件。
Remove (=9)	服务器发送这个事件给客户端让它删除一个插槽。
Request Remove (=10)	客户端删除一个插槽给服务器发送个事件。
Use Success (=11)	在客户端成功连接时，务器发送这个事件

7.1.4. 音频消息（Audio Message（8））

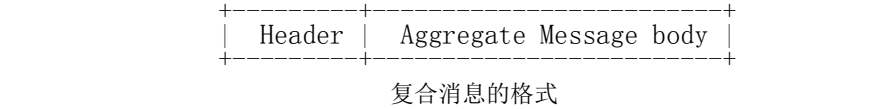
客户端或服务器发送这个消息来向对象发送音频数据。类型为 8 的消息保留给音频消息。

7.1.5. 视频消息（Video Message（9））

客户端或服务器发送这个消息来向对象发送视频数据。类型为 9 的消息保留给视频消息。

7.1.6. 复合消息（Aggregate Message（9））

一个复合消息是一个单个消息，它包括了一系列 6.1 节描述的那种 RTMP 子消息。类型为 22 的消息用作复合消息。



复合消息的格式

复合消息体的格式

复合消息的消息流 ID 覆盖了复合内部的子消息的消息流 ID。

复合消息与第一个子消息的时间戳之间的差值用来重新偏移的所有消息的时间刻度。该偏移量加到每一个子消息的时间戳上才是正常的流时间。第一个子消息时间戳应该（SHOULD）与聚合消息的时间戳相同，这样这个偏移量应当（SHOULD）是 0。

后向指针（Back Pointer 字段）包含了前一个消息包括它的头部的大小。包含这个字段的目的是用来匹配 FLV 文件和用来向前定位。

使用复合消息有几个性能优势：

- 块流可以使用一个块最多只能发送一个完整的消息。所以，只要增加块大小并且使用复合消息能减少块的发送数量。

- 子消息可能连续存储在内存中。当使用系统调用通过网络发送数据时会更高效。

7.1.7. 用户控制消息事件（User Control Message Events）

客户端与服务器发送个消息来通知对端用户控制事件。更多关于消息格式的信息，查看6.2 节。

下面是用户控制事件支持的类型：

Event	Description
Stream Begin (=0)	服务器发送这个消息通知客户端一个流变成就绪并且可用于通信。默认情况，在服务器从客户端接收到连接命令后，服务器会发送这个 ID 为 0 的事件。事件数据占 4 个字节，它表示变成就绪的流的 ID。
Stream EOF (=1)	服务器发送这个消息通知客户端请求流的回放数据已经结束。如果没有额外的命令，数据将不再发送。客户端丢弃来自该流的数据。事件数据占 4 个字节，它表示回放已经结束的流的 ID。
StreamDry (=2)	服务器发送个消息通知客户端流上没有更多的数据了。如果服务器在一段时间没有侦测到任何数据，它会通知订阅该了该流的客户端流已枯竭。事件数据占 4 个字节，它表示枯竭的流的 ID。
SetBuffer Length (=3)	客户端发送这个消息告知服务器通过流到达的任何数据的缓存大小（单位是毫秒）。这个事件在服务器开始处理流数据前发送。事件数据的前 4 字节表示流 ID,接下来的 4 字节表示缓冲区的大小（单位是毫秒）。
StreamIs Recorded (=4)	服务器发送这个事件去通知客户端这是一个录制流。4 个字节的事件数据表示了录制流的 ID。
PingRequest (=6)	服务器发送个命令给客户端去测试客户端的连通性。事件数据是 4 字节的时间戳，表示服务器分发这个命令时的本地服务器时间。客户端在接收到 PingRequest 请求时使用 PingResponse 回复。
PingResponse (=7)	客户端发送这个事件给服务器用来响应服务器的 ping 请求。事件数据是客户端接收到 PingRequest 请求时的 4 字节时间戳。

7.2. 命令类型

客户端与服务器交换使用 AMF 编码的命令。发送者发送一个由命令名称、事务 ID 和包括相关参数的命令对象组成的命令消息。例如，连接命令包含了‘app’参数，它告诉服务器客户端连接的应用程序的名字。接收者处理命令和发回相同事务 ID 的响应。响应串是_result、error 或者一个方法名字中的其中一个，比如 verifyClient 或 contractExternalServer。

命令串_result 和_error 是响应的标志。事务 ID 标明了响应所指向的命令。它与 IMAP 和其它协议中的标签一样。命令串中的方法名称指明发送者试图在接收端执行的方法。

下面类的对象用来发送多种命令：

NetConnection 表示服务器与客户端之间的上层的一个连接对象。

NetStream 表示发送音频流、视频流或其它相关数据的一个通道。我们也发送控制数据流动的命令如 play、pause 等。

7.2.1 NetConnection 的命令

NetConnection 管理着客户端应用与服务器之间的双路连接。并且，它提供异常的远程调用。

下面命令可以通过 NetConnection 发送：

- connect
- call
- close
- createStream

7.2.1.1. connect

客户端发送连接命令给服务器去请求连接服务器上的一个应用实例。

从客户端到服务器发送的这个命令的结构如下：

Field Name	Type	Description
命令名称	String	命令的名称. 设置成 "connect"。
事务 ID	Number	总是设置成 1
命令对象	Object	键值对的命令信息
可选用户参数	Object	任意可选的信息

下面是 connect 命令中命令对象中使用的键值对的描述信息。

Property	Type	Description	Example Value
app	String	客户端连接到服务器上的应用实例名称	testapp
flashver	String	Flash 播放器版本。它与 ApplictionScript getversion() 函数返回的串相同	FMSc/1.0
swfUrl	String	当前连接的 SWF 源文件地址	file://C:/FlvPlayer.swf
tcUrl	String	服务器的地址。它的格式如下。protocol://servername:port/appName/appInstance	rtmp://localhost:1935/testapp/instance1
fpad	Boolean	如果使用代理设置为 True。	true or false
audioCodecs	Number	表明客户端支持的音频编码	SUPPORT_SND_MP3

videoCodecs	Number	表明客户端支持的视频频编码	SUPPORT_VID_SORENSON
videoFunction	Number	表明支持的特殊的视频函数	SUPPORT_VID_CLIENT_SEEK
pageUrl	String	加载 SWF 文件的网页地址	http://somehost/sample.html
objectEncoding	Number	AMF 编码方法	AMF3

audioCodecs 属性的标志值。

Codec Flag	Usage	Value
SUPPORT_SND_NONE	原始声音，没有压缩	0x0001
SUPPORT_SND_ADPCM	ADPCM 压缩	0x0002
SUPPORT_SND_MP3	mp3 压缩	0x0004
SUPPORT_SND_INTEL	未使用	0x0008
SUPPORT_SND_UNUSED	未使用	0x0010
SUPPORT_SND_NELLY8	NellyMoser 压缩 8-kHz	0x0020
SUPPORT_SND_NELLY	NellyMoser 压缩 (5, 11, 22, and 44 kHz)	0x0040
SUPPORT_SND_G711A	G711A 声音压缩 (只适用于 FMS)	0x0080
SUPPORT_SND_G711U	G711U 声音压缩 (只适用于 FMS)	0x0100
SUPPORT_SND_NELLY16	NellyMouser 压缩 16-kHz	0x0200
SUPPORT_SND_AAC	高级音频编码(AAC)	0x0400
SUPPORT_SND_SPEEX	Speex 音频压缩	0x0800
SUPPORT_SND_ALL	所有 RTMP 支持的音频编码	0x0FFF

videoCodecs 属性的标志值。

Codec Flag	Usage	Value
SUPPORT_VID_UNUSED	废弃的值	0x0001
SUPPORT_VID_JPEG	废弃的值	0x0002
SUPPORT_VID_SORENSON	Sorenson Flash 视频	0x0004
SUPPORT_VID_HOMEBREW	V1 屏幕共享	0x0008
SUPPORT_VID_VP6 (On2)	On2 视频 (Flash 8+)	0x0010
SUPPORT_VID_VP6ALPHA (On2 with alpha channel)	带有 alpha 通道的 On2 视频	0x0020
SUPPORT_VID_HOMEBREWV (screensharing v2)	屏幕共享版本 2 (Flash 8+)	0x0040
SUPPORT_VID_H264	H264 视频	0x0080

SUPPORT_VID_ALL	所有 RTMP 支持的视频频编码	0x00FF

videoFunction 属性的标志值。

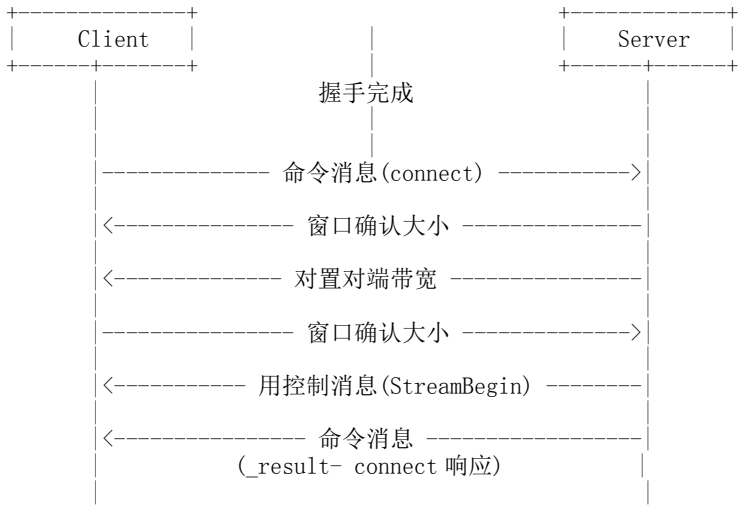
Function Flag	Usage	Value
SUPPORT_VID_CLIENT_SEEK	表示客户端可以进行帧定位。	1

对象的编码属性值:

Encoding Type	Usage	Value
AMF0	AMF0 对象编码可以被 Flash 6 及以后的版本支持	0
AMF3	Flash 9(AS3)起支持 AMF3 编码	3

服务器发送给客户端的命令结构如下：

Field Name	Type	Description
Command Name	String	_result 或 _error; 表明响应的是结果还是错误
Transaction ID	Number	1 表示 connect 响应的事务 ID
Properties	Object	连接的键值对属性（fmsver 等）
Information	Object	服务器响应的键值对的描述。'code', 'level','description'名字是这些信息一小部分。



connect 命令的消息流

命令执行期间消息流动如下：

1.客户端发送 connect 到服务器去与服务器的应用实例建立连接。

2.接收到连接命令后，服务器发送协议消息“窗口确认大小（Window Acknowledgement Size)”到客户端。服务器同时会连接到命令中提到的应用。

3.服务器发送协议消息“设置对端带宽（Set Peer Bandwidth)”到客户端。

4.客户端在处理完“设置对端带宽（Set Peer Bandwidth）后，发送协议消息“窗口确认大小（Window Acknowledgement Size）”到服务器。

5.服务器发送另外一个类型为用户控制消息（User Control Message（StreamBegin））的协议消息到客户端。

6.服务器发送结果的命令消息通知客户端连接状态（成功/失败）。这个命令指定了事务 ID（对于 connect 命令来说值为 1）。消息同时了指明了一些属性，如果 Flash Media Sever 的版本（字符串）。还有一些其它与连接相关的信息如 level（字符串）、code（字符串）、描述（字符串）、对象编码方式（数字）等等。

7.2.1.2. Call

NetConnection 对象的 call 方法运行在接收端的远程过程调用（RPC）。Call 命令传送了 RPC 的名字。

发送者给接收者发送的命令结构如下：

Field Name	Type	Description
Procedure Name	String	被调用的远程过程的名字。
Transaction ID	Number	如期望命令被回复我们指定一个事务 ID，否则我们传入 0 值。
Command Object	Object	如果有命令信息在这里设置，否则这里设置为空类型。
Optional Arguments	Object	提供任意的可选的参数。

回复的命令结构如下：

Field Name	Type	Description
Command Name	String	命令的名字。
Transaction ID	Number	响应所属的命令 ID
Command Object	Object	如果有命令信息在这里设置，否则这里设置为空类型。
Response	Object	调用方法的回复。

7.2.1.3. createStream

客户端发送这个命令去创建一个用于消息通信的逻辑通道。音频、视频和 metadata 都是通过使用 createStream 命令创建的流通道进行传输的。

NetConnection 是默认的通信通道，这个通道的流 ID 是 0。协议消息和少量的命令消息，包括 createStream，使用默认的通信通道。

Field Name	Type	Description
Command Name	String	命令的名称. 设置成 ″createStream″。

Transaction ID	Number	命令的事务 ID。
Command Object	Object	如果有命令信息在这里设置，否则这里设置为空类型。

服务器发往客户端的命令结构如下：

Field Name	Type	Description
Command Name	String	_result 或 _error; 表明响应的是结果还是错误
Transaction ID	Number	响应所属的命令 ID
Command Object	Object	如果有命令信息在这里设置，否则这里设置为空类型。
Stream ID	Number	返回值是一个流 ID 或者一个错误信息对象。

7.2.2 NetStream 的命令

NetStream 定义了通道，音频、视频、数据消息通过这个通道在客户端与服务器连接的 NetConnection 上进行流动。一个 NetConnection 对象支持多个 NetStream 用于多个数据流。

可以在客户端的 NetStream 发送到服务器的命令如下：

- play
- play2
- deleteStream
- closeStream
- receiveAudio
- receiveVideo
- publish
- seek
- pause

服务器使用“onStatus”命令发送 NetStream 状态来更新客户端：

Field Name	Type	Description
Command Name	String	命令名字是"onStatus".
Transaction ID	Number	事务 ID 为 0。
Command Object	Null	onStatus 没有命令对象
Info Object	Object	AMF 对象至少有下面的三个属性： “level”（字符串）：消息的等级取“warning”或“status”或“error”； “code”（字符串）：消息代码如“Netstream.Play.Start”； “description”（字符串）：消息的自然语言的描述。 这个 Info Object 字段可能 （MAY）包含其它适当的属性代码。

NetStream 的状态消息格式

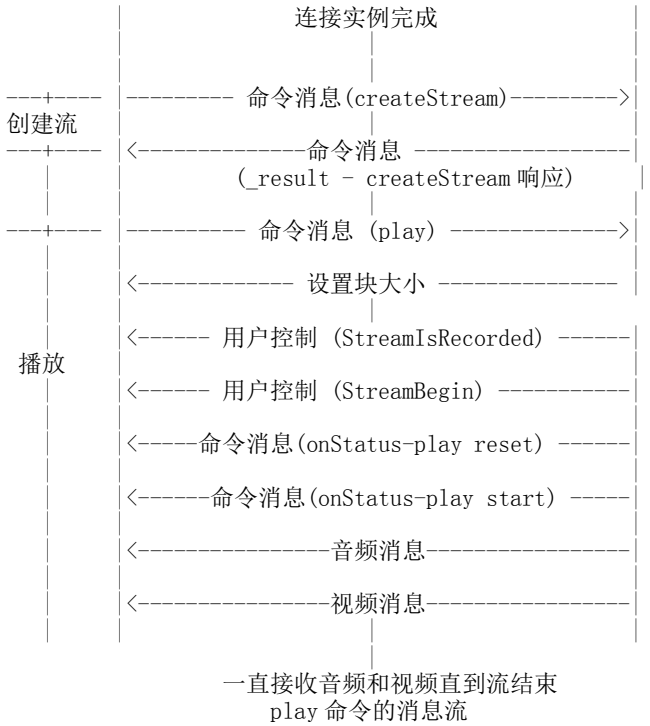
7.2.2.1. play

客户端发送个命令给服务器用以播放一个流。多次使用这个命令也可以创建一个播放列表。

如果你想创建一个在不同的直播流和录制流之间切换的动态播放列表，多次调用 play 并且每次 reset 都置为 false。相反地，如果你想立即播放某个流清除其它等待播放的流，置 reset 为 true。

从客户端到服务器传输的命令结构如下：

Field Name	Type	Description
Command Name	String	命令的名字，设置成"play"。
Transaction ID	Number	事务的 ID 设置为 0。
Command Object	Null	命令信息不存在。设置为空
Stream Name	String	需要播放的流的名字。 播放视频文件（FLV）需要指定不带后缀的流的名字（比如说，“sample”）。回放 MP3 或 ID3 标签，你必须在流名前面带上 mp3:（比如说，“mp3:sample”）。 播放 H.264/AAC 文件，你必须在流名前面带上 mp4，并且指定文件的扩展名。比如说，播放文件名 sample.m4v 文件， 使用“mp4:sample.m4v”。
Start	Number	一个可选的参数,它指定了起始的时间以秒为单位。默认的值是-2，它意味着本客户端播放的是 Stream Name 指定的直播流。 如该名字的直播流不存在，它将会播放相同名字的录制流。如果没有录制流，客户端将会等待该名字的流的创建并在可用的时候播放。如果将 Start 字段中传入-1，只会播放 Stream Name 字段指定的直播流。如是将 Start 字段传入 0 或者一个正数，将会从正数指定的位置开播放 Stream Name 字段指定的录制流。 如果没有录制流，将会播放播放列表中的下一个项目。
Duration	Number	一个可选的参数，它指定了回放的持续时间以秒为单位。默认值是-1，它意味着播放直播流直到不可用或播放录制流直到结束。 假定 Start 字段大于等于 0，如果传入 0 表示，只播放录制流中的 Start 字段指的起始位置的一帧。 如果你传入一个正数，它将在直播流可用时播放时长由 Duration 指定的直播流或者播放时长由 Duration 指定的录制流。(如果流在 Duration 指定的时间之前结束了，回放将在流结束的时候结束。) 如果你传入了一个除-1 之外的负数，它与-1 的意义是一样的。
Reset	Boolean	一个可选的布尔值或布尔数字，指定示是否刷新之前的播放列表。



play 命令的消息流

命令执行期间的消息流如下：

1.客户端在接收到来自服务器的 createStream 命令的成功结果后发送 play 命令。

2.在接收到 play 命令后，服务器发送协议数据来设置块大小。

3.服务器发送一些另外一个协议数据（用户控制），在这个消息里包含事件“StreamIsRecord”和流 ID。这个消息的前 2 个字节是事件类型随后的 4 字节是流 ID。

4.服务器向客户端发送另外一个协议消息（用户控制），这个消息指示了“StreamBegin”事件，表示流开始了。

5.如果客户端向服务器发送的 play 命令成功执行了，服务器会发送 onStatus 命令消息包含 NetStream.Play.Start 或 NetStream.Play.Reset。仅当客户端发送的 play 命令中的设置了 reset 标志 NetStream.Play.Reset 才会被发送。如果播放的流不存在，服务器会在发送 onStatus 消息包含 NetStream.Play.StreamNotFound。

随后，服务器就发送客户端播放的音频和视频数据。

7.2.2.2. play2

与 play 命令不同的是，play2 命令可以在不改变播放内容的时间轴的前提下切换不同码率的流。服务器为所有支持的码率维护多个文件，这样客户端就可以请求 play2 命令。

从客户端发往服务器的命令结构如下：

Field Name	Type	Description
Command Name	String	命令的名字，设置为“play2”。
Transaction ID	Number	事务 ID 为 0
Command Object	Null	命令信息不存储，设置为空。
Parameters	Object	一个 AMF 编码的对象，它的属性是描述 flash.net.NetStreamPlayOptions 的

		ActionScript 对象的公有属性

NetStreamPlayOptions 对象的公有属性在 ActionScript 3 手册中有描述[AS3]。
这个命令的消息流如下图所示。

play2 命令的消息流

7.2.2.3. deleteStream

当 NetStream 需要销毁 NetStream 会发送 deleteStream 命令。
从客户端发往服务器的这个命令结构如下：

Field Name	Type	Description
Command Name	String	命令的名字，设置为“deleteStream”。
Transaction ID	Number	事务 ID 为 0
Command Object	Null	命令信息不存在，设置为空。
Stream ID	Object	在服务器上需要销毁的流的 ID

服务器不回复任何响应。

7.2.2.4. receiveAudio

NetStream 通过发送 receiveAudio 消息去通知服务器要不要发送音频给客户端。

从客户端发往服务器的这个命令结构如下：

Field Name	Type	Description
Command Name	String	命令的名字，设置为“receiveAudio”。
Transaction ID	Number	事务 ID 为 0
Command Object	Null	命令信息不存在，设置为空。
Bool Flag	Boolean	true 或 false 表明是否接收音频

如果发送过来的 receiveAudio 命令的 bool 标志设置成了 false，服务器不做任何响应。如果这个标志被设置成了 true，服务器响应 NetStream.Seek.Notify 和 NetStream.Play.Start 的状态消息。

7.2.2.5. receiveVideo

NetStream 通过发送 receiveAudio 消息去通知服务器要不要发送视频给客户端。

从客户端发往服务器的这个命令结构如下：

Field Name	Type	Description
Command Name	String	命令的名字，设置为“receiveVideo”。
Transaction ID	Number	事务 ID 为 0
Command Object	Null	命令信息不存在，设置为空。
Bool Flag	Boolean	true 或 false 表明是否接收视频

如果发送过来的 receiveAudio 命令的 bool 标志设置成了 false，服务器不做任何响应。如果这个标志被设置成了 true，服务器响应 NetStream.Seek.Notify 和 NetStream.Play.Start 的状态消息。

7.2.2.6. publish

客户端向服务器发送 publish 命令用来发布带有名字的流。任何客户端都可以使用这个名字播放这个流和接收发布的音频，视频和数据信息。

从客户端发往服务器的这个命令结构如下：

Field Name	Type	Description
Command Name	String	命令的名字，设置为“publish”。
Transaction ID	Number	事务 ID 为 0
Command Object	Null	命令信息不存在，设置为空。
Publishing Name	String	发布的流的名称。

Publishing Type	String	发布的类型。设置为“live”、“record”或“append”。 record：发布流并且数据被录制在一个新文件中。这个文件被存储在服务器应用程序的子目录中。如果文件已经存在，它将会被覆盖。 append：发布流并且数据被追加在文件的后面，如果文件不存在，它将会被创建。 live：发布直播流，数据不会被录制。

服务器使用 onStatus 命令回复并标记了发布的起点。

7.2.2.7. seek

客户端发送 seek 命令去定位一个媒体文件或者播放列表的偏移量（毫秒为单位）。
从客户端发往服务器的这个命令结构如下：

Field Name	Type	Description
Command Name	String	命令的名字，设置为“receiveAudio”。
Transaction ID	Number	事务 ID 为 0
Command Object	Null	命令信息不存在，设置为空。
milliSeconds	Number	播放列表定位的毫秒数。

当定位成功时，服务器回复一个状态 NetSteam.Seek.Notify 消息。如果失败，它将返回一个_error 消息。

7.2.2.8. pause

7. 2. 2. 8. pause

客户端发送 pause 命令去告诉服务器暂停或继续播放。
从客户端发往服务器的这个命令结构如下：

Field Name	Type	Description
Command Name	String	命令的名字，设置为“pause”。
Transaction ID	Number	事务 ID 为0
Command Object	Null	命令信息不存在，设置为空。
Pause/Unpause Flag	Boolean	true 或者 false，表明是要暂停流还是继续播放。
milliSeconds	Number	流暂停或者继续播放的毫秒数。当流暂停时，这里是客户端当前的流时间。当回放流继续时，服务器只会发送时间戳比这个值的消息。

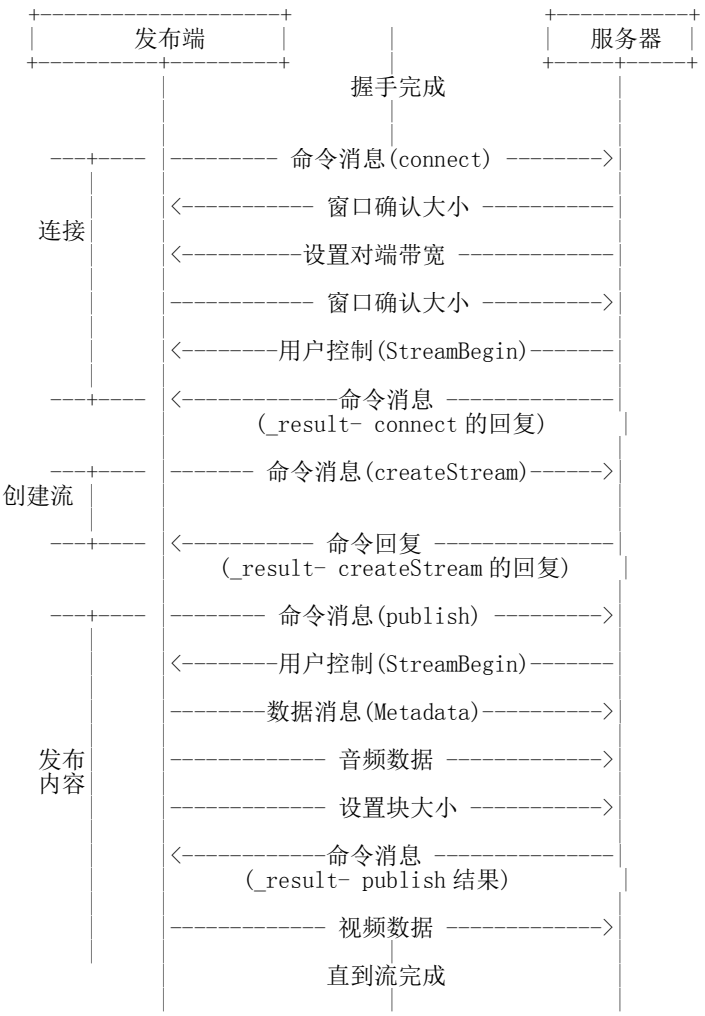
如果暂停成功，服务器会发送一个 NetStream.Pause.Notify 的状态消息。如果已经不在暂停状态将会发送 NetStream.Unpause.Notify 的状态消息。
如果操作失败，将返回_error 消息。

7.3. 消息交换实例

这里提供一点例子用来阐述使用 RTMP 消息的交换。

7.3.1. 发布录制视频

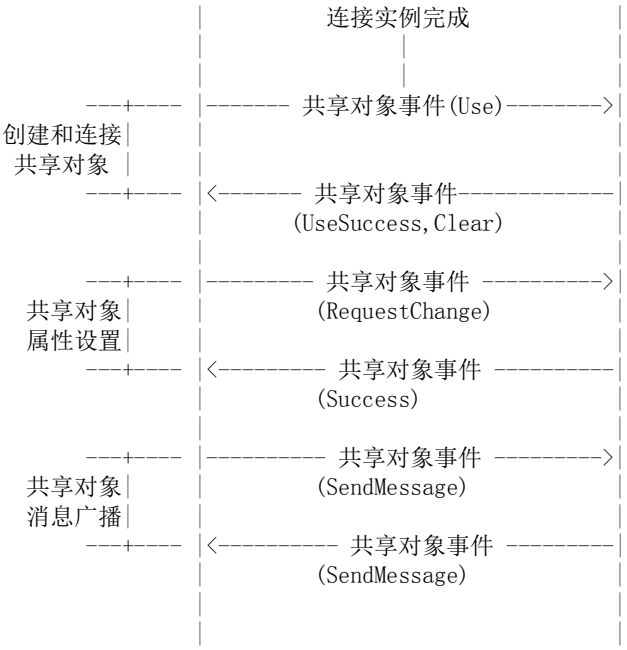
这个例子演示了发布端如何能发布一个流然后将它视频推送到服务器上。其它客户端可以订阅这个发布的流并且还可以播放这个视频。



发布一个视频流的消息流

7.3.2. 广播共享对象

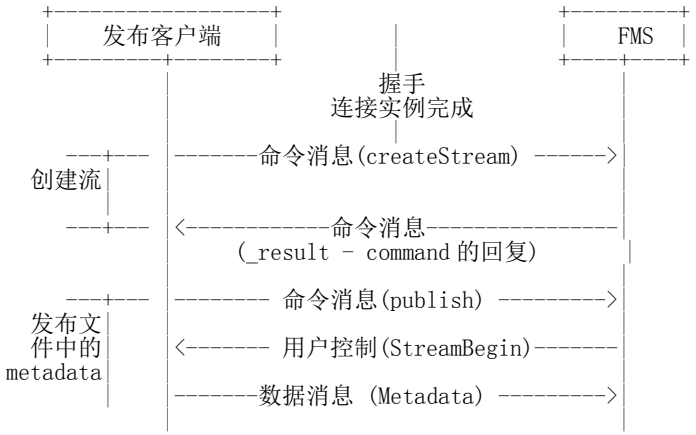
这个例子演示了共享对象的创建和改变的消息的交换。同时也演示了共享对象广播消息的过程。



共享对象的消息广播

7.3.3. 发布来自录制流的 Metadata

这个例子描述了发布 metadata 的消息交换。



发布 metadata

7.参考

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791,September 1981.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7,RFC 793, September 1981.

[RFC1982] Elz, R. and R. Bush, "序列号算法（Serial Number Arithmetic）", RFC 1982,August 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to IndicateRequirement Levels", BCP 14, RFC 2119, March 1997.

[AS3] Adobe Systems, Inc., "Adobe Flash 平台的 ActionScript 3.0 参考手册", 2011, <http://www.adobe.com/devnet/actionscript/documentation.html>.

[AMF0] Adobe Systems, Inc., "动作消息格式 -- AMF 0",December 2007, <http://opensource.adobe.com/wiki/download/attachments/1114283/amf0_spec_121207.pdf>.
[AMF3] Adobe Systems, Inc., "动作消息格式 -- AMF 3",May 2008, <http://opensource.adobe.com/wiki/download/attachments/1114283/amf3_spec_05_05_08.pdf>.

8.作者地址

Hardeep Singh Parmar (编辑)
Adobe Systems Incorporated
345 Park Ave
San Jose, CA 95110-2704
US
Phone: +1 408 536 6000
Email: hparmar at adobe.com
URI: http://www.adobe.com/

Michael C. Thornburgh (编辑)
Adobe Systems Incorporated
345 Park Ave
San Jose, CA 95110-2704
US
Phone: +1 408 536 6000
Email: mthornbu at adobe.com
URI: http://www.adobe.com/