H. Parmar, Ed.
M. Thornburgh, Ed.
Adobe
December 21, 2012

Adobe's Real Time Messaging Protocol

Abstract

This memo describes Adobe's Real Time Messaging Protocol (RTMP), an application-level protocol designed for multiplexing and packetizing multimedia transport streams (such as audio, video, and interactive content) over a suitable transport protocol (such as TCP).

Table of Contents

1. Introduction

Adobe's Real Time Messaging Protocol (RTMP) provides a bidirectional message multiplex service over a reliable stream transport, such as TCP [RFC0793], intended to carry parallel streams of video, audio, and data messages, with associated timing information, between a pair of communicating peers. Implementations typically assign different priorities to different classes of messages, which can effect the order in which messages are enqueued to the underlying stream transport when transport capacity is constrained.

This memo describes the syntax and operation of the Real Time Messaging Protocol.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this memo are to be interpreted as described in [RFC2119].

2. Contributors

Rajesh Mallipeddi, formerly of Adobe Systems, was the original editor of this specification, and provided most of its original text.

Mohit Srivastava of Adobe Systems contributed to the development of this specification.

3. Definitions

Payload: The data contained in a packet, for example audio samples or compressed video data. The payload format and interpretation are beyond the scope of this document.

Packet: A data packet consists of fixed header and payload data. Some underlying protocols may require an encapsulation of the packet to be defined.

Port: The "abstraction that transport protocols use to distinguish among multiple destinations within a given host computer. TCP/IP protocols identify ports using small positive integers." The transport selectors (TSEL) used by the OSI transport layer are equivalent to ports.

Transport address: The combination of a network address and port that identifies a transport-level endpoint, for example an IP address and a TCP port. Packets are transmitted from a source transport address to a destination transport address.

Message stream: A logical channel of communication in which messages flow.

Message stream ID: Each message has an ID associated with it to identify the message stream in which it is flowing.

Chunk: A fragment of a message. The messages are broken into smaller parts and interleaved before they are sent over the network. The chunks ensure timestamp-ordered end-to-end delivery of all messages, across multiple streams.

Chunk stream: A logical channel of communication that allows flow of chunks in a particular direction. The chunk stream can travel from the client to the server and reverse.

Chunk stream ID: Every chunk has an ID associated with it to identify the chunk stream in which it is flowing.

Multiplexing: Process of making separate audio/video data into one coherent audio/video stream, making it possible to transmit several video and audio simultaneously.

DeMultiplexing: Reverse process of multiplexing, in which interleaved audio and video data are assembled to form the original audio and video data.

Remote Procedure Call (RPC): A request that allows a client or a server to call a subroutine or procedure at the peer end.

Metadata: A description about the data. The metadata of a movie includes the movie title, duration, date of creation, and so on.

Application Instance: The instance of the application at the server with which the clients connect by sending the connect request.

Action Message Format (AMF): A compact binary format that is used to serialize ActionScript object graphs. AMF has two versions: AMF 0 [AMF0] and AMF 3 [AMF3].

4. Byte Order, Alignment, and Time Format

All integer fields are carried in network byte order, byte zero is the first byte shown, and bit zero is the most significant bit in a word or field. This byte order is commonly known as big-endian. The transmission order is described in detail in Internet Protocol [RFC0791]. Unless otherwise noted, numeric constants in this document are in decimal (base 10).

Except as otherwise specified, all data in RTMP is byte-aligned; for example, a 16-bit field may be at an odd byte offset. Where padding is indicated, padding bytes SHOULD have the value zero.

Timestamps in RTMP are given as an integer number of milliseconds relative to an unspecified epoch. Typically, each stream will start with a timestamp of 0, but this is not required, as long as the two endpoints agree on the epoch. Note that this means that any synchronization across multiple streams (especially from separate hosts) requires some additional mechanism outside of RTMP.

Because timestamps are 32 bits long, they roll over every 49 days, 17 hours, 2 minutes and 47.296 seconds. Because streams are allowed to run continuously, potentially for years on end, an RTMP application SHOULD use serial number arithmetic [RFC1982] when processing timestamps, and SHOULD be capable of handling wraparound. For example, an application assumes that all adjacent timestamps are within 2^31 - 1 milliseconds of each other, so 10000 comes after 4000000000, and 3000000000 comes before 4000000000.

Timestamp deltas are also specified as an unsigned integer number of milliseconds, relative to the previous timestamp. Timestamp deltas may be either 24 or 32 bits long.

5. RTMP Chunk Stream

This section specifies the Real Time Messaging Protocol Chunk Stream (RTMP Chunk Stream). It provides multiplexing and packetizing services for a higher-level multimedia stream protocol.

While RTMP Chunk Stream was designed to work with the Real Time Messaging Protocol (Section 6), it can handle any protocol that sends a stream of messages. Each message contains timestamp and payload type identification. RTMP Chunk Stream and RTMP together are suitable for a wide variety of audio-video applications, from one-to-one and one-to-many live broadcasting to video-on-demand services to interactive conferencing applications.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.