

March 28, 2024 by Red5 Team < <https://www.red5.net/author/red5-team/> >

Last updated July 6, 2026

# What Is RTSP Streaming and Why It Is Still Relevant in 2026



## TABLE OF CONTENTS

- What is RTSP?
- RTSP Technical Specifications
- RTSP Methods
- How Does RTSP Work?
- Benefits and Drawbacks
- 3 Reasons RTSP Streaming Is Still Relevant
- RTSP vs. Alternative Streaming Protocols
  - RTSP vs RTMP
  - RTSP vs WebRTC
  - RTSP vs SRT
  - RTSP vs MPEG-TS UDP Multicast
  - RTSP vs Zixi
- Red5 as Your RTSP Server

What is RTSP, and why is it still widely used in modern streaming workflows? RTSP, or Real Time Streaming Protocol, is a control protocol designed for managing real-time audio and video streams, especially from devices like IP cameras and encoders. In this guide, I will explain how RTSP works, where it fits in the streaming pipeline, and why it remains relevant in 2026.

## Conclusion

## What is RTSP?



The Real Time Streaming Protocol, or RTSP, is an application-level protocol for control over the delivery of data with real-time properties. It provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips. This protocol is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as UDP, multicast UDP and TCP, and provide a means for choosing delivery mechanisms based upon RTP (source: the official [2326 RTSP standard](https://datatracker.ietf.org/doc/html/rfc2326) < <https://datatracker.ietf.org/doc/html/rfc2326>> published in April 1998.)

Developed by experts from [RealNetworks](https://realnetworks.com/) < <https://realnetworks.com/>>, [Netscape](https://lsp.netscape.com/) < <https://lsp.netscape.com/>>, and [Columbia University](https://www.columbia.edu/) < <https://www.columbia.edu/>> around 1996, the protocol defines how the data in the stream should be packaged for delivery. It also defines how both ends of the connection should behave to prepare a pathway for transportation.

## RTSP Technical Specifications

To get a high-level understanding of this protocol's technical capabilities, security, and primary use cases, see the information below.

- **Video Codecs:** H.265 (HEVC), H.264 (AVC), VP9, VP8.
- **Audio Codecs:** AAC, AAC-LC, HE-AAC+ v1 & v2, MP3, Speex, Opus, Vorbis.
- **Variant Formats:** RTSP as an umbrella term describes the entire stack of RTP, RTCP (Real-Time Control Protocol), RTSPS (RTSP over SSL / Secure RTSP), and good-old RTSP.
- **Encryption:** Basic, less standardized. Native RTSP has no built-in encryption. RTSPS over TLS (port 322).
- **Playback Compatibility:** Limited support and rarely used for end-user playback.
- **CDN Friendly:** No.
- **Use Cases:** [IP camera ingest](https://www.red5.net/blog/ip-camera-live-streaming-rtsp-to-webrtc/) < <https://www.red5.net/blog/ip-camera-live-streaming-rtsp-to-webrtc/>>, CCTV, [surveillance](https://www.red5.net/solutions/streaming-for-video-surveillance-and-public-safety/) < <https://www.red5.net/solutions/streaming-for-video-surveillance-and-public-safety/>>, NVR (Network Video Recorder), DVR (Digital Video Recorder), streaming from [drones](https://www.red5.net/blog/drone-live-streaming/) < <https://www.red5.net/blog/drone-live-streaming/>> and robots.

## RTSP Methods

RTSP defines a set of case-sensitive methods <

<https://datatracker.ietf.org/doc/html/rfc2326#page-30>>

used to control operations on media resources identified by a Request-URI. Each method token specifies the action to be performed on a presentation (P), a stream (S), or both, and can be sent from client to server (C→S) or, in some cases, from server to client (S→C). Methods must follow the RTSP token format (they cannot start with the "\$" character), and the protocol is extensible, allowing new methods to be introduced beyond those defined in the specification. The standard methods and their behavior are summarized below.

| Method   | Direction | Description                                                                                                                                                                       |
|----------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPTIONS  | C→S, S→C  | Queries the server for supported RTSP methods and capabilities. Sent by both client and server. Required for capability negotiation and typically the first request in a session. |
| DESCRIBE | C→S       | Retrieves the session description (SDP), including codecs, bitrate, and transport parameters. Sent from client to server. Recommended for initializing playback.                  |
| ANNOUNCE | C→S, S→C  | Sends or updates the session description. From client to server, it describes the presentation; from server to client, it updates it. (optional)                                  |
| SETUP    | C→S       | Defines how media will be transported (e.g., RTP over UDP or TCP, port ranges) before playback begins. Sent from client to server. Required to establish the stream.              |
| PLAY     | C→S       | Starts media delivery by instructing the server to begin sending RTP packets. Sent from client to server. Required for playback.                                                  |
| PAUSE    | C→S       | Temporarily halts media delivery without terminating the session. Sent from client to server. Recommended for session control.                                                    |
| RECORD   | C→S       | Initiates recording of the media stream on the server. Sent from client to server and used in publishing workflows. (optional)                                                    |
| REDIRECT | S→C       | Instructs the client to connect to a different server by providing a new                                                                                                          |

|               |          |                                                                                                                                                   |
|---------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------|
|               |          | URL. Sent from server to client. (optional)                                                                                                       |
| TEARDOWN      | C→S      | Terminates the RTSP session and stops all associated media streams. Sent from client to server. Required for session cleanup.                     |
| GET_PARAMETER | C→S, S→C | Retrieves parameter values (such as session state or keepalive info). Can be sent by both client and server, often used for keepalive. (optional) |
| SET_PARAMETER | C→S, S→C | Sets or updates session parameters. Can be sent by both client and server. (optional)                                                             |

## How Does RTSP Work?

RTSP is a stateful control protocol used to manage media streaming sessions while separating signaling from media transport. The client communicates with the server over a persistent TCP connection, sending requests that define how the stream should be delivered, including the application type, requested media, and transport mechanisms such as unicast or multicast over UDP or TCP.

An RTSP session typically follows this sequence:

1. OPTIONS: The client queries the server to discover supported methods and capabilities.
2. DESCRIBE: The client requests an SDP (Session Description Protocol) file describing the media, including codecs, bitrate, and transport parameters.
3. SETUP: The client specifies how the media will be delivered, negotiating transport options such as RTP over UDP or TCP and defining port ranges.
4. PLAY: The server begins sending media data.
5. PAUSE / TEARDOWN: The client can temporarily halt or fully terminate the session.

RTSP itself does not transport media. Instead, audio and video are delivered separately using the Real-time Transport Protocol (RTP), typically alongside the Real-time Control Protocol (RTCP) for synchronization and quality feedback. Secure transport can be achieved using SRTP. In practice, RTSP protocol is most commonly used for first-mile delivery (ingest), such as streaming from IP cameras or encoders into a media server.

## Benefits and Drawbacks



RTSP remains a widely used protocol for ingesting real-time video, but like any technology, it comes with both strengths and limitations.

### Benefits of RTSP Streaming

- **Ultra-low latency streaming:** RTSP streams typically achieve latency under one second, making the RTSP protocol suitable for real-time monitoring and control use cases.
- **Native support in RTSP cameras:** Most IP cameras output an RTSP stream by default, making it easy to integrate RTSP camera feeds into a streaming workflow without additional configuration.
- **Flexible transport options:** RTSP supports both TCP and UDP for media delivery, allowing developers to optimize performance based on network conditions.
- **Efficient RTSP server workflows:** An RTSP server can manage multiple RTSP streams from cameras and encoders, making it a reliable choice for ingest pipelines.

### Drawbacks of RTSP

- **Limited playback support:** RTSP playback is not supported natively in modern browsers or most mobile devices, which requires conversion to other formats.
- **Not CDN-friendly:** RTSP streams cannot be distributed at scale using standard CDN infrastructure, limiting its use for large audiences.
- **Requires RTSP URL configuration:** To access a stream, you need a properly formatted RTSP URL, which can vary between devices and vendors.
- **Security depends on implementation:** RTSP does not include encryption by default, so secure transport requires additional protocols like RTSPS or SRTP.

## 3 Reasons RTSP Streaming is Still Relevant

### 1. Better for a Client-Server Model

Unlike WebRTC, RTSP is a little simpler to run as it does not perform all the signaling and NAT traversal techniques that WebRTC does. This is mainly because RTSP was designed to send and receive media streams from media servers in a client-to-server model, as opposed to WebRTC which was designed as a peer-to-peer protocol. With each WebRTC connection, you also have to maintain a separate signaling connection, whether that's WebSocket's or the new web standards WHIP and WHEP

<https://www.red5.net/blog/whip-and-whep-creating-simpler-faster-webrtc-connections/>. With RTSP, there's a single connection per subscriber/publisher client. Without the higher overhead of WebRTC, RTSP streams put less strain on the server and thus allows for more connections. We have detailed the WebRTC signaling process in [another post < https://www.red5.net/blog/webrtc-security-cors-dtls-srtp-omg/ >](https://www.red5.net/blog/webrtc-security-cors-dtls-srtp-omg/).

## 2. Many Supported Devices

The RTSP protocol provides for incredible cross-device compatibility.

### IP Cameras

Since IP cameras have been around since the 90s they were one of the earliest adopters of the Real Time Streaming Protocol for streaming, and thus they still continue to use it today. If it ain't broke, don't fix it. There are various uses for IP cameras, such as traffic monitoring for reporting or enforcement, security surveillance, and even home monitoring. Acting as a video streaming server to support multiple IP cameras ingest streams continues to be one of the most common use cases for Red5 Pro. This solution combined with Red5 Pro's highly scalable clustering model allows for virtually unlimited numbers of concurrent RTSP streams from many cameras.

### Other IoT Devices

**Drones** < <https://www.red5.net/blog/drone-live-streaming/> > enjoy widespread use expanding well beyond a backyard hobby. Smartphone or laptop controlled drones can be guided with the assistance of a live video feed (often an RTSP stream). Firefighters and U.S. Border patrol agents have used drones to conduct operations for **public safety** < <https://www.red5.net/solutions/drone-public-safety/> > and **defence** < <https://www.red5.net/solutions/air-gapped-live-streaming/> >. Additionally, aerial surveying is useful for maintaining infrastructure by examining power lines, and roads or even conducting geologic surveying. RTSP support is often built right into the drone software and is a common way to access a drone's video stream in near real-time.

When connecting the video stream from the drone to an RTSP compatible media server, or even a cluster of media servers, it's possible to scale the number of viewers of the RTSP stream to as many users as needed. In critical military or first responder use cases, this real-time one-to-many stream can be reviewed with other agencies, high ranking officials, and others to make critical decisions. Most current implementations of this use case involves a high latency stream, usually using high latency streaming HTTP based protocols like **HLS** < <https://www.red5.net/hls-server/> > (HTTP Live Streaming Protocol) which incurs many seconds of delay. An RTSP to **WebRTC** < <https://www.red5.net/webrtc-server/> > solution brings the latency down so that decision makers can make critical calls immediately since the video stream is being captured and transported in mere milliseconds.

A **great example** < <https://www.red5.net/case-studies/real-time-drone-streaming-in-san-diego/> > of this is the San Diego Sheriff's Department use of Red5 Pro with **Nomad Media** < <https://www.nomad-cms.com/> > for real-time drone streaming, allowing for a

cost effective solution deployed on AWS to remotely monitor fires, natural disasters, and other emerging situations.

### Robots

From underwater submersibles and radiation tests to bomb diffusing, robots are being created for a variety of video streaming applications. As a well-established solution with very low latency, robot-based computer systems will typically use the RTSP protocol for video delivery. The use of video allows the operator to control the robot and perform a wide variety of operations. Among the most impactful are medical robots such as remote surgery and telepresence robots, which enable doctors to communicate and work in isolated areas. Clearly, real-time latency, measured in milliseconds, is key to this use case.

As is the case with the first responder/military example mentioned above, scaling the single RTSP stream to many viewers can be an important aspect. The use of multiple RTSP servers set up in a cluster to fan out the delivery of the media stream to many viewers (other surgeons/experts) makes real-time collaboration possible.

Luckily, the RTSP protocol, < <https://www.red5.net/rtsp-protocol/>> with its ability to transport over UDP using secure RTP and connect to compatible media servers, means it is well-suited for these types of applications.

## 3. Ultra-Low Latency

By using the efficient RTP protocol, RTSP achieves a very low latency < <https://www.red5.net/red5-pro/low-latency-streaming-software/>>; well under 500 milliseconds when used with Red5 Pro. As RTP also forms the underlying protocol for WebRTC, most RTSP implementations are essentially a stripped-down version of WebRTC < <https://www.red5.net/blog/7-ways-webrtc-solves-ultra-low-latency-streaming/>>. One can get the same performance without the complexity.

In order to achieve this low latency, RTP sends video and audio data in small chunks suitable for quick transmission between the servers and clients. Each chunk of data is preceded by an RTP header; RTP header and data are in turn contained in a UDP packet. As each packet is processed, the following packets may already be in the stage of decompression or demultiplexing.

To cope with the occasional loss of packets (a hazard of Internet delivery in general), the RTP header contains timing information and a sequence number that allows the receivers to reconstruct the timing produced by the source. So if anything is out of order it can be quickly organized in proper order for playback.

The general structure of RTP consolidates essential information which in turn streamlines the process of media delivery. Thus it can achieve effective delivery of media streams with very low latency.

For a very detailed explanation of how RTP works please refer to our blog article on WebRTC < <https://www.red5.net/blog/6-ways-webrtc-solves-ultra-low-latency-streaming/>>.

[streaming/](#)>. Curious how RTSP stacks up against other protocols? Read our [RTMP vs RTSP blog <https://www.red5.net/blog/rtmp-vs-rtsp/>](#) to learn where each fits best in live streaming.

## RTSP vs. Alternative Streaming Protocols

RTSP is primarily used for first-mile delivery (ingest), so it only makes sense to **compare it with other ingest protocols**. Comparing RTSP to playback-focused technologies like [HLS <https://www.red5.net/blog/what-is-hls-streaming/>](#) or Media over QUIC ([MOQ <https://www.red5.net/media-over-quic-moq/>](#)) is misleading because they operate at different stages of the streaming pipeline. RTSP brings streams into your system, while those protocols deliver video to end users.

Below is a high-level comparison of RTSP and other common ingest protocols.

### RTSP vs RTMP

[RTMP <https://www.red5.net/blog/what-is-rtmp-streaming-protocol/>](#) is one of the most widely supported ingest protocols, especially for live streaming to platforms like YouTube, Facebook, and Twitch. While RTSP is commonly used by IP cameras and surveillance systems, RTMP is the default output for software encoders such as OBS, Wirecast, and vMix. RTMP runs over TCP and is easier to integrate with streaming platforms, but it typically introduces higher latency (250ms-3s) than RTSP (<100ms). Read a detailed comparison in our [RTMP vs RTSP <https://www.red5.net/blog/rtmp-vs-rtsp/#rtmp-vs-rtsp-specification-comparison>](#) guide.

### RTSP vs WebRTC

[WebRTC <https://www.red5.net/blog/what-is-webrtc/>](#) is designed for real-time, interactive streaming with sub 250ms latency and built-in browser support. Unlike RTSP, which is used for controlled ingest from devices like cameras, WebRTC can be used end-to-end, from ingestion to playback. However, ingesting via WebRTC often requires additional signaling and infrastructure compared to RTSP. These protocols are often used together, with RTSP for ingest and WebRTC for [ultra-low latency <https://www.red5.net/blog/what-is-ultra-low-latency-why-does-it-matter/>](#) delivery. Read a detailed comparison in our [RTSP vs WebRTC <https://www.red5.net/blog/rtsp-vs-webrtc-which-to-use-for-a-mobile-app/>](#) guide.

### RTSP vs SRT

SRT (Secure Reliable Transport) is optimized for streaming over unpredictable networks such as the public internet. It provides packet loss recovery, encryption, and adaptive retransmission, making it more reliable than RTSP in unstable conditions. While RTSP is still widely used in controlled environments like local networks and surveillance systems, SRT is gaining adoption for contribution workflows that require both low latency and reliability.

## RTSP vs MPEG-TS UDP Multicast

MPEG-TS over UDP multicast is used in broadcast and IPTV environments to efficiently deliver a single stream to multiple recipients on the same network. Unlike RTSP, which typically operates in a unicast client-server model, multicast reduces bandwidth usage by sending one stream to many viewers simultaneously. However, it requires specialized network infrastructure and is not suitable for public internet delivery.

## RTSP vs Zixi

Zixi is a proprietary protocol designed for high-quality contribution over managed and unmanaged networks. It includes advanced error correction, congestion control, and monitoring capabilities. Compared to RTSP, Zixi is more robust in large-scale broadcast workflows but requires licensed infrastructure and ecosystem support.

## Using Red5 as Your RTSP Server

Depending on your level of control and customization needs, you can use either of our real-time streaming products to ingest RTSP streams from encoders, NVRs, CCTV systems, [IP cameras < https://www.red5.net/blog/ip-camera-live-streaming-rtsp-to-webrtc/>](https://www.red5.net/blog/ip-camera-live-streaming-rtsp-to-webrtc/), or [drones < https://www.red5.net/blog/how-to-connect-drone-or-ip-camera-feeds-to-red5/>](https://www.red5.net/blog/how-to-connect-drone-or-ip-camera-feeds-to-red5/). If you're wondering [how to view an RTSP stream in a browser < https://www.red5.net/blog/how-to-view-an-rtsp-stream-in-a-browser >](https://www.red5.net/blog/how-to-view-an-rtsp-stream-in-a-browser), we've got you covered.

## 1. Red5 Cloud

*Watch the Red5 Cloud product overview on YouTube.*

If you want a fully managed, globally distributed streaming solution with an intuitive dashboard, **Red5 Cloud** <<https://www.red5.net/red5-cloud-low-latency-live-streaming-platform/>> is the easiest place to start. **Sign up** <<https://cloud.red5.net/signup>> with no credit card required and get the free tier every month:

- 50 GB of streaming.
- 6,000 instance hours.
- Hosting in US Central.
- Access to documentation and email support.

Once you reach the 50 GB monthly limit, you can:

- Continue streaming immediately with our usage-based **pricing** <<https://www.red5.net/red5-cloud-pricing/>> model, Pay-As-You-Grow, at \$0.08 per GB and \$0.69 per instance hour. No monthly minimums and no fixed commitments.
- Or wait until the next monthly cycle for your free 50 GB to reset.

Check out this **Youtube playlist** <[https://youtube.com/playlist?list=PLmLbmqsYGqWVfHFPYmwCykeO818ksthda&si=xrYL8bVQmCL-\\_uAe](https://youtube.com/playlist?list=PLmLbmqsYGqWVfHFPYmwCykeO818ksthda&si=xrYL8bVQmCL-_uAe)> to get started with Red5 Cloud.

## 2. Red5 Pro

If you need full customization, maximum flexibility, and complete control over your infrastructure, choose **Red5 Pro** < <https://www.red5.net/red5-pro/low-latency-streaming-software/> >, our server software built for ultra-low latency streaming at scale. **Start with a 30-day trial** < <https://account.red5.net/register> > and get access to all capabilities without limitations.

Red5 Pro is a great fit for high-stakes environments such as **public safety** < <https://www.red5.net/solutions/drone-public-safety/> > and **defense** < <https://www.red5.net/solutions/air-gapped-live-streaming/> >, government systems, **traffic monitoring** < <https://www.red5.net/blog/traffic-monitoring-systems/> >, and **smart cities** < <https://www.red5.net/solutions/smart-city-streaming/> >.

### Choosing the Right Setup

Not sure which solution is the best fit for your streaming challenges? Visit our **product comparison page** < <https://www.red5.net/red5-product-comparison/> > or **contact us** < <https://www.red5.net/contact/> > to discuss your project.

## Conclusion

What is RTSP in today's streaming landscape? It is still one of the most reliable protocols for ingesting real-time video from devices into streaming systems. While it is not designed for direct playback, RTSP remains essential for workflows that require low latency, device compatibility, and efficient first-mile delivery.

f


<https://www.facebook.com/red5pro/>

[Red5 Team < https://www.red5.net/author/red5-team/ >](https://www.red5.net/author/red5-team/)

<https://www.youtube.com/@Red5Pro>

<https://twitter.com/Red5Pro>




← [Previous post](#)

[H.264 vs H.265 vs VP9: Which Video Codec Should You Use in 2026?](#)

< <https://www.red5.net/blog/h264-vs-h265-vp9/> >

[Next post](#) →

[Server Release 12.4.0](#)

< <https://www.red5.net/blog/server-release-12-4-0/> >

## PRODUCT

### Red5 Cloud

Fully Managed PaaS Streaming Solution

< <https://www.red5.net/red5-cloud-low-latency-live-streaming-platform/> >

### Open Source

Red5 Media Server, Cordova Plugin, Load Testing Tool and more

< <https://www.red5.net/open-source-live-streaming/> >

Product Comparison < <https://www.red5.net/red5-product-comparison/> >

### Red5 Pro

Ultra-Low Latency Server Solution

< <https://www.red5.net/red5-pro/low-latency-streaming-software/> >

### Red5 SDKs

Android, iOS, Windows, macOS, Linux Unreal Engine, Unity and more

< <https://www.red5.net/live-streaming-sdks/> >

Professional Services < <https://www.red5.net/professional-services/>>

#### WHY RED5?

TrueTime Solutions < <https://www.red5.net/truetime/>>

Use Cases < <https://www.red5.net/solutions/>>

Demo < <https://www.red5.net/demo/>>

Case Studies < <https://www.red5.net/case-studies/>>

Partners < <https://www.red5.net/partners/>>

Patents < <https://www.red5.net/patents/>>

#### RESOURCES

Blog < <https://www.red5.net/blog/>>

Whitepapers < <https://www.red5.net/whitepapers/>>

Technology < <https://www.red5.net/video-streaming-technology/>>

Help Center < <https://customer.support.red5.net/servicedesk/customer/portals>>

#### GET STARTED

Red5 Pro Pricing < <https://www.red5.net/pricing/>>

Red5 Cloud Pricing < <https://www.red5.net/red5-cloud-pricing/>>

Documentation < <https://www.red5.net/docs/red5-pro-basics/redirects/>>

Red5 Pro 30-Day Free Trial < <https://account.red5.net/register>>

Red5 Cloud for Free < <https://cloud.red5.net/signup>>

Red5 Pro Login < <https://account.red5.net/login>>

Red5 Cloud Login < <https://cloud.red5.net/login>>

#### COMPANY

About Us < <https://www.red5.net/company/>>

Careers < <https://www.red5.net/company/#open-positions>>

Contact < <https://www.red5.net/contact/>>

Legal < <https://www.red5.net/legal/>>

FOLLOW US



<https://www.linkedin.com/company/red5-> <https://www.facebook.com/red5pro/> <https://twitter.com/red5pro/>

© 2005 - 2026 Infrared5, Inc.