Abstract

This memo describes an extension RTSP method, ANNOUNCE, which extends the core RTSP protocol to enable a RTSP protocol entity to push to the other side various types of information without the other side asking for it, as long as the other side has signaled the capability to support ANNOUNCE.

Examples of information in ANNOUNCE requests include session descriptions and end of stream events.

The receiver of this RTSP request is expected to reply with 200 OK response.

Since ANNOUNCE represents an extension to the core RTSP protocol, a feature tag, "method.announce", is defined to facilitate capability negotiation using RTSP extension mechanism.
1. Motivation

In [RFC2326], ANNOUNCE is part of the RTSP protocol. In the updated core RTSP protocol ([RTSP_NEW]), ANNOUNCE method is NOT part of the core RTSP protocol.

There are use cases that require ANNOUNCE method to convey session description as well as other event information. Hence there is a need to make the announce functionality still available but in a way consistent to the extension mechanism in [RTSP_NEW].

As an example, there is a need to signal end of stream event from server to client, as detailed below.

In the core RTSP protocol ([RTSP_NEW]), an RTSP client relies on the media transport mechanism to signal end of stream.

When the media transport mechanism happens to be RTP over UDP, this is carried out by RTCP BYE packet ([RTP_NEW]). In practice, there are some drawbacks with this approach:

1. When the server sends an RTCP BYE packet with its SSRC, the server is giving up on the SSRC (see section 8.2 in [RTP_NEW]). The server would be required to switch to a new SSRC on a subsequent PLAY of the same media stream. Since server’s SSRC is only communicated in the Transport header of SETUP response, the server would not have an opportunity to send a new value to the player, and the client would have to discover the SSRC from the incoming RTP packets -- a non-trivial process.

2. RTCP BYE packet method does not offer a simple, guaranteed method of delivering an end-of-stream announcement, given BYE packet is carried over UDP.

3. RTCP BYE packet method does not offer the option to have a single aggregate end-of-stream announcement for all media streams in the RTSP session.

4. Section 6.3.7 of RFC3550 stipulates that an RTP sender cannot send RTCP BYE before leaving the RTP session if it has not already sent at least one RTP or RTCP packet. This is a problem under error conditions. Consider the case where an RTP session has just started (i.e., RTSP PLAY has been successfully acknowledged with an RTSP 200 OK response), and the sender attempts to retrieve media frames from its media source. The media source fails to provide any media frame due to its internal error such as file corruption. The sender should inform its receiver(s) but it cannot send BYE packets.

The motivation to solve the above issues is particularly high for unicast-streaming applications that use RTSP over TCP in the control plane, and RTP over UDP in the media transport.
There is also the desire to have an EOS (End Of Stream) signaling mechanism for non-RTP delivery. One such delivery is MPEG2 transport streams used in the cable TV environment. In non-IP delivery environments, the transport typically remains allocated even if no media is being delivered. This means that a client cannot watch for the server to close the transport to signal the end of stream. Meanwhile, watching for the incoming media to stop is unreliable. Short timeouts can trigger a false end of media detection if the media flow is temporarily delayed. Long timeouts introduce unacceptable latencies. Clients are unable to distinguish between a normal end of stream and an error condition that resulted in the media delivery stopping.

We note that using TEARDOWN from server to client is not appropriate because:

1. TEARDOWN is currently not allowed from server to client [RTSP_NEW];
2. Even if TEARDOWN is made available in server to client direction, the definition of TEARDOWN requires that, if the request URI is aggregate, that the session must be de-allocated by the server. There are RTSP applications that use SET_PARAMETER from client to server as the means to report session QoS statistics, but if server uses TEARDOWN on aggregate URL to signal end of stream, the client can no longer use SET_PARAMETER with a session header.
3. In general, RTSP, being a client-server protocol, should let client, not server to control session state. But TEARDOWN on aggregate URL will change session from PLAYING state to INIT state.

We note that using PAUSE from server to client is not appropriate either, because PAUSE will change the state of the RTSP session.

We note that using SET_PARAMETER from server to client will require at least two parameters in the entity body, one for event type that should be set to End-Of-Stream, and the other parameter for the reason of End-Of-Stream. Also using SET_PARAMETER with an SDP entity body to update session descriptoin will not be compatible with RFC2326 where ANNOUNCE was defined for that purpose. Therefore, SET_PARAMETER is not appropriate to convey the announcement of End-Of-Stream and other events.

This memo proposes to define ANNOUNCE method to signal serveral event types including End-Of-Stream events.

The ANNOUNCE method is an RTSP request originally defined in RFC2326 but excluded from the core RTSP protocol due to lack of support [RTSP_NEW].

This memo defines ANNOUNCE as an extension to the core RTSP protocol [RTSP_NEW]. It presents ANNOUNCE method as a general mechanism for RTSP server to signal to its clients various events including end of stream events or session description updates events. This memo will discuss the general usage of ANNOUNCE, its feature tag, as well as well as a
new "Event-Type" header for ANNOUNCE method.

3. The Definition of ANNOUNCE method

[RTSP_NEW] has defined a mechanism to extend the core RTSP protocol. Following that mechanism, a feature tag is used to identify ANNOUNCE method as an extension to the core RTSP protocol.

The ANNOUNCE method is an RTSP request that can be sent in both directions, either from client to server or server to client. When server intends to send ANNOUNCE to client, it must have the means to reach the client, because the RTSP client is not required to keep a persistent connection with the RTSP server. It is beyond the scope of this memo to define the exact means for server to reach client. It suffices to say that if the client intends to receive server's ANNOUNCE requests, it must keep the RTSP connection open, or inform the server on how to reach it without a persistent RTSP connection.

Below is an example RTSP conversation in which an RTSP server announces an end of stream event for a media stream using a non-aggregate URI. The new header, "Event-Type" is to be defined later.

S->C: ANNOUNCE rtsp://foo.com/bar.avi/streamid=0 RTSP/1.0
    CSeq: 10
    Session: 12345678
    Event-Type: End-Of-Stream
    Range: npt=0-200
    RTP-Info: url=rtsp://foo.com/bar.avi/streamid=0;seq=45102
    Content-Type: text/parameters
    Content-Length: 49
    eos-reason: Reached the end of requested range.

C->S: RTSP/1.0 200 OK
    CSeq: 10
    Session: 12345678

3.1 Normative definitions

"ANNOUNCE" is an "extension-method" in the ABNF in section 16.2 "RTSP Protocol Definitions" in [RTSP_NEW].

The request-URI of an ANNOUNCE request can be either aggregate or non-aggregate URI.

An ANNOUNCE request must include "CSeq" header. It MAY include the following optional headers:

"Range",
"Session",
"RTP-Info",
"Event-Type"

An ANNOUNCE request MAY include entity body, in which case it MUST follow the rules for entity body defined in section 8.2 of [RTSP_NEW]. Entity body can be used to convey further details specific to an event type. For instance, if the event type is session description announcement, the actual SDP SHOULD be included in the entity body. If the event type is end-of-stream announcement, the entity body MAY contain "text/parameter" content type that conveys the reason of the end-of-stream
The next section defines a new RTSP headers for ANNOUNCE method: "Event-Type".

3.2 Event-Type Header

The Event-Type header is an optional header to identify the type of event pertaining to the ANNOUNCE request. Example event types include session description, end of stream and error.

If an ANNOUNCE request does not contain Event-Type header, the Event-Type defaults to "Session-Description", consistent with RFC2326.

The Event-Type header is defined in ABNF as:

```
Event-Type           = "Event-Type" ":" event-type
event-type           = "Session-Description" / "End-Of-Stream"
                     / "Error" / extension-event-type
extension-event-type = token
```

where:
-- token is defined in section 17 of [RTSP_NEW].

The only method that "Event-Type" applies is the ANNOUNCE method, either from client to server or from server to client.

3.3 Limitations on serve to client "ANNOUNCE" requests

Server to client ANNOUNCE method is issued only if the server has the means to contact the client when it has information to push. This may not be possible if the RTSP connection between server and client is not persistent. In such cases, the server will simply skip the sending of ANNOUNCE requests. That is to say, the server will not queue up the ANNOUNCE requests to be sent when client eventually connects. Such a queue would unnecessarily complicate server implementations.

4. Feature tag

The support of the ANNOUNCE method is represented by the feature tag below:

```
method.announce
```

This feature tag applies to both servers and proxies.

Implementations claiming "method.announce" feature tag MUST support the new "Event-Type" header defined in previous section.

5. Use Cases

This section presents several use cases of the ANNOUNCE method.

5.1 Client Announcing SDP To Server For Recording
This use case is the same as the first RTSP exchange presented in section 14.6 in RFC2326, with capability exchange via OPTIONS method.

The conference participant client C asks the media server M to record the audio and video portions of a meeting. After first verifying that the server supports the "ANNOUNCE" feature, the client uses the ANNOUNCE method to provide meta-information about the recorded session to the server. The client omits "Event-Type" because "Event-Type: Session-Description" is the default.

C→M: OPTIONS * RTSP/1.0
   Require: method.announce
   CSeq: 1

M→C: RTSP/1.0 200 OK
   CSeq: 1
   Supported: method.announce
   Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, RECORD, ANNOUNCE

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

5.2 Server Announcing End Of Stream

In this example, the server announces the End-Of-Stream event to client for one live media stream, because upstream source terminates the stream after 200 seconds. The fact that the stream has played for 200 seconds is communicated by the Range header in the ANNOUNCE request.

C→S: PLAY rtsp://foo.com/bar.avi/streamid=0 RTSP/1.0
   Supported: method.announce
   CSeq: 10
   Session: 12345678
   Range: npt=0-200

S→C: RTSP/1.0 200 OK
   Supported: method.announce
   CSeq: 10
   Session: 12345678
S->C: ANNOUNCE rtsp://foo.com/bar.avi/streamid=0 RTSP/1.0
  CSeq: 123
  Session: 12345678
  Range: npt=0-200
  RTP-Info: url=rtsp://foo.com/bar.avi/streamid=0;seq=45102
  Require: method.announce
  Event-Type: End-Of-Stream
  Content-Type: text/parameters
  Content-Length: 49
  eos-reason: reached the end of requested range.

C->S: RTSP/1.0 200 OK
  CSeq: 123
  Session: 12345678

The Require header in the above ANNOUNCE request indicates that in order to understand the Event-Type header the client must support the feature tag in the Require header. In this case the client happens to signal its support in its PLAY request.

From the ANNOUNCE request, the client will learn that the server has completed the stream as requested.

From the two RTP-Info headers, one in PLAY response, one in ANNOUNCE request, the client can derive the total number of RTP packets that the server has sent. From the npt field in the Range header, the client knows the presentation time that this stream has covered, from the server’s point of view.

6. Security Considerations

Because there is only one new TEXT header, "Event-Type", added by the extension RTSP method, the security considerations outlined in [RTSP_NEW] apply here as well.

7. IANA Considerations

A new method name, its associated feature tag, and a new header, need to be registered with IANA.

-- Method name: ANNOUNCE. See section 3.1 for the relevant definition.
-- Feature tag: method.announce. See section 4 for the relevant definition.
-- Header name: Event-Type, see section 3.2 for the relevant information.

Normative References


The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Copyright Notice

Copyright (C) The Internet Society (2004). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

This Internet-Draft expires in January 2005.

Acknowledgement

Thanks to Sean Sheedy for suggesting the inclusion of "Range" header and for contributing some text in the motivations section.

Funding for the RFC Editor function is currently provided by the Internet Society.

Author Addresses

Thomas Zeng
PacketVideo Network Solutions
9605 Scranton Road, Suite 400
San Diego, CA 92121
email: zeng@pvnetsolutions.com

P. Greg Sherwood
PacketVideo Device Solutions.
10350 Science Center Dr., Suite 210
San Diego, CA 92121
email: sherwood@pv.com