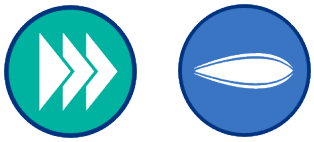

# Packet Ship *Streamline* Media Server

## Application Note
## AN-SL-605
v.3.1.2

## RTSP Profile for the Packet Ship Streamline Media Server

Documents ps-streamd v3.1.2

# Introduction

This application note describes the implementation of RTSP in the Packet Ship Streamline media server, and recommendations for client behaviour for optimum integration.

## Conformance to standards

The implementation of RTSP provided by the `ps-streamd` daemon is intended to follow the RTSP 1.0 standard (RFC 2326) as far as possible, while retaining compatibility with as many existing clients as possible, which may have been originally implemented and tested against other servers. It is not always possible to square this circle perfectly, and in some respects the server is forced to act contrary to the standard, and in some ways more like an RTSP 1.1 implementations.

### Non-conformance to RFC 2326

The main differences between the Packet Ship implementation and 'pure' RTSP 1.0 as defined in RFC 2326 are as follows:

- Like most real-world servers, Packet Ship does not implement queued PLAY requests, nor scheduled PLAY requests (in this respect it behaves more like RTSP 1.1)
- Packet Ship does not currently provide HTTP-like authentication (although a range of security measures based on IP and MAC addresses are available)

### Optional parts of RFC 2326 not implemented

The optional parts of RFC 2326 which are not implemented are as follows:

- SMTPE time stamps are not supported (NPT format only)

### Extensions to RFC 2326

The following are extensions not found in RFC 2326:

- Following a draft proposal pre RTSP 1.1, the Packet Ship RTSP server provides support for ANNOUNCE from the server at the end of a stream (see below).
- Variant DESCRIBE data can be returned depending on the Content-Type requested.

### RTP Implementation

The Packet Ship RTP implementation includes only the outgoing RTP datagram. RTCP is not implemented in either direction.

Within the RTP header:

- The payload type is globally configurable
- The SSRC is randomly generated for each stream
- The timestamp is set to real time (in 90Khz increments)

 

## General streaming

At its core, the Packet Ship server is format independent, and can stream any format at a constant bit-rate as raw UDP datagrams or RTP packets.

The server offers more advanced functionality for MPEG-2 Transport Streams:

### Variable Bit Rate

The server can use PCR values in Transport Stream packets to estimate the transient rate between any two PCR values, and hence provide Variable Bit Rate support. This requires that the file is 'indexed' with `ps-index-mpeg2ts` (see Application Note AN-SL-603).

### Trick mode

The `ps-index-mpeg2-ts` utility can also analyze Transport Streams containing MPEG-2 or H.264 video to create an index file which locates the independent video frames (I-frames), removing all the audio, plus an index that allows random-access seek into either the index or the main asset at an access point (I-frame boundary). This allows the Packet Ship video server to create 'trick mode' visual fast forward and rewind effects and smooth seeks.

## RTSP Connections

As mandated by the RTSP/1.0 specification, RTSP protocol connections are quite separate from sessions. Hence a client may connect and disconnect multiple times while the session remains active, or run several sessions from the same connection. However most clients in practice maintain a single connection for each session, throughout playback.

This separation can be deliberately broken if no other way to detect death of a client is available – see 'Keepalives' below.

## DESCRIBE

The Packet Ship RTSP server implements DESCRIBE to provide any textual format (but usually standard SDP) as a DESCRIBE response, with configurable insertion of asset-specific variables such as length, rate etc.

To handle clients which require non-standard Content-Types returned, multiple DESCRIBE blocks can be configured with different Content-Types, and the first one that matches the options given by the client will be returned. Variant DESCRIBE text can be also be chosen on the basis of a pattern match on the asset ID, for example enabling RTP only for H.264.

DESCRIBE responses are not well standardised, and we recommend that new clients do not use DESCRIBE at all. Metadata such as the type, length and rate of the asset should be available from an external source, the SETUP response (for duration), or from the stream data itself.

 

## SETUP

The RTSP SETUP command conforms to the RTSP/1.0 standard:

### URL

The URL quoted in SETUP should be absolute, as per the standard, but server-relative paths are in fact accepted. The 'path' of the URL with any trailing slash removed is the asset id requested – e.g.

```
rtsp://10.0.0.1/casino-royale/
```

requests asset ID 'casino-royale'. Any non-trailing slashes in the path are passed straight into the asset ID – hence the ID is treated as an opaque string for lookup in the master daemon's asset lists.

Any fragment identifier (after '#') in the URL provides an NPT offset to seek to – e.g.

```
rtsp://10.0.0.1/casino-royale#3600
```

Seeks to 3600 seconds into the stream. A second part can also be given to select a specific end point – e.g.

```
rtsp://10.0.0.1/casino-royale#3600-3900
```

### Transport header

The mapping of RTSP Transport header type to internal protocol can be changed in rtspd.cfg.xml, but the standard mapping for raw UDP is to accept either of "RAW/RAW/UDP" or "MP2T/H2221/UDP", for compatibility with other servers. For new clients, we recommend "RAW/RAW/UDP", because it is more descriptive and general. Only the first client_port value is used, since raw UDP only requires a single destination port.

For RTP, the server accepts "RTP/AVP/UDP" or "RTP/AVP" by default. Again, only the first client_port value is used, because RTCP is not implemented.

For RTSP/TCP (interleaved data and commands on the RTSP connection), the server accepts "RAW/RAW/UDP/TCP" or "RTP/AVP/TCP" by default.

### Immediate play

Note that unlike some other servers the Packet Ship RTSP implementation does not provide extensions to SETUP to start playback immediately, but this can be configured globally to do so. This is turned off by default, but can be enabled if clients require it.

### Session header

The RTSP server will append a 'timeout' parameter to the Session header to inform the client of the keepalive timeout in use (see below). It also accepts the (incorrect) reflection of this in subsequent commands by clients that have not parsed it out from the session ID itself.

### Range header

The response to a SETUP request also contains a Range header, which contains 0 as the first value, and the end-point (either the natural duration or a specifically requested one, if earlier) as the second value.

 

# PLAY

The RTSP PLAY command takes immediate effect (more like RTSP/1.1 than RTSP/1.0), and queuing is not supported.

## Range header

Only NPT times in Range headers are supported. The first value in a Range: header seeks the location to seek to, and the second (if present) sets the end point. The value 'now' is supported in the first value to indicate the current position (no seek), as is the absence of a Range header altogether. An absent endpoint value means “play to the natural end of the stream”.

The response to a PLAY request also contains a Range header, which contains the current play offset as the first value, and the current end-point (either the natural duration or a specifically requested one, if earlier) as the second value.

## Scale header

The Packet Ship server implements 'trick mode' (visual fast-forward and rewind) by creating a parallel 'index' file of the asset, as described above. The index allows a continuous range of different fast-forward/rewind speeds to be chosen, from at least 32x rewind to 32x fast-forward. The speed is set with the 'Scale' header in a PLAY request. The 'Speed' header is not supported.

The server dynamically adjusts the proportion of frames output to exactly match the speed requested while holding the rate at the normal average rate for the asset. Hence any speed, even fractional ones, can be obtained.

If a Scale header is given in the request, it will be reflected in the response. Only if trick mode is requested when no index file is available will it be different to that requested.

## Stream timing values

By default, the server updates both the stream PCR and PTS values and Transport Packet continuity counters to retain the impression of a continuous stream even if it is comprised of multiple assets with different timebases, or is looping.

## Transitions

If an index file is present for an asset, the server can ensure that transitions between play, pause and trick mode, and also seeks during play, occur at I-frame boundaries. The server will locate the next I-frame boundary using the index, and continue playing in the current speed until it is reached. Then it will immediately switch to the new location / speed and continue from there. The effect at the client is a seamless jump at an I-frame boundary, with a perfect match of Transport Stream packets at the join.

# PAUSE

The RTSP PAUSE command takes immediate effect. No Range header is supported in the request, however the current NPT offset and end-point are returned in a Range header in the response, as with PLAY.

 

## GET-PARAMETER / SET_PARAMETER

The RTSP GET-PARAMETER / SET_PARAMETER commands are supported mainly as a mechanism for keepalives (see below).

GET_PARAMETER also supports “position” and “duration” parameters, which return the current NPT and total play time of the session, respectively. If both are requested, the position is always returned first. SET_PARAMETER with a position is ignored and *cannot* be used to seek – use PLAY with a Range: header instead.

## TEARDOWN

The RTSP TEARDOWN command takes immediate effect. The stream will stop without regard to alignment to an I-frame boundary. If alignment is required, it is recommended to issue a PAUSE first, then wait a second before issuing the TEARDOWN.

## ANNOUNCE

To announce to the client that a stream has ended naturally, rather than because of a server or network failure, the server can send an ANNOUNCE command back to the client. Note that RTSP is a bidirectional protocol in this sense, unlike HTTP, and this functionality may confuse clients which are not expecting it (it can however be disabled).

The exact header sent can be configured. The default comes from a pre RTSP/1.1 proposal which we believe is supported by other servers:

```
Notice: 2101 End of Stream
```

There is also an option (enabled by default) to send an empty packet at the end of the stream, to enable clean shutdown in VLC.

## Keepalive

Clients – particularly embedded ones - are prone to disappear without warning, either because of loss of power or network problems. To stop 'orphan' streams continuing forever if a client does not tear them down properly, the server is configured by default to require some action from each stream on a regular basis – a 'keepalive' event – otherwise it will stop the stream itself.

The usual convention for such keepalive events is to issue either a PLAY with the current Scale and no Range, or an empty GET_PARAMETER or SET_PARAMETER. The latter is recommended. The timeouts are configurable, but using the default of a 60 second timeout, a keepalive every 30 seconds is suggested.

Ideally, the client should use the 'timeout' parameter from the server's Session header to determine the timeout in use, and send keepalives at least as often. Note that we recommend the timeout advertised by the server is less than the actual timeout in use (they are configurable independently), in case the client opts to send keepalives exactly at that interval, which could lead to unexpected

 

termination. Given this it is wise for the client to send keepalives slightly less than the timeout stated, but not as much as twice as often.

If an existing client is used which does not support regular keepalive events, then the timeouts can be disabled, but in this case it is strongly recommended to enable 'disconnect teardown' so that sessions can be stopped on disconnect. If this is not done there is no way to close down orphan sessions, and if playing a looped asset they may continue forever.

## Summary of recommendations

The following is a brief summary of recommendations for client developers:

- Follow RFC2326 in general
- Use content encapsulated in MPEG-2 Transport Streams
- Avoid use of DESCRIBE – try to obtain metadata externally, from the SETUP response or from the stream itself. If you absolutely require it, use SDP as the format.
- Use RAW/RAW/UDP , RTP/AVP/UDP or RTP/AVP/TCP as Transport in SETUP
- Do not expect queued PLAY commands to work
- Use an empty SET_PARAMETER as a keepalive at an interval slightly less than that stated in the 'timeout' parameter of the server's Session header from SETUP, or 30 seconds if fixed.
- Expect an ANNOUNCE command back at the end of stream

The Packet Ship server is designed to be as flexible as possible and to make it easy for clients to be integrate with little or no changes, but in difficult cases where clients cannot be modified we are also able to help by providing work-rounds in the server. If you are struggling with a client integration, please contact support@packetship.com, and we will endeavour to help.

 