Tunneling QuickTime RTSP and RTP over HTTP

QuickTime 4.1 adds HTTP to its' Streaming Transport capabilities. The addition of HTTP streaming allows QuickTime to utilize HTTP (RFC 1945 Hypertext Transfer Protocol 1.0 and RFC 2068 1945 Hypertext Transfer Protocol 1.1) proxies so viewers behind a firewall can access QuickTime presentations.

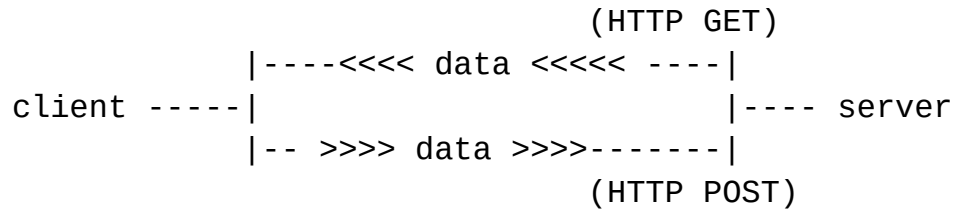
The HTTP transport is built from two separate HTTP GET and POST requests initiated by the client. The server then binds the connections to form a virtual full-duplex connection.
This paper documents the protocol used forming this type of connection, and notes implementation details associated with its use.

Protocol Requirements

- It must support use of unmodified RTSP/RTP.
- It must be acceptable to HTTP proxies.
- It must identify the requests and replies as non-cacheable by a proxy.
- All requests must originate from the client.
- It must uniquely identify the request pairs so that they may be successfully bound by the server.
- All requests must be made in such a way that they connect to the same RTSP server. This can be a challenge when the server is behind a DNS round robin for example.
- Any request must be identifiable as one that will eventually tunnel an RTSP conversation and RTP data.

Relationship to RTSP/RTP

Using standard RTSP/RTP it is possible to stream a presentation to a user via a single TCP connection. (See RFC 2036 "Real Time Streaming Protocol (RTSP)", section 10.12) Unfortunately, that is not sufficient to reach a significant population of Internet users. These users are typically on private IP networks where the client machines have indirect access to the public Internet via email and HTTP Proxies.
The QuickTime HTTP transport exploits the capability of HTTP GET and POST methods to carry an indefinite amount of data in their reply, and message body respectively. In the most simple case, the client makes a HTTP GET request to the streaming server to open the server to client channel. Then the client makes a POST request to the server to open the client to server channel.



The virtual connection makes it possible to use unmodified RTSP over the connection.

HTTP Requirements

Version – All requests are made using HTTP version 1.0.

Binding the Channels – Each client HTTP request must include a “x-sessioncookie” header with a Globally Unique Identifier (GUID) as it’s value. This makes it possible for the server to unambiguously bind the 2 channels. This protocol uses the value as a simple opaque token passed to the Clib 'strcmp' function.

Caching - Server replies include "Cache-Control: no-store", and client requests include the "Pragma: no-cache" as directives in the headers. These are not required by the protocol. It is recommended that implementations

x-server-ip-address Header - The server may optionally include a "x-server-ip-address" followed by an IP address in dotted quad format in it's reply to the client's initial GET request. When this header is present, the client must direct its’ POST request to the stated IP address, regardless of the IP address returned via DNS lookup.

This insures that both requests will connect to the same server among potentially several servers in a server farm. Many times these server clusters will be allocated

connections via a round robin DNS or other load balancing algorithms.

Meaning of HTTP errors in replies - HTTP errors reflect the unwillingness or inability of the server to form the virtual full duplex connection. They do not imply RTSP errors. In general, the server will always reply with “200 OK”, and simply close the connection if problems occur. The POST request is never replied to by the server

Request Identification

QuickTime Streaming uses the 'application/x-rtsp-tunnelled' MIME type in both the Content-Type and Accept headers. This reflects the data type that is expected and delivered by the client and server.

RTSP Request Encoding

The RTSP requests made by the client on the POST channel must be encoded using the base64 method. (See RFC 2045 “Internet Message Bodies”, section 6.8 Base64 Content-Transfer-Encoding; and RFC 1421 “Privacy Enhancement for Electronic Mail,” section 4.3.2.4, Printable Encoding)
The base64 encoding prevents HTTP proxies from mistaking the embodied RTSP requests as malformed HTTP requests. Our testing revealed that some HTTP proxies did in fact fail to pass the raw RTSP data properly.

Encoding a DESCRIBE request.

Original request:

```
DESCRIBE rtsp://tuckru.apple.com/sw.mov RTSP/1.0
CSeq: 1
Accept: application/sdp
Bandwidth: 1500000
Accept-Language: en-US
User-Agent: QTS (qtver=4.1;cpu=PPC;os=Mac 8.6)
```

Encoded format:

```
REVTQ1JJQkUgcnRzcDovL3R1Y2tydS5hcHBsZS5jb20vc3cubW92IFJUU1AvMS4w
DQpDU2VxOiAxDQpBY2NlcHQ6IGFwcGxpY2F0aW9uL3NkcA0KQmFuZHdpZHRoOiAx
NTAwMDAwDQpBY2NlcHQtTGFuZ3VhZ2U6IGVuLVVTDQpVc2VyLUFnZW50OiBRVFMg
KHF0dmVyPTQuMTtjcHU9UFBDO29zPU1hYyA4LjYpDQoNCg==
```

Sample HTTP Requests

Sample GET transaction.

Client ⇶ Server

```
GET /sw.mov HTTP/1.0
User-Agent: QTS (qtver=4.1;cpu=PPC;os=Mac 8.6)
x-sessioncookie: tD9hKgAAfB8ABCftAAAAAw
Accept: application/x-rtsp-tunnelled
Pragma: no-cache
Cache-Control: no-cache
```

Server ⇶ Client

```
HTTP/1.0 200 OK
Server: QTSS/2.0 [v101] MacOSX
Connection: close
Date: Thu, 19 Aug 1982 18:30:00 GMT
Cache-Control: no-store
Pragma: no-cache
Content-Type: application/x-rtsp-tunnelled
```

Sample POST request

Client ⇶ Server

```
POST /sw.mov HTTP/1.0
User-Agent: QTS (qtver=4.1;cpu=PPC;os=Mac 8.6)
x-sessioncookie: tD9hKgAAfB8ABCftAAAAAw
Content-Type: application/x-rtsp-tunnelled
```

```
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 32767
Expires: Sun, 9 Jan 1972 00:00:00 GMT
```

Server ⫩ Client

There is no reponse from the server! The client will continue to send RTSP as the message body of this POST request. The chosen Content-Length of 32767 is an arbitrarily large value. All POST requests are required to have a content-length header by HTTP. In practice the actual value seems to be ignored by proxy servers, so it is possible to send more than this amount of data in the form of RTSP requests. The QuickTime Server ignores the content-length header.

Connection Maintenance

The client may at its' option close the POST connection at any given time. Doing so frees socket and memory resources at the server that might otherwise be unused for a long time. In QuickTime Streaming the best time for this usually occurs after the PLAY request.

Implementation Best Practices

This section covers details that are not required by the protocol, but communicate important information to proxies so that they handle the streams better.

Date Header – In QuickTime streaming, the GET reply includes a "Date" headers with a value at an arbitrary point in the past. This keeps proxies from caching the transaction.

Expires Header – Same as the Date header.

Pragma "No cache" - Tells many HTTP 1.0 proxies not to cache.

Cache-Control "no-cache" – Tells HTTP 1.1 proxies not to cache.

Unsupported HTTP

Support for HTTP features not documented here is not required for RTSP tunneling. HTTP tunnel should mimic a normal TCP connection as closely as possible without adding unnecessary features.