Microsoft RTSP Streaming Protocol website http://sdp.ppona.com

MS - RTSP Protocol Version 1.0

By Paul @ SDP Team

>>> Start of History Date 7.Dec.2003

About RTSP.

Real Time Streaming Protocol is a streaming protocol that closely resembles that of HTTP protocol. Both use plain text for sending header and body data and the syntax of RTSP closely resembles that of standard host HTTP. RTSP was designed this way from the outset to maintain close compatibility with HTTP syntax allowing code written for HTTP to be used for RTSP also. That's the good news.

The main difference is that HTTP protocol is stateless, connections are broken between commands and one command does not rely on another. RTSP protocol on the other hand requires a known state between commands i.e. a command sent follows another command received in sequence. RTSP connections are not broken between states.

HTTP protocol uses port 80 as a default whereas RTSP uses port 554. Some proxies and firewalls may have problems with RTSP if the server administrator closes these ports for security reasons. Admin operators are required to allow port 554 for RTSP to operate.

RTSP is not used exclusively by Microsoft!

It is an open specification that has been implemented by many streaming server designers. Even Linux servers are using RTSP together with Mac programmers and Real Networks streaming media. It seems the whole world is starting to embrace this protocol for network streaming. However, MS does have its own 'flavour' of RTSP.

Microsoft and RTSP.

At the time of writing this document Microsoft are in the process of changing over from MMS to RTSP as the first choice of streaming protocol. This means that from Media Player version 9.0 onward and with streaming server version 2003 or later, we will see RTSP being used as the main streaming protocol of choice.

Slowly as time goes by, we may see MMS being phased out completely. It is only assumed that this is so MS can say they are being open with their protocols (rtsp is an open standard) while at the same time disregarding the need to publicise their own MMS protocol once its gone from media player. However, MMS is not dead yet, and
we could still see it being used for streaming over the next couple of years at least.

Is Microsoft's version of RTSP different from standard open versions of RTSP?

Yes. There are slight modifications from the original RTSP protocol as defined in RFC2326 (April 1998). There are now much later versions of this RTSP document and a simple search on the web will direct you to these sources of information.

What is different?

The main change in the MS specification is the way packet payloads are delivered to the client. There are some other changes to the command request messages but the individual media packet delivery system is undocumented (as far as I know) and may well be proprietary to MS.

Rather like MMS and HTTP 1.0 streaming protocols using a media Pre-Header, RTSP uses one also. But the MS version of pre-header is unspecified in any rtsp document. This is the main problem when trying to communicate using the MS version of RTSP.

The actual syntax of MS - RTSP commands do follow that specified in the open version of the RTSP protocol document. There are small changes and additions but basically you can see how to construct commands by reading these readily available documents from the web. The command part is well documented.

A Typical RTSP protocol session exchange
This example excludes packet pair delivery for simplicity.

To Server =>	NETWORK	<= To Client
Client connects to port 554 on server.		
Client Sends “DESCRIBE” command		
		Server replies with standard rtsp header. This header and body contains the ASF header and all stream bit rate data associated with the media file requested.
Client Sends “SETUP” command with audio stream selection (stream 1)		
		Server replies with standard rtsp header.
Client Sends “SETUP” command with video stream selection (stream 2)		
		Server replies with standard rtsp header.
Client Sends “PLAY” command		
		Server replies with standard rtsp header.
Client Sends “SET_PARAMETER” command This command includes logging data to the server regarding the client operating system, CPU type, player version, transport, time and date etc. Format is tagged XML.		
		Server replies with standard rtsp header.
		#### Server then starts to stream the media packets with pre-headers. See below ####
		An EOF indication is sent when the stream ends from the server to client.
		Server disconnects at socket level.

A Typical RTSP Command Sent To Server

DESCRIBE rtsp://wm.microsoft.com/ms/video/0001-hi.wmv RTSP/1.0
User-Agent: WMPlayer/9.0.0.2980 guid/3300AD50-2C39-46C0-AE0A-81D88F547805
Accept: application/sdp
Accept-Charset: UTF-8, *;q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Accept-Language: en-GB, *;q=0.1
CSeq: 1
Supported: com.microsoft.wm.srvppair, com.microsoft.wm.sswitch, com.microsoft.wm.eosmsg, com.microsoft.wm.predstrm

Notes:
DESCRIBE rtsp://wm.microsoft.com/ms/video/0001-hi.wmv RTSP/1.0
This is the connecting server location followed by the RTSP version (1)

User-Agent: WMPlayer/9.0.0.2980 guid/3300AD50-2C39-46C0-AE0A-81D88F547805
This shows what the client player is and its version along with a unique GUID.

X-Accept-Authentication: NTLM, Digest, Basic
What authentication types are available from the client.

Note that tag CSeq starts at 1. A reply to this command would also be Cseq: 1 to show that this was the command replied to.
Successive commands from the client to server are incremented by 1 for every command issued.

A Typical RTSP response header. They closely resemble those used in HTTP.
An example of a response from the DESCRIBE command is shown below:

<<<HEADER START>>>

RTSP/1.0 200 OK
Content-Type: application/sdp
Vary: Accept
X-Playlist-Gen-Id: 27006
X-Broadcast-Id: 0
Content-Length: 3324
Date: Tue, 18 Nov 2003 15:57:05 GMT
CSeq: 1
Server: WMServer/9.0.0.3372
Supported: com.microsoft.wm.srvppair, com.microsoft.wm.sswitch, com.microsoft.wm.eosmsg, com.microsoft.wm.fastcache, com.microsoft.wm.packetpairssrc
Last-Modified: Tue, 18 Jun 2002 21:05:39 GMT
Cache-Control: x-wms-content-size=23180160, max-age=86399, must-revalidate, proxy-revalidate
Etag: "23180160"

<<<BODY START>>> (indicated by \r\n\r\n).
<<< this is Session Description Protocol >>>

v=0
o=- 200311171721060249 200311171721060249 IN IP4 127.0.0.1
s=<No Title>
c=IN IP4 0.0.0.0
b=AS:1091
a=maxps:13632
t=0 0
a=control:rtsp://wm.microsoft.com/ms/video/0001-hi.wmv/
a=etag:{9D7121C3-1A1B-8ED6-6675-CB15D19D1FB7}
a=range:npt=3.100-173.903
a=recvonly
a=pgmpu:data:application/x-wms-contentdesc,8,language,31,0,,5,title,31,0,,6,author,31,0,,9,copyright,31,0,,35,WMS_CONTENT_DESCRIPTION_DESCRIPTION,31,0,,30,WMS_CONTENT_DESCRIPTION_RATING,31,0,,44,WMS_CONTENT_DESCRIPTION_SERVER_BRANDING_INFO,31,12,WMServer/9.0,51,WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_START_OFFSET,3,4,3100,47,WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_DURATION,3,6,170803,58,WMS_CONTENT_DESCRIPTION_COPIED_METADATA_FROM_PLAYLIST_FILE,3,1,1,42,WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_URL,31,17,0001-hi.wmv%0D%0A
a=pgmpu:data:application/vnd.ms.wms-hdr.asfv1;base64,Mcay...dY5mm2QCAQ==
m=audio 0 RTP/AVP 96
b=AS:35
b=RS:0
b=RR:0
a=rtpmap:96 x-asf-pf/1000
a=control:audio
a=stream:1
m=application 0 RTP/AVP 96
b=RS:0
b=RR:0
a=rtpmap:96 x-wms-rtx/1000
a=control:rtx
a=stream:65536
m=video 0 RTP/AVP 96

b=AS:471
b=RS:0
b=RR:0
a=rtpmap:96 x-asf-pf/1000
a=control:stream=2
a=stream:2

ending with \r\n\r\n

Looking at the line:
a=pgmpu:data:application/vnd.ms.wms-hdr.asfv1;base64, Mcay..…==

This is the actual ASF header data in a Base64 encoded format. This needs to be decoded into standard ascii before writing to file or buffer.

The SDP Body data also tells us what streams we have available in the media.
This file has 2 media streams:
ID = 1 audio with a label of “audio”
ID = 2 video with a label of “stream=2”
There is also a third stream labelled “rtx” which is a control stream but does not carry any media content to the client. The control stream uses stream ID = 65536.

Looking at the line:
a=pgmpu:data:application/x-wms-contentdesc, …..

This shows that what follows is a content description object.

Standard RTSP Header Error Codes – as line 1 in the response header

”100” ; Continue (all 100 range)
“200” ; OK
”201” ; Created
”250” ; Low on Storage Space
”300” ; Multiple Choices
”301” ; Moved Permanently
”302” ; Moved Temporarily
”303” ; See Other
”304” ; Not Modified
”305” ; Use Proxy
”350” ; Going Away
”351” ; Load Balancing
”400” ; Bad Request
”401” ; Unauthorized
”402” ; Payment Required
”403” ; Forbidden
”404” ; Not Found
”405” ; Method Not Allowed
”406” ; Not Acceptable
”407” ; Proxy Authentication Required
”408” ; Request Time-out
”410” ; Gone
”411” ; Length Required
”412” ; Precondition Failed
”413” ; Request Entity Too Large
”414” ; Request-URI Too Large
”415” ; Unsupported Media Type
”451” ; Parameter Not Understood
”452” ; reserved
”453” ; Not Enough Bandwidth
”454” ; Session Not Found
”455” ; Method Not Valid in This State
”456” ; Header Field Not Valid for Resource
”457” ; Invalid Range
”458” ; Parameter Is Read-Only
”459” ; Aggregate operation not allowed
”460” ; Only aggregate operation allowed
”461” ; Unsupported transport
”462” ; Destination unreachable
”500” ; Internal Server Error
”501” ; Not Implemented
”502” ; Bad Gateway
”503” ; Service Unavailable
”504” ; Gateway Time-out
”505” ; RTSP Version not supported
”551” ; Option not supported

RTSP Pre Header (Start of each Media Packet)

Bytes in the Pre header immediately precede the actual media packet (asf / wmv etc.)
Values are in Big Endian order. (not little endian as in mms and http protocols).

-------- these data bytes are specified in the open RTSP specification -------

BYTE	"$" or 0x24
BYTE	Channel id normally = 0x00, you may see "P" or 0x50 for packet pairs data, see below for description.
WORD	Length of packet (starting from the next field)

-------- the following bytes are not specified and may be MS specific ----------

WORD ??? fixed value of 0x80E0

WORD	Sequence. This value starts from a random value, then increments by 1 for every media packet sent. After the value 0xffff, the value wraps to 0. The range of this value is small compared to other protocols making it unusable for sequence referenced packet writing to file. Large files with many packets would fail because of wrap around. Use this sequence for error detection only – a missing sequence indicates a missing packet error.
DWORD	Send Time. This is stated in milliseconds and shows the time that this media packet should be presented. T = 0 at the start of pre-recorded files and with other values for live streams.
DWORD	??? fixed value. This is random in value. But once the session has started, it remains the same value for all media packets. Use this value as an ID number for the session. New sessions give new values.
BYTE	Flag. As below:

Bit 0 (lsb)	? = 0
Bit 1	? = 0
Bit 2	? = 0
Bit 3	? = 0
Bit 4	Duration field present in pre header if set (see below)
Bit 5	? = 0
Bit 6	? = 1 (always)
Bit 7 (msb)	packet contains a key frame if set

BYTE	??? always 0
WORD	Packet length. This shows the packet length starting from the Flag byte field.
DWORD	[optional] Duration time. This field is only present if the state of Flags is set to use it. Otherwise this field does not exist. Duration time is specified in milliseconds and shows the total duration of the media packet being sent.

<<< Media data follows, like 82 00 00 ….. >>>

Useful Tag Values used in RTSP Commands and Responses.

CSeq:	Sequence value of command and increments by 1. This is sent in all commands and responses.
Content-Length:	Shows we have body data if present. Indicates the length of the body data in bytes.
X-Playlist-Gen-Id:	This is used to check the playlist is valid. This tag is first sent to the client after the DECRIBE command is issued from the client. The client must respond with this exact tag value on sending the SETUP commands to the server.
X-Playlist-Seek-Id:	This must be set to the same value as the X-Playlist-Gen-Id value. This is then sent in the PLAY command.
Blocksize:	Shows the total length of the media packet in bytes.
Session:	Session ID is used to check for correct client and server connection. The Session ID is sent to the client in the response header from the previous SETUP command to server. We only see the Session value on the first stream selected (usually this is the audio stream). Session values are quite long, 20 digits in all. Following the session value, you may see a value: "timeout= xxxx". This is the required response or ACK time needed for the client to keep a connection alive with the server. The client must send an ACK within this time or the connection is force closed. An ACK is sent by sending the GET_PARAMETER command.
X-Accept-Authentication:	Shows what authentication methods are allowed. NTLM, Digest and Basic are standard.
X-Broadcast-Id:	Indicates whether this is a live or pre-recorded stream. Pre-recorded streams = 0, live = other value.
Range:	Range is the offset and end time positions to stream the media. For a zero start and full file stream, this value is set to: npt=0.000- where 0.000 is the offset and –0.000 (optional) is the ending time. Values are stated in seconds.
Speed:	Speed is used to adjust the rate of the stream to be sent to the client. Assuming your bandwidth can handle the increase in network data sent, this value can be used to speed up a download by making it greater than 1. Normal setting: Speed: 1.0 i.e. x1 rate Media player uses : Speed: 1.294 this maybe dependent on your connection speed.
Server:	Shows the server type and version of software.
EOF:	End of File indicator, also end of stream.
Date:	Shows the time and date e.g. Tue, 18 Nov 2003 15:57:07 GMT
Bandwidth:	Shows the maximum required bandwidth in Bits per second for the stream.

Transport:	Indicates what transport protocol like TCP or UDP we are to use.
Etag:	Entity tag, shows a value assigned to the session. Like : "23180160"
Supported:	COM modules supported, some are optional. com.microsoft.wm.srvppair - packet pairs at server com.microsoft.wm.sswitch - stream ID selection com.microsoft.wm.eosmsg - end of stream message com.microsoft.wm.fastcache - fast cache for buffering com.microsoft.wm.packetpairssrc. - packet pairs
Content-Type:	This shows the purpose of the command or response. Here are some commonly used types: application/x-wms-Logconnectstats This is used in the SET_PARAMETER command and indicates logging data to server. application/sdp This indicates session description protocol data follows in the body section and is sent in response to the DESCRIBE command to server. application/x-wms-contentdesc Indicates that the data what follows is a content description object. It sets the layout of the dialog. application/vnd.ms.wms-hdr.asfv1 Indicates a media header follows (ASF header). This can encoded using BASIC or DIGEST encoding. application/x-rtsp-packetpair Packet Pair data is random non-compressible data and is sent to the client and timed for response times. This is used to determine the available bandwidth of the connection. Packet pair data is optional, you do not always need to request this data. This content type is sent to the server in the GET_PARAMATER command.