HTTP streaming protocol version 1.0

By the SDP team : http://sdp.ppona.com

History:

Update 07.Dec.2003
Section: ‘Content-Types’, MIME downloadable and skin types added.
Section: ‘Server Sent HTTP Body Data’, type header structure updated.
Sub Section: ‘Pragma: stream-switch-entry’, data updated.
Section: ‘Some Examples of HTTP Headers’, updated with comments.

Update 04.July.2003
Section: ‘Client sent HTTP headers’ Authorization field been added.
Section: ‘Client sent HTTP headers’ the Notes data has been updated.
Section: ‘Server sent HTTP body data’ Type Header Structure commands updated.

Update 12.Jan.2003
Section: ’server sent HTTP headers’ HTTP progressive types updated with new content type ‘wm’.

<<< Start of History: Date 18.Dec.2002

How is streaming HTTP different from standard host HTTP?
Http protocol is used by Microsoft® streaming servers to stream live or pre-recorded media over the Internet or via a local network to a client computer. HTTP protocol is wasteful in terms of packet size overhead, yet reliable by way of its transport error correction ability. For this reason, it is a good protocol method for reliable streaming, downloading or copying of media data over the Internet. Being HTTP, it can also operate safely through a firewall (like port 80) and via standard HTTP proxy servers. Streaming HTTP protocol version 1.0 or more recently, version 1.1 is very similar to well-known HTTP protocol used by standard host web servers. The kind normally associated with viewing web site locations and email delivery etc. The only difference with HTTP 'streaming' protocol is the inclusion of custom 'Pragma' values and the use of special 'type header' and 'pre-header' data structures attached to ASF packets. Typically, the media content is of the Microsoft® ASF standard.
Other than that, streaming HTTP protocol closely resembles that of standard HTTP host protocol.

About basic HTTP
HTTP protocol is really just an elaboration on simple TCP socket connections. HTTP uses TCP as its actual transport protocol and uses a TCP socket connection for that purpose. Once a TCP socket connection is established between the client player and the streaming server computer, HTTP protocol then acts on top of that transport to deliver media content and to send various commands to the player. All packet error correction and handshaking is then achieved using this TCP socket.

A typical HTTP streaming protocol session exchange

To Server =>	NETWORK	<= To Client
Client connects to server TCP socket		
Client requests ASF media header		Server replies with ASF header only Then server disconnects from client
Client parses ASF header Client re-connects to server TCP socket And re-requests with selected streams and 'live' or 'pre-recorded' media type		Server replies with ASF header (again) followed by media packets using the selected Streams. < ### Media stream packets now follow ### > Server ends with 'End of Media' command Then server disconnects from client
Client player stops.		

Notes:
During a streaming session, the server can indicate either a 'End of media' or 'Change Media' by sending commands to the client as explained later.

All HTTP packets, either client-sent or server-sent have HTTP headers at the beginnings. HTTP headers are explained below.

Server sent HTTP headers

We could talk a lot about HTTP headers, but instead we will only give a brief outline of what we need to know to establish a HTTP streaming link. There are many websites and documents already available that give details of this subject. For more detailed information, please refer to any HTTP protocol document. More information is also available regarding headers and body data at:
http://www.jmarshall.com/easy/http/

Notes:
All header data is of a string type.
String lines are terminated with a CRFL character. End of line.
The very end of the header block is indicated by a CRLFCRLF - double end of line.
Pragma command values can be strung together on one line and separated by commas. In this way only one Pragma keyword is used at the start of line, followed by multiple Pragma values.

The Status Line
Once a connection is established to the server by the client, an initial status response (line 1 of the returned HTTP header) is detailed below:

Typical status lines are:
HTTP/1.0 200 OK
or;
HTTP/1.0 404 Not Found

Notes:
The HTTP version is in the same format as in the request line, "HTTP/x.x" as used in client HTTP header generation.
The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary.

The status code is a three-digit integer, and the first digit identifies the general category of response:
1xx indicates an informational message only
2xx indicates success of some kind
3xx redirects the client to another URL
4xx indicates an error on the client's part
5xx indicates an error on the server's part

Of course, there are many more status values, more information can be found on the web regarding HTTP status message values.

Content-Type: <value>

Where <value> is the MIME type of data to be transferred.
Typically, values for ASF HTTP streaming are:

Streaming HTTP 1.0 protocol (using MMS frames):

'application/x-mms-framed'	ASF media packets using MMS style pre-headers
'application/vnd.ms.wms-hdr.asfv1'	ASF header data (version 1.0 asf)
'application/octet-stream'	ASF format.

Http downloadable and skin types:

'application/x-ms-wmd'	.wmd download files
'application/x-ms-wmz'	.wmz skin files

Simple HTTP progressive downloading (not streaming protocol):
'audio/x-ms-wax'
'audio/x-ms-wma'
'video/x-ms-asf'
'video/x-ms-afs'
'video/x-ms-wvx'
'video/x-ms-wmv'
'video/x-ms-wma'
'video/x-ms-wm'
'text/plain'

Server: <value>
Where <value> shows the server name and version string.
An example is: cougar 4.1.0.3927

Content-Length: <value>
Shows the total length of the HTTP packet. (we see this in header requests only, not media requests).

Date: <value>
Shows the current connection date and time as a readable string.

Pragma: client-id= <value>
Where <value> is the client id number e.g. 123456 issued by the server and is a connection value which increments by 1 for every client connected.

Pragma: features="<value>" - (quotes are required)
Where <value> can be 'broadcast' which indicates the stream is LIVE (not pre-recorded)
And other values like 'seekable' and 'stridable' show the stream is pre-recorded.

Pragma: x-wms-content-size=<value>
Where <value> is the total length of the media file in bytes.

Via: <value>
Where <value> shows the return protocol (HTTP/1.0) and return URL route back to the client.

Client sent HTTP headers

Notes:
All header data is of a string type.
String lines are terminated with a CRFL character. End of line.
The very end of the header block is indicated by a CRLFCRLF - double end of line.
Pragma command values can be strung together on one line and separated by commas. In this way only one Pragma keyword is used at the start of line, followed by multiple Pragma values.

GET <value> HTTP/1.0
The <value> field in this string depends on whether you connect to a proxy server or not.

For a normal direct server connection:
Value = "/file path and file name.ext"

And for proxy server connections:
Value = "http://the.serverhost.com/file path and file name.ext"
Where: the.serverhost.com is the actual streaming server address and not the proxy address.

Where <file path and name.ext> locates the media at the server and indicates its file name.
HTTP/1.0 shows the desired protocol version needed to stream.

Accept: */*
This can specify exactly what purpose the data is used for and what to accept as valid data. This is set to */* as a default setting.

User-Agent: <value>
Where <value> shows the client player version and type. E.g. NSPlayer/4/1/0/3925

Host: <value>
Where <value> shows the host computer (or server) location the client player is connecting to, e.g. netshow.streamsoft.com
Even if we are connecting to a proxy server address, <value> always shows the actual streaming server address. Not the proxy.

Pragma: no-cache
Indicates no cache is to be used – this is normally set as shown.

Pragma: rate=<value>
Where <value> is always set to 1.000000

Pragma: stream-time=<value>
Where <value> is the offset time to start streaming from. This value is stated in milliseconds. The start of media has the value of 0.

Pragma: stream-offset=<value1>:<value2> **-** includes the colon separator
Where both value 1 and 2 fields show the value of 0.
These values are explained as being the offset point into the media, but no use has been found for this command and it is not used, but it must be present.

Pragma: request-context=<value>
Where <value> can be:
1 for initial header request
2 for pre-recorded or live media packets request
4 for seek into media request (using stream-time Pragma)

Pragma: max-duration=<value>
Where <value> sets the maximum time allowed for streaming. After this time, the server will disconnect from the client, regardless of how far the media has played. A default value of 0 indicates no limit.

Pragma: xClientGUID=<value>
Where <value> is a unique GUID string in the standard form:
{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}
This value is generated by the client upon connection to the server and changes for each new connection made.

Authorization: BASIC (or DIGEST) <value>
Where <value> is the encoded data string using the required method of encryption.

Other header lines exist, these are only the main ones used for ASF streaming.

More HTTP header lines used only in media packet (not ASF header) requests:

Pragma: xPlayStrm=<value>
Where <value> is always set to 1.

Pragma: stream-switch-count=<value>
Where <value> shows the total streams that this media request is making. A count value of 1 would mean only one stream and is used for audio only streams. More than 1 stream usually indicates video with audio or a command stream with audio. This value is derived from the ASF header previously parsed and counts the total requested streams.

Pragma: stream-switch-entry=<value>
Where <value> is a string to either select or deselect the required streams of the media to send:

Starts with	ffff:
Then stream ID number	1:
Then state of stream	0

Using ':' colons as separators.

Where ffff is preset to this value, stream ID number shows the ID of the stream we are selecting and the state of the stream is either:
0 for 'full frame rate' (ON);
1 for 'key frames only';
2 for 'deselected' (OFF).

Stream strings are joined together, as an example for:
Stream ID 1 (audio) is ON, Stream ID 2 (video low rate) is OFF and Stream ID 4 is ON (video hi rate):

Would equal: ffff:1:0 ffff:2:2 ffff:4:0

More details regarding stream switch selectors can be found in the MMS protocol document.

Notes:
For LIVE streams, the following Pragma values can be omitted or if they are sent, they must = 0:
Pragma: stream-time
Pragma: stream-offset
Are not used in header data generation to the server.
These values are only used for pre-recorded media.

Server Sent HTTP Body Data

HTTP body data describes all data that immediately follows the HTTP header data as described above. Unlike HTTP headers, HTTP body data uses binary BYTE format rather than strings.

Body data contains all the actual data and payloads of the ASF header and media packets which make up the stream.

HTTP Body data has the following structure starting from the beginning of a HTTP data packet:

Type Header Structure

Command 2 bytes	Length 2 bytes

Where 'Command' can have one of the following meanings:

Hex	ASCII	Meaning
0x2448	"$H"	Header packet follows
0x2444	"$D"	Data (media) packet follows
0xA444	"?D"	Data (sometimes used instead of $D)
0x2445	"$E"	End of stream
0x2443	"$C"	Changing media – reload new media, new ASF header follows
0x244D	"$M"	Meta data follows (used for custom scripts)
0x2450	"$P"	Packet pair data follows

And 'Length' means:
For both Header Packets and Data/Media Packets, length indicates the total length in bytes of the packet. Starting from the very first byte after this Type Header structure, and up to the very last byte in the packet.
For End Of Stream and Changing Media commands, 'length' shows a value of 0x08, which is the length of this Type Header structure and not the length of any following data! Note also that no 'Type Object' follows for these commands. See below.

Type Object or 'MMS Pre-header' Structure

Sequence number	ID	Flags	Length

This is a similar structure as used by MMS protocol.

Where:

Sequence (4 bytes) — When pre-recorded media stream, this packet sequence value starts from 0 for the first ASF packet sent, then incrementing by 1 for every sequence. When live media stream, the value can start from the sequence number we happen to have came in during the broadcast. Then sequence increments.

ID (1 byte) — ID –
When stream is pre-recorded media, this value is fixed as 0x01.
When stream is live, this value is fixed as 0x00.

Flags (1 byte) — When stream is live media or a header packet, values are indicated as below:
0x00 = middle of packet series
0x04 = first packet of packet series
0x08 = last packet of packet series
0x0C = only one packet in series

When stream is pre-recorded, flags start from value 0x00 and then increment by 1 for every packet sequence sent to the client. After the value 0xff is reached, the value wraps around to 0x00 again.

Length (2 bytes) — the total length in bytes of this packet including this Type Object structure.

Finally, ASF data follows to the end of packet.
Immediately after the Type Object structure, either an ASF header or ASF Media data packet follows to the end of the packet 'length'. The type of data sent is indicated by 'Command'. Typically, ASF media packets start with the values 0x82, 0x00, 0x00. This shows an error correction start sequence of a common ASF media packet.

Some examples of generated HTTP headers (hashes are just comments)

Start client request to server (none proxy connection):

```
GET /ms/contoso_100_files/0MM0.wmv HTTP/1.0
Accept: */*
User-Agent: NSPlayer/4.1.0.3925
Host: netshow.micro.com
Pragma: no-cache,rate=1.000000,stream-time=0,stream-offset=0:0,request-context=1,max-duration=0
Pragma: xClientGUID={2200AD50-2C39-46c0-AE0A-2CA76D8C766D}
```

Server reply with header only:

```
HTTP/1.0 200 OK
Content-Type: application/vnd.ms.wms-hdr.asfv1      #### this is the ASF Header Content-Type ####
Server: Cougar/9.00.00.3101
Content-Length: 1765
Date: Thu, 28 Nov 2002 23:52:06 GMT
Pragma: no-cache, client-id=2555138553, features="seekable,stridable"
Cache-Control: no-cache, x-wms-content-size=1487295, max-age=86399, user-public, must-revalidate,
proxy-public, proxy-revalidate
Last-Modified: Wed, 30 Jan 2002 00:52:31 GMT
Etag: "1487295"
Supported: com.microsoft.wm.srvppair, com.microsoft.wm.sswitch, com.microsoft.wm.predstrm,
com.microsoft.wm.fastcache
Age: 0
Via: HTTP/1.0 cache-ink1-cro.cableinet.com (Traffic-Server/5.2.1 [cMsSf ])
```

Second client request this time with stream selection:

```
GET /ms/contoso_100_files/0MM0.wmv HTTP/1.0
Accept: */*
User-Agent: NSPlayer/4.1.0.3925
Host: netshow.micro.com
Pragma: no-cache,rate=1.000000,stream-time=0,stream-offset=4294967295:4294967295,request-
context=2,max-duration=2147609515
Pragma: xPlayStrm=1
Pragma: xClientGUID={2200AD50-2C39-46c0-AE0A-2CA76D8C766D}
Pragma: stream-switch-count=2
Pragma: stream-switch-entry=ffff:1:0 ffff:2:0
```

Final server reply with header followed by media packets:

```
HTTP/1.0 200 OK
Content-Type: application/x-mms-framed          #### this is the HTTP streaming Content-Type ####
Server: Cougar/9.00.00.3101
Date: Thu, 28 Nov 2002 23:59:12 GMT
Pragma: no-cache, client-id=2807872270, features="seekable,stridable", AccelBW=0,
AccelDuration=0, Speed=1.000
Cache-Control: no-cache
Supported: com.microsoft.wm.srvppair, com.microsoft.wm.sswitch, com.microsoft.wm.predstrm,
com.microsoft.wm.fastcache
Age: 0
Via: HTTP/1.0 cache-ink1-cro.cableinet.com (Traffic-Server/5.2.1 [cMsSf ])
```