Appendix A. Examples

This section contains several different examples trying to illustrate possible ways of using RTSP. The examples can also help with the understanding of how functions of RTSP work. However, remember that these are examples and the normative and syntax description in the other sections takes precedence. Please also note that many of the example contain syntax illegal line breaks to accommodate the formatting restriction that the RFC series impose.

A.1. Media on Demand (Unicast)

This is an example of media on demand streaming of a media stored in a container file. For purposes of this example, a container file is a storage entity in which multiple continuous media types pertaining to the same end-user presentation are present. In effect, the container file represents an RTSP presentation, with each of its components being RTSP controlled media streams. Container files are a widely used means to store such presentations. While the components are transported as independent streams, it is desirable to maintain a common context for those streams at the server end.

> This enables the server to keep a single storage handle open easily. It also allows treating all the streams equally in case of any priorization of streams by the server.

It is also possible that the presentation author may wish to prevent selective retrieval of the streams by the client in order to preserve the artistic effect of the combined media presentation. Similarly, in such a tightly bound presentation, it is desirable to be able to control all the streams via a single control message using an aggregate URI.

The following is an example of using a single RTSP session to control multiple streams. It also illustrates the use of aggregate URIs. In a container file it is also desirable to not write any URI parts which is not kept, when the container is distributed, like the host and most of the path element. Therefore this example also uses the "*" and relative URI in the delivered SDP.

Also this presentation description (SDP) is not cachable, as the Expires header is set to an equal value with date indicating immediate expiration of its valididty.

Client C requests a presentation from media server M. The movie is stored in a container file. The client has obtained an RTSP URI to the container file.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1520
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
   C->M: DESCRIBE rtsp://example.com/twister.3gp RTSP/2.0
         CSeq: 1
         User-Agent: PhonyClient/1.2

   M->C: RTSP/2.0 200 OK
         CSeq: 1
         Server: PhonyServer/1.0
         Date: Thu, 23 Jan 1997 15:35:06 GMT
         Content-Type: application/sdp
         Content-Length: 271
         Content-Base: rtsp://example.com/twister.3gp/
         Expires: 24 Jan 1997 15:35:06 GMT

         v=0
         o=- 2890844256 2890842807 IN IP4 198.51.100.5
         s=RTSP Session
         i=An Example of RTSP Session Usage
         e=adm@example.com
         c=IN IP4 0.0.0.0
         a=control: *
         a=range: npt=0-0:10:34.10
         t=0 0
         m=audio 0 RTP/AVP 0
         a=control: trackID=1
         m=video 0 RTP/AVP 26
         a=control: trackID=4
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1521
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
   C->M: SETUP rtsp://example.com/twister.3gp/trackID=1 RTSP/2.0
         CSeq: 2
         User-Agent: PhonyClient/1.2
         Require: play.basic
         Transport: RTP/AVP;unicast;dest_addr=":8000"/":8001"
         Accept-Ranges: NPT, SMPTE, UTC

   M->C: RTSP/2.0 200 OK
         CSeq: 2
         Server: PhonyServer/1.0
         Transport: RTP/AVP;unicast; ssrc=93CB001E;
                    dest_addr="192.0.2.53:8000"/"192.0.2.53:8001";
                    src_addr="198.51.100.5:9000"/"198.51.100.5:9001"
         Session: 12345678
         Expires: 24 Jan 1997 15:35:12 GMT
         Date: 23 Jan 1997 15:35:12 GMT
         Accept-Ranges: NPT
         Media-Properties: Random-Access=0.02, Immutable, Unlimited

   C->M: SETUP rtsp://example.com/twister.3gp/trackID=4 RTSP/2.0
         CSeq: 3
         User-Agent: PhonyClient/1.2
         Require: play.basic
         Transport: RTP/AVP;unicast;dest_addr=":8002"/":8003"
         Session: 12345678
         Accept-Ranges: NPT, SMPTE, UTC


   M->C: RTSP/2.0 200 OK
         CSeq: 3
         Server: PhonyServer/1.0
         Transport: RTP/AVP;unicast; ssrc=A813FC13;
                    dest_addr="192.0.2.53:8002"/"192.0.2.53:8003";
                    src_addr="198.51.100.5:9002"/"198.51.100.5:9003";

         Session: 12345678
         Expires: 24 Jan 1997 15:35:13 GMT
         Date: 23 Jan 1997 15:35:13 GMT
         Accept-Range: NPT
         Media-Properties: Random-Access=0.8, Immutable, Unlimited

   C->M: PLAY rtsp://example.com/twister.3gp/ RTSP/2.0
         CSeq: 4
         User-Agent: PhonyClient/1.2
         Range: npt=30-
         Seek-Style: RAP
         Session: 12345678
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1522
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
   M->C: RTSP/2.0 200 OK
         CSeq: 4
         Server: PhonyServer/1.0
         Date: 23 Jan 1997 15:35:14 GMT
         Session: 12345678
         Range: npt=30-623.10
         Seek-Style: RAP
         RTP-Info: url="rtsp://example.com/twister.3gp/trackID=4"
            ssrc=0D12F123:seq=12345;rtptime=3450012,
           url="rtsp://example.com/twister.3gp/trackID=1"
            ssrc=4F312DD8:seq=54321;rtptime=2876889

   C->M: PAUSE rtsp://example.com/twister.3gp/ RTSP/2.0
         CSeq: 5
         User-Agent: PhonyClient/1.2
         Session: 12345678

   M->C: RTSP/2.0 200 OK
         CSeq: 5
         Server: PhonyServer/1.0
         Date: 23 Jan 1997 15:36:01 GMT
         Session: 12345678
         Range: npt=34.57-623.10

   C->M: PLAY rtsp://example.com/twister.3gp/ RTSP/2.0
         CSeq: 6
         User-Agent: PhonyClient/1.2
         Range: npt=34.57-623.10
         Seek-Style: Next
         Session: 12345678

   M->C: RTSP/2.0 200 OK
         CSeq: 6
         Server: PhonyServer/1.0
         Date: 23 Jan 1997 15:36:01 GMT
         Session: 12345678
         Range: npt=34.57-623.10
         Seek-Style: Next
         RTP-Info: url="rtsp://example.com/twister.3gp/trackID=4"
            ssrc=0D12F123:seq=12555;rtptime=6330012,
           url="rtsp://example.com/twister.3gp/trackID=1"
            ssrc=4F312DD8:seq=55021;rtptime=3132889


   C->M: TEARDOWN rtsp://example.com/twister.3gp/ RTSP/2.0
         CSeq: 7
         User-Agent: PhonyClient/1.2
         Session: 12345678
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1523
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
   M->C: RTSP/2.0 200 OK
         CSeq: 7
         Server: PhonyServer/1.0
         Date: 23 Jan 1997 15:49:34 GMT
```

A.2. Media on Demand using Pipelining

This example is basically the example above (Appendix A.1), but now utilizing pipelining to speed up the setup. It requires only two round trip times until the media starts flowing. First of all, the session description is retrieved to determine what media resources need to be setup. In the second step, one sends the necessary SETUP requests and the PLAY request to initiate media delivery.

Client C requests a presentation from media server M. The movie is stored in a container file. The client has obtained an RTSP URI to the container file.

```
   C->M: DESCRIBE rtsp://example.com/twister.3gp RTSP/2.0
         CSeq: 1
         User-Agent: PhonyClient/1.2

   M->C: RTSP/2.0 200 OK
         CSeq: 1
         Server: PhonyServer/1.0
         Date: Thu, 23 Jan 1997 15:35:06 GMT
         Content-Type: application/sdp
         Content-Length: 271
         Content-Base: rtsp://example.com/twister.3gp/
         Expires: 24 Jan 1997 15:35:06 GMT

         v=0
         o=- 2890844256 2890842807 IN IP4 192.0.2.5
         s=RTSP Session
         i=An Example of RTSP Session Usage
         e=adm@example.com
         c=IN IP4 0.0.0.0
         a=control: *
         a=range: npt=0-0:10:34.10
         t=0 0
         m=audio 0 RTP/AVP 0
         a=control: trackID=1
         m=video 0 RTP/AVP 26
         a=control: trackID=4

   C->M: SETUP rtsp://example.com/twister.3gp/trackID=1 RTSP/2.0
         CSeq: 2
         User-Agent: PhonyClient/1.2
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1524
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
        Require: play.basic
        Transport: RTP/AVP;unicast;dest_addr=":8000"/":8001"
        Accept-Ranges: NPT, SMPTE, UTC
        Pipelined-Requests: 7654

   C->M: SETUP rtsp://example.com/twister.3gp/trackID=4 RTSP/2.0
        CSeq: 3
        User-Agent: PhonyClient/1.2
        Require: play.basic
        Transport: RTP/AVP;unicast;dest_addr=":8002"/":8003"
        Accept-Ranges: NPT, SMPTE, UTC
        Pipelined-Requests: 7654

   C->M: PLAY rtsp://example.com/twister.3gp/ RTSP/2.0
        CSeq: 4
        User-Agent: PhonyClient/1.2
        Range: npt=0-
        Seek-Style: RAP
        Session: 12345678
        Pipelined-Requests: 7654

   M->C: RTSP/2.0 200 OK
        CSeq: 2
        Server: PhonyServer/1.0
        Transport: RTP/AVP;unicast;
                   dest_addr="192.0.2.53:8000"/"192.0.2.53:8001";
                   src_addr="198.51.100.5:9000"/"198.51.100.5:9001";
                   ssrc=93CB001E
        Session: 12345678
        Expires: 24 Jan 1997 15:35:12 GMT
        Date: 23 Jan 1997 15:35:12 GMT
        Accept-Ranges: NPT
        Pipelined-Requests: 7654
        Media-Properties: Random-Access=0.2, Immutable, Unlimited

   M->C: RTSP/2.0 200 OK
        CSeq: 3
        Server: PhonyServer/1.0
        Transport: RTP/AVP;unicast;
                   dest_addr="192.0.2.53:8002"/"192.0.2.53:8003;
                   src_addr="198.51.100.5:9002"/"198.51.100.5:9003";
                   ssrc=A813FC13
        Session: 12345678
        Expires: 24 Jan 1997 15:35:13 GMT
        Date: 23 Jan 1997 15:35:13 GMT
        Accept-Range: NPT
        Pipelined-Requests: 7654
        Media-Properties: Random-Access=0.8, Immutable, Unlimited
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1525
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
M->C: RTSP/2.0 200 OK
      CSeq: 4
      Server: PhonyServer/1.0
      Date: 23 Jan 1997 15:35:14 GMT
      Session: 12345678
      Range: npt=0-623.10
      Seek-Style: RAP
      RTP-Info: url="rtsp://example.com/twister.3gp/trackID=4"
         ssrc=0D12F123:seq=12345;rtptime=3450012,
        url="rtsp://example.com/twister.3gp/trackID=1"
         ssrc=4F312DD8:seq=54321;rtptime=2876889
      Pipelined-Requests: 7654
```

A.3. Media on Demand (Unicast)

An alternative example of media on demand with a bit more tweaks is the following. Client C requests a movie distributed from two different media servers A (audio.example.com) and V (video.example.com). The media description is stored on a web server W. The media description contains descriptions of the presentation and all its streams, including the codecs that are available, dynamic RTP payload types, the protocol stack, and content information such as language or copyright restrictions. It may also give an indication about the timeline of the movie.

In this example, the client is only interested in the last part of the movie.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1526
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
   C->W: GET /twister.sdp HTTP/1.1
         Host: www.example.com
         Accept: application/sdp

   W->C: HTTP/1.0 200 OK
         Date: Thu, 23 Jan 1997 15:35:06 GMT
         Content-Type: application/sdp
         Content-Length: 278
         Expires: 23 Jan 1998 15:35:06 GMT

         v=0
         o=- 2890844526 2890842807 IN IP4 198.51.100.5
         s=RTSP Session
         e=adm@example.com
         c=IN IP4 0.0.0.0
         a=range:npt=0-1:49:34
         t=0 0
         m=audio 0 RTP/AVP 0
         a=control:rtsp://audio.example.com/twister/audio.en
         m=video 0 RTP/AVP 31
         a=control:rtsp://video.example.com/twister/video

   C->A: SETUP rtsp://audio.example.com/twister/audio.en RTSP/2.0
         CSeq: 1
         User-Agent: PhonyClient/1.2
         Transport: RTP/AVP/UDP;unicast;dest_addr=":3056"/":3057",
                    RTP/AVP/TCP;unicast;interleaved=0-1
         Accept-Ranges: NPT, SMPTE, UTC

   A->C: RTSP/2.0 200 OK
         CSeq: 1
         Session: 12345678
         Transport: RTP/AVP/UDP;unicast;
                    dest_addr="192.0.2.53:3056"/"192.0.2.53:3057";
                    src_addr="198.51.100.5:5000"/"198.51.100.5:5001"
         Date: 23 Jan 1997 15:35:12 GMT
         Server: PhonyServer/1.0
         Expires: 24 Jan 1997 15:35:12 GMT
         Cache-Control: public
         Accept-Ranges: NPT, SMPTE
         Media-Properties: Random-Access=0.02, Immutable, Unlimited
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1527
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
C->V: SETUP rtsp://video.example.com/twister/video RTSP/2.0
      CSeq: 1
      User-Agent: PhonyClient/1.2
      Transport: RTP/AVP/UDP;unicast;
                 dest_addr="192.0.2.53:3058"/"192.0.2.53:3059",
                 RTP/AVP/TCP;unicast;interleaved=0-1
      Accept-Ranges: NPT, SMPTE, UTC

V->C: RTSP/2.0 200 OK
      CSeq: 1
      Session: 23456789
      Transport: RTP/AVP/UDP;unicast;
         dest_addr="192.0.2.53:3058"/"192.0.2.53:3059";
         src_addr="198.51.100.5:5002"/"198.51.100.5:5003"
      Date: 23 Jan 1997 15:35:12 GMT
      Server: PhonyServer/1.0
      Cache-Control: public
      Expires: 24 Jan 1997 15:35:12 GMT
      Accept-Ranges: NPT, SMPTE
      Media-Properties: Random-Access=1.2, Immutable, Unlimited

C->V: PLAY rtsp://video.example.com/twister/video RTSP/2.0
      CSeq: 2
      User-Agent: PhonyClient/1.2
      Session: 23456789
      Range: smpte=0:10:00-

V->C: RTSP/2.0 200 OK
      CSeq: 2
      Session: 23456789
      Range: smpte=0:10:00-1:49:23
      Seek-Style: First-Prior
      RTP-Info: url="rtsp://video.example.com/twister/video"
                ssrc=A17E189D:seq=12312232;rtptime=78712811
      Server: PhonyServer/2.0
      Date: 23 Jan 1997 15:35:13 GMT
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1528
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
   C->A: PLAY rtsp://audio.example.com/twister/audio.en RTSP/2.0
         CSeq: 2
         User-Agent: PhonyClient/1.2
         Session: 12345678
         Range: smpte=0:10:00-

   A->C: RTSP/2.0 200 OK
         CSeq: 2
         Session: 12345678
         Range: smpte=0:10:00-1:49:23
         Seek-Style: First-Prior
         RTP-Info: url="rtsp://audio.example.com/twister/audio.en"
                   ssrc=3D124F01:seq=876655;rtptime=1032181
         Server: PhonyServer/1.0
         Date: 23 Jan 1997 15:35:13 GMT



   C->A: TEARDOWN rtsp://audio.example.com/twister/audio.en RTSP/2.0
         CSeq: 3
         User-Agent: PhonyClient/1.2
         Session: 12345678

   A->C: RTSP/2.0 200 OK
         CSeq: 3
         Server: PhonyServer/1.0
         Date: 23 Jan 1997 15:36:52 GMT

   C->V: TEARDOWN rtsp://video.example.com/twister/video RTSP/2.0
         CSeq: 3
         User-Agent: PhonyClient/1.2
         Session: 23456789

   V->C: RTSP/2.0 200 OK
         CSeq: 3
         Server: PhonyServer/2.0
         Date: 23 Jan 1997 15:36:52 GMT
```

Even though the audio and video track are on two different servers that may start at slightly different times and may drift with respect to each other over time, the client can perform initial synchronization of the two media using RTP-Info and Range received in the PLAY responses. If the two servers are time synchronized the RTCP packets can also be used to maintain synchronization.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1529
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

A.4. Single Stream Container Files

Some RTSP servers may treat all files as though they are "container files", yet other servers may not support such a concept. Because of this, clients needs to use the rules set forth in the session description for Request-URIs, rather than assuming that a consistent URI may always be used throughout. Below are an example of how a multi-stream server might expect a single-stream file to be served:

```
C->S: DESCRIBE rtsp://foo.example.com/test.wav RTSP/2.0
      Accept: application/x-rtsp-mh, application/sdp
      CSeq: 1
      User-Agent: PhonyClient/1.2

S->C: RTSP/2.0 200 OK
      CSeq: 1
      Content-base: rtsp://foo.example.com/test.wav/
      Content-type: application/sdp
      Content-length: 163
      Server: PhonyServer/1.0
      Date: Thu, 23 Jan 1997 15:35:06 GMT
      Expires: 23 Jan 1997 17:00:00 GMT

      v=0
      o=- 872653257 872653257 IN IP4 192.0.2.5
      s=mu-law wave file
      i=audio test
      c=IN IP4 0.0.0.0
      t=0 0
      a=control: *
      m=audio 0 RTP/AVP 0
      a=control:streamid=0
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1530
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
C->S: SETUP rtsp://foo.example.com/test.wav/streamid=0 RTSP/2.0
      Transport: RTP/AVP/UDP;unicast;
         dest_addr=":6970"/":6971";mode="PLAY"
      CSeq: 2
      User-Agent: PhonyClient/1.2
      Accept-Ranges: NPT, SMPTE, UTC

S->C: RTSP/2.0 200 OK
      Transport: RTP/AVP/UDP;unicast;
          dest_addr="192.0.2.53:6970"/"192.0.2.53:6971";
          src_addr="198.51.100.5:6970"/"198.51.100.5:6971";
          mode="PLAY";ssrc=EAB98712
      CSeq: 2
      Session: 2034820394
      Expires: 23 Jan 1997 16:00:00 GMT
      Server: PhonyServer/1.0
      Date: 23 Jan 1997 15:35:07 GMT
      Accept-Ranges: NPT
      Media-Properties: Random-Acces=0.5, Immutable, Unlimited


C->S: PLAY rtsp://foo.example.com/test.wav/ RTSP/2.0
      CSeq: 3
      User-Agent: PhonyClient/1.2
      Session: 2034820394

S->C: RTSP/2.0 200 OK
      CSeq: 3
      Server: PhonyServer/1.0
      Date: 23 Jan 1997 15:35:08 GMT
      Session: 2034820394
      Range: npt=0-600
      Seek-Style: RAP
      RTP-Info: url="rtsp://foo.example.com/test.wav/streamid=0"
         ssrc=0D12F123:seq=981888;rtptime=3781123
```

Note the different URI in the SETUP command, and then the switch back to the aggregate URI in the PLAY command. This makes complete sense when there are multiple streams with aggregate control, but is less than intuitive in the special case where the number of streams is one. However, the server has declared that the aggregated control URI in the SDP and therefore this is legal.

In this case, it is also required that servers accept implementations that use the non-aggregated interpretation and use the individual media URI, like this:

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1531
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
   C->S: PLAY rtsp://example.com/test.wav/streamid=0 RTSP/2.0
         CSeq: 3
         User-Agent: PhonyClient/1.2
         Session: 2034820394
```

A.5. Live Media Presentation Using Multicast

The media server M chooses the multicast address and port. Here, it is assumed that the web server only contains a pointer to the full description, while the media server M maintains the full description.

```
   C->W: GET /sessions.html HTTP/1.1
         Host: www.example.com

   W->C: HTTP/1.1 200 OK
         Content-Type: text/html

         <html>
           ...
           <a href "rtsp://live.example.com/concert/audio">
              Streamed Live Music performance </a>
           ...
         </html>


   C->M: DESCRIBE rtsp://live.example.com/concert/audio RTSP/2.0
         CSeq: 1
         Supported: play.basic, play.scale
         User-Agent: PhonyClient/1.2

   M->C: RTSP/2.0 200 OK
         CSeq: 1
         Content-Type: application/sdp
         Content-Length: 183
         Server: PhonyServer/1.0
         Date: Thu, 23 Jan 1997 15:35:06 GMT
         Supported: play.basic

         v=0
         o=- 2890844526 2890842807 IN IP4 192.0.2.5
         s=RTSP Session
         t=0 0
         m=audio 3456 RTP/AVP 0
         c=IN IP4 233.252.0.54/16
         a=control: rtsp://live.example.com/concert/audio
         a=range:npt=0-
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1532
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
C->M: SETUP rtsp://live.example.com/concert/audio RTSP/2.0
      CSeq: 2
      Transport: RTP/AVP;multicast;
           dest_addr="233.252.0.54:3456"/"233.252.0.54:3457";ttl=16
      Accept-Ranges: NPT, SMPTE, UTC
      User-Agent: PhonyClient/1.2

M->C: RTSP/2.0 200 OK
      CSeq: 2
      Server: PhonyServer/1.0
      Date: Thu, 23 Jan 1997 15:35:06 GMT
      Transport: RTP/AVP;multicast;
           dest_addr="233.252.0.54:3456"/"233.252.0.54:3457";ttl=16
           ;ssrc=4D12AB92/0DF876A3
      Session: 0456804596
      Accept-Ranges: NPT, UTC
      Media-Properties: No-Seeking, Time-Progressing, Time-Duration=0


C->M: PLAY rtsp://live.example.com/concert/audio RTSP/2.0
      CSeq: 3
      Session: 0456804596
      User-Agent: PhonyClient/1.2

M->C: RTSP/2.0 200 OK
      CSeq: 3
      Server: PhonyServer/1.0
      Date: 23 Jan 1997 15:35:07 GMT
      Session: 0456804596
      Seek-Style: Next
      Range:npt=1256-
      RTP-Info: url="rtsp://live.example.com/concert/audio"
                ssrc=0D12F123:seq=1473; rtptime=80000
```

A.6. Capability Negotiation

This examples illustrate how the client and server determines their capability to support a special feature, in this case "play.scale". The server, through the clients request and the included Supported header, learns the client supports RTSP 2.0, and also supports the playback time scaling feature of RTSP. The server's response contains the following feature related information to the client; it supports the basic media delivery functions (play.basic), the extended functionality of time scaling of content (play.scale), and one "example.com" proprietary feature (com.example.flight). The client also learns the methods supported (Public header) by the server for the indicated resource.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1533
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
C->S: OPTIONS rtsp://media.example.com/movie/twister.3gp RTSP/2.0
      CSeq: 1
      Supported: play.basic, play.scale
      User-Agent: PhonyClient/1.2

S->C: RTSP/2.0 200 OK
      CSeq: 1
      Public: OPTIONS,SETUP,PLAY,PAUSE,TEARDOWN,DESCRIBE,GET_PARAMETER
      Allow: OPTIONS, SETUP, PLAY, PAUSE, TEARDOWN, DESCRIBE
      Server: PhonyServer/2.0
      Supported: play.basic, play.scale, com.example.flight
```

When the client sends its SETUP request it tells the server that it requires support of the play.scale feature for this session by including the Require header.

```
C->S: SETUP rtsp://media.example.com/twister.3gp/trackID=1 RTSP/2.0
      CSeq: 3
      User-Agent: PhonyClient/1.2
      Transport: RTP/AVP/UDP;unicast;
                 dest_addr="192.0.2.53:3056"/"192.0.2.53:3057",
                 RTP/AVP/TCP;unicast;interleaved=0-1
      Require: play.scale
      Accept-Ranges: NPT, SMPTE, UTC
      User-Agent: PhonyClient/1.2

S->C: RTSP/2.0 200 OK
      CSeq: 3
      Session: 12345678
      Transport: RTP/AVP/UDP;unicast;
         dest_addr="192.0.2.53:3056"/"192.0.2.53:3057";
         src_addr="198.51.100.5:5000"/"198.51.100.5:5001"
      Server: PhonyServer/2.0
      Accept-Ranges: NPT, SMPTE
      Media-Properties: Random-Access=0.8, Immutable, Unlimited
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1534
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix B. RTSP Protocol State Machine

The RTSP session state machine describes the behavior of the protocol from RTSP session initialization through RTSP session termination.

The State machine is defined on a per session basis which is uniquely identified by the RTSP session identifier. The session may contain one or more media streams depending on state. If a single media stream is part of the session it is in non-aggregated control. If two or more is part of the session it is in aggregated control.

The below state machine is a informative description of the protocols behavior. In case of ambiguity with the earlier parts of this specification, the description in the earlier parts take precedence.

B.1. States

The state machine contains three states, described below. For each state there exist a table which shows which requests and events are allowed and whether they will result in a state change.

Init: Initial state no session exists.

Ready: Session is ready to start playing.

Play: Session is playing, i.e. sending media stream data in the direction S->C.

B.2. State variables

This representation of the state machine needs more than its state to work. A small number of variables are also needed and is explained below.

NRM: The number of media streams part of this session.

RP: Resume point, the point in the presentation time line at which a request to continue playing will resume from. A time format for the variable is not mandated.

B.3. Abbreviations

To make the state tables more compact a number of abbreviations are used, which are explained below.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1535
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

IFI: IF Implemented.

md: Media

PP: Pause Point, the point in the presentation time line at which the presentation was paused.

Prs: Presentation, the complete multimedia presentation.

RedP: Redirect Point, the point in the presentation time line at which a REDIRECT was specified to occur.

SES: Session.

B.4. State Tables

This section contains a table for each state. The table contains all the requests and events that this state is allowed to act on. The events which is method names are, unless noted, requests with the given method in the direction client to server (C->S). In some cases there exist one or more requisite. The response column tells what type of response actions should be performed. Possible actions that is requested for an event includes: response codes, e.g. 200, headers that needs to be included in the response, setting of state variables, or setting of other session related parameters. The new state column tells which state the state machine changes to.

The response to a valid request meeting the requisites is normally a 2xx (SUCCESS) unless other noted in the response column. The exceptions need to be given a response according to the response column. If the request does not meet the requisite, is erroneous or some other type of error occur, the appropriate response code is to be sent. If the response code is a 4xx the session state is unchanged. A response code of 3rr will result in that the session is ended and its state is changed to Init. A response code of 304 results in no state change. However, there exist restrictions to when a 3rr response may be used. A 5xx response does not result in any change of the session state, except if the error is not possible to recover from. A unrecoverable error results in the ending of the session. As it in the general case can't be determined if it was a unrecoverable error or not the client will be required to test. In the case that the next request after a 5xx is responded with 454 (Session Not Found) the client knows that the session has ended. For any request message that cannot be responded to within the time defined in Section 10.4, a 100 response must be sent.

The server will timeout the session after the period of time specified in the SETUP response, if no activity from the client is

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1536
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

detected. Therefore there exist a timeout event for all states except Init.

In the case that NRM = 1 the presentation URI is equal to the media URI or a specified presentation URI. For NRM > 1 the presentation URI needs to be other than any of the medias that are part of the session. This applies to all states.

Event	Prerequisite	Response
DESCRIBE	Needs REDIRECT	3rr, Redirect
DESCRIBE		200, Session description
OPTIONS	Session ID	200, Reset session timeout timer
OPTIONS		200
SET_PARAMETER	Valid parameter	200, change value of parameter
GET_PARAMETER	Valid parameter	200, return value of parameter

Table 13: None state-machine changing events

The methods in Table 13 do not have any effect on the state machine or the state variables. However, some methods do change other session related parameters, for example SET_PARAMETER which will set the parameter(s) specified in its body. Also all of these methods that allows Session header will also update the keep-alive timer for the session.

Action	Requisite	New State	Response
SETUP		Ready	NRM=1, RP=0.0
SETUP	Needs Redirect	Init	3rr Redirect
S -> C: REDIRECT	No Session hdr	Init	Terminate all SES

Table 14: State: Init

The initial state of the state machine, see Table 14 can only be left by processing a correct SETUP request. As seen in the table the two

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1537
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

state variables are also set by a correct request. This table also shows that a correct SETUP can in some cases be redirected to another URI and/or server by a 3rr response.

Action	Requisite	New State	Response
SETUP	New URI	Ready	NRM +=1
SETUP	URI Setup prior	Ready	Change transport param
TEARDOWN	Prs URI,	Init	No session hdr, NRM = 0
TEARDOWN	md URI,NRM=1	Init	No Session hdr, NRM = 0
TEARDOWN	md URI,NRM>1	Ready	Session hdr, NRM -= 1
PLAY	Prs URI, No range	Play	Play from RP
PLAY	Prs URI, Range	Play	According to range
PLAY	md URI, NRM=1, Range	Play	According to range
PLAY	md URI, NRM=1	Play	Play from RP
PAUSE	Prs URI	Ready	Return PP
SC:REDIRECT	Terminate-Reason	Ready	Set RedP
SC:REDIRECT	No Terminate-Reason time parameter	Init	Session is removed
Timeout		Init	
RedP reached		Init	TEARDOWN of session

Table 15: State: Ready

In the Ready state, see Table 15, some of the actions are depending

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1538
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

on the number of media streams (NRM) in the session, i.e., aggregated or non-aggregated control. A SETUP request in the Ready state can either add one more media stream to the session or, if the media stream (same URI) already is part of the session, change the transport parameters. TEARDOWN is depending on both the Request-URI and the number of media stream within the session. If the Request-URI is the presentations URI the whole session is torn down. If a media URI is used in the TEARDOWN request and more than one media exist in the session, the session will remain and a session header is returned in the response. If only a single media stream remains in the session when performing a TEARDOWN with a media URI the session is removed. The number of media streams remaining after tearing down a media stream determines the new state.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1539
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Action	Requisite	New State	Response
PAUSE	Prs URI	Ready	Set RP to present point
End of media	All media	Play	Set RP = End of media
End of range		Play	Set RP = End of range
PLAY	Prs URI, No range	Play	Play from present point
PLAY	Prs URI, Range	Play	According to range
SC:PLAY_NOTIFY		Play	200
SETUP	New URI	Play	455
SETUP	Setuped URI	Play	455
SETUP	Setuped URI, IFI	Play	Change transport param.
TEARDOWN	Prs URI	Init	No session hdr
TEARDOWN	md URI,NRM=1	Init	No Session hdr, NRM=0
TEARDOWN	md URI	Play	455
SC:REDIRECT	Terminate Reason with Time parameter	Play	Set RedP
SC:REDIRECT		Init	Session is removed
RedP reached		Init	TEARDOWN of session
Timeout		Init	Stop Media playout

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1540
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Table 16: State: Play

The Play state table, see Table 16, contains a number of requests that needs a presentation URI (labeled as Prs URI) to work on (i.e., the presentation URI has to be used as the Request-URI). This is due to the exclusion of non-aggregated stream control in sessions with more than one media stream.

To avoid inconsistencies between the client and server, automatic state transitions are avoided. This can be seen at for example "End of media" event when all media has finished playing, the session still remain in Play state. An explicit PAUSE request needs to be sent to change the state to Ready. It may appear that there exist automatic transitions in "RedP reached" and "PP reached". However, they are requested and acknowledged before they take place. The time at which the transition will happen is known by looking at the range header. If the client sends a request close in time to these transitions it needs to be prepared for receiving error message, as the state may or may not have changed.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1541
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix C. Media Transport Alternatives

This section defines how certain combinations of protocols, profiles and lower transports are used. This includes the usage of the Transport header's source and destination address parameters "src_addr" and "dest_addr".

C.1. RTP

This section defines the interaction of RTSP with respect to the RTP protocol [RFC3550]. It also defines any necessary media transport signalling with regards to RTP.

The available RTP profiles and lower layer transports are described below along with rules on signalling the available combinations.

C.1.1. AVP

The usage of the "RTP Profile for Audio and Video Conferences with Minimal Control" [RFC3551] when using RTP for media transport over different lower layer transport protocols is defined below in regards to RTSP.

One such case is defined within this document, the use of embedded (interleaved) binary data as defined in Section 14. The usage of this method is indicated by include the "interleaved" parameter.

When using embedded binary data the "src_addr" and "dest_addr" MUST NOT be used. This addressing and multiplexing is used as defined with use of channel numbers and the interleaved parameter.

C.1.2. AVP/UDP

This part describes sending of RTP [RFC3550] over lower transport layer UDP [RFC0768] according to the profile "RTP Profile for Audio and Video Conferences with Minimal Control" defined in RFC 3551 [RFC3551]. This profile requires one or two uni- or bi-directional UDP flows per media stream. The first UDP flow is for RTP and the second is for RTCP. Embedding of RTP data with the RTSP messages, in accordance with Section 14, SHOULD NOT be performed when RTSP messages are transported over unreliable transport protocols, like UDP [RFC0768].

The RTP/UDP and RTCP/UDP flows can be established using the Transport header's "src_addr", and "dest_addr" parameters.

In RTSP PLAY mode, the transmission of RTP packets from client to server is unspecified. The behavior in regards to such RTP packets

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1542
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

MAY be defined in future.

The "src_addr" and "dest_addr" parameters are used in the following way for media delivery and playback mode, i.e. Mode=PLAY:

- The "src_addr" and "dest_addr" parameters MUST contain either 1 or 2 address specifications.
- Each address specification for RTP/AVP/UDP or RTP/AVP/TCP MUST contain either:
  - both an address and a port number, or
  - a port number without an address.
- The first address and port pair given in either of the parameters applies to the RTP stream. The second address and port pair if present applies to the RTCP stream.
- The RTP/UDP packets from the server to the client MUST be sent to the address and port given by first address and port pair of the "dest_addr" parameter.
- The RTCP/UDP packets from the server to the client MUST be sent to the address and port given by the second address and port pair of the "dest_addr" parameter. If no second pair is specified RTCP MUST NOT be sent.
- The RTCP/UDP packets from the client to the server MUST be sent to the address and port given by the second address and port pair of the "src_addr" parameter. If no second pair is given RTCP MUST NOT be sent.
- The RTP/UDP packets from the client to the server MUST be sent to the address and port given by the first address and port pair of the "src_addr" parameter.
- RTP and RTCP Packets SHOULD be sent from the corresponding receiver port, i.e. RTCP packets from server should be sent from the "src_addr" parameters second address port pair.

C.1.3. AVPF/UDP

The RTP profile "Extended RTP Profile for RTCP-based Feedback (RTP/AVPF)" [RFC4585] MAY be used as RTP profiles in session using RTP. All that is defined for AVP MUST also apply for AVPF.

The usage of AVPF is indicated by the media initialization protocol

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1543
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

used. In the case of SDP it is indicated by media lines (m=) containing the profile RTP/AVPF. That SDP MAY also contain further AVPF related SDP attributes configuring the AVPF session regarding reporting interval and feedback messages to be used. This configuration MUST be followed.

C.1.4. SAVP/UDP

The RTP profile "The Secure Real-time Transport Protocol (SRTP)" [RFC3711] is an RTP profile (SAVP) that MAY be used in RTSP sessions using RTP. All that is defined for AVP MUST also apply for SAVP.

The usage of SRTP requires that a security context is established. The default key-management unless otherwise signalled shall be MIKEY in RSA-R mode as defined in Appendix C.1.4.1, and not according to the procedure defined in "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)" [RFC4567]. The reason is that RFC 4567 sends the initial MIKEY message in SDP, thus both requiring the usage of the DESCRIBE method and forcing the server to keep state for client performing DESCRIBE in anticipation that they might require key management.

MIKEY is selected as default method for establishing SRTP cryptographic context within an RTSP session as it can be embedded in the RTSP messages, while still ensuring confidentiality of content of the keying material, even when using hop-by-hop TLS security for the RTSP messages. This method does also support pipelining of the RTSP messages.

C.1.4.1. MIKEY Key Establishment

This method for using MIKEY to establish the SRTP cryptographic context is initiated in the client's SETUP request, and the servers response to the SETUP carries the MIKEY response. Thus ensuring that the crypto context establishment happens simultaneously with the establishment of the media stream being protected. By using MIKEY's RSA-R mode [RFC4738] the client can be initiator and still allow the server to set the parameters in accordance with the actual media stream.

The SRTP cryptographic context establishment is done according to the following process:

1. The client determines that SAVP or SAVPF shall be used from media description format, e.g. SDP. If no other key management method is explicitly signalled, then MIKEY SHALL be used as defined here in. This specification does not specify an explicit method for indicating this SRTP cryptographic context

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1544
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

establishment method, but future specifications may.

2. The client SHALL establish a TLS connection for RTSP messages, directly or hop by hop with the server. If hop-by-hop TLS security is used, the User method SHALL be indicated in the Accept-Credentials header. We do note that using hop-by-hop does allow the proxy to insert itself as a man in the middle also in the MIKEY exchange by providing one of its certificates, rather than the server's in the Connection-Credentials header. The client shall therefore validate the server certificate.

3. The client retrieves the servers certificate from a direct TLS connection, or if hop by hop from Connection-Credentials header. The client then checks that the server certificate is valid and belongs to server.

4. The client forms the MIKEY Initiator message using RSA-R mode in unicast mode as specified in [RFC4738]. The client SHOULD use the same certificate for TLS and in MIKEY to enable server to bind the two together. The client's certificate SHALL be included in the MIKEY message. The client SHALL indicate its SRTP capabilities in the message.

5. The MIKEY message from the previous step is base64 [RFC4648] encoded and becomes the value of the MIKEY parameter that are included in the transport specification(s) that specifies a SRTP based profile (SAVP, SAVPF) in the SETUP request.

6. Any proxy encountering the MIKEY parameter SHALL forward it without modification. A proxy requiring to understand transport specification which doesn't support SAVP/SAVPF with MIKEY will discard the whole transport specification. Most types of proxy can easily support SAVP and SAVPF with MIKEY. If possible bypassing the proxy should be tried.

7. The server upon receiving the SETUP request, will need to decide upon the transport specification to use, if multiple are included by the client. In the determination of which transport specifications that are supported and preferred, the server should decode the MIKEY message to take the embedded SRTP parameters into account. If all transport specs require SRTP but no MIKEY parameter or other supported keying method is included, the server shall respond with 403.

8. Upon generating a response the following outcomes can occur:

   * A transport spec not using SRTP and MIKEY is selected. Thus the answer will not contain any MIKEY parameter.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1545
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

* A transport spec using SRTP and MIKEY is selected but an error is encountered in the MIKEY processing. In that case an RTSP error response code of 466 "Key Management Error" SHALL be used. A MIKEY message describing the error MAY be included.

* A transport spec using SRTP and MIKEY is selected and a MIKEY response message can be created. The server SHOULD use the same certificate for TLS and in MIKEY to enable client to bind the two together. If a different certificate is used it SHALL be included in the MIKEY message. It is RECOMMENDED that the envelope key cache type is set to 'Cache' and that a single envelope key is reused for all MIKEY messages to the client. That message is included in the MIKEY parameter part of the single selected transport specification in the SETUP response. The server will set the SRTP parameters as preferred for this media stream within the supported range by the client.

9. The server transmits the SETUP response back to the client.

10. The client receives the SETUP response and if the response code indicates a successful request it decodes the MIKEY message and establish the SRTP cryptographic context from the parameters in the MIKEY response.

In the above method the client's certificate may be self-signed in cases where the client's identify is not necessary to establish and the security goal is only to ensure that the RTSP signalling client is the same as the one receiving the SRTP security context.

C.1.5. SAVPF/UDP

The RTP profile "Extended Secure RTP Profile for RTCP-based Feedback (RTP/SAVPF)" [RFC5124] is an RTP profile (SAVPF) that MAY be used in RTSP sessions using RTP. All that is defined for AVP MUST also apply for SAVPF.

The usage of SRTP requires that a cryptographic context is established. The default mechanism for establishing that security association is to use MIKEY[RFC3830] with RTSP as defined Appendix C.1.4.1.

C.1.6. RTCP usage with RTSP

RTCP has several usages when RTP is used for media transport as explained below. Due to that RTCP MUST be supported if an RTSP agent handles RTP.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1546
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

C.1.6.1. Media synchronization

RTCP provides media synchronization and clock drift compensation. The initial media synchronization is available from RTP-Info header. However, to be able to handle any clock drift between the media streams, RTCP is needed.

C.1.6.2. RTSP Session keep-alive

RTCP traffic from the RTSP client to the RTSP server MUST function as keep-alive. Which requires an RTSP server supporting RTP to use the received RTCP packets as indications that the client desires the related RTSP session to be kept alive.

C.1.6.3. Bit-rate adaption

RTCP Receiver reports and any additional feedback from the client MUST be used adapt the bit-rate used over the transport for all cases when RTP is sent over UDP. An RTP sender without reserved resources MUST NOT use more than its fair share of the available resources. This can be determined by comparing on short to medium term (some seconds) the used bit-rate and adapt it so that the RTP sender sends at a bit-rate comparable to what a TCP sender would achieve on average over the same path.

C.1.6.4. RTP and RTCP Multiplexing

RTSP can be used to negotiate the usage of RTP and RTCP multiplexing as described in [RFC5761]. This allows servers and client to reduce the amount of resources required for the session by only requiring one underlying transport stream per media stream instead of two when using RTP and RTCP. This lessens the server port consumption and also the necessary state and keep-alive work when operating across Network and Address Translators [RFC2663].

Content must be prepared with some consideration for RTP and RTCP multiplexing, mainly ensuring that the RTP payload types used does not collide with the ones used for RTCP packet types this option likely needs explicit support from the content unless the RTP payload types can be remapped by the server and that is correctly reflected in the session description. Beyond that support of this feature should come at little cost and much gain.

It is recommended that if the content and server supports RTP and RTCP multiplexing that this is indicated in the session description, for example using the SDP attribute "a=rtcp-mux". If the SDP message contains the a=rtcp-mux attribute for a media stream, the server MUST support RTP and RTCP multiplexing. If indicated or otherwise desired

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1547
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

by the client it can include the Transport parameter "RTCP-mux" in any transport specification where it desires to use RTCP-mux. The server will indicate if it supports RTCP-mux. Server and Client SHOULD support RTP and RTCP multiplexing.

For capability exchange, an RTSP feature tag for RTP and RTCP multiplexing is defined: "setup.rtp.rtcp.mux".

C.2. RTP over TCP

Transport of RTP over TCP can be done in two ways, over independent TCP connections using RFC 4571 [RFC4571] or interleaved in the RTSP control connection. In both cases the protocol MUST be "rtp" and the lower layer MUST be TCP. The profile may be any of the above specified ones; AVP, AVPF, SAVP or SAVPF.

C.2.1. Interleaved RTP over TCP

The use of embedded (interleaved) binary data transported on the RTSP connection is possible as specified in Section 14. When using this declared combination of interleaved binary data the RTSP messages MUST be transported over TCP. TLS may or may not be used.

One should, however, consider that this will result that all media streams go through any proxy. Using independent TCP connections can avoid that issue.

C.2.2. RTP over independent TCP

In this Appendix, we describe the sending of RTP [RFC3550] over lower transport layer TCP [RFC0793] according to "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport" [RFC4571]. This Appendix adapts the guidelines for using RTP over TCP within SIP/SDP [RFC4145] to work with RTSP.

A client codes the support of RTP over independent TCP by specifying an RTP/AVP/TCP transport option without an interleaved parameter in the Transport line of a SETUP request. This transport option MUST include the "unicast" parameter.

If the client wishes to use RTP with RTCP, two ports (or two address/port pairs) are specified by the dest_addr parameter. If the client wishes to use RTP without RTCP, one port (or one address/port pair) is specified by the dest_addr parameter. Ordering rules of dest_addr ports follow the rules for RTP/AVP/UDP.

If the client wishes to play the active role in initiating the TCP

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1548
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

connection, it MAY set the "setup" parameter (See Section 16.52) on the Transport line to be "active", or it MAY omit the setup parameter, as active is the default. If the client signals the active role, the ports for all dest_addr values MUST be set to 9 (the discard port).

If the client wishes to play the passive role in TCP connection initiation, it MUST set the "setup" parameter on the Transport line to be "passive". If the client is able to assume the active or the passive role, it MUST set the "setup" parameter on the Transport line to be "actpass". In either case, the dest_addr port value for RTP MUST be set to the TCP port number on which the client is expecting to receive the RTP stream connection, and the dest_addr port value for RTCP MUST be set to the TCP port number on which the client is expecting to receive the RTCP stream connection.

If upon receipt of a non-interleaved RTP/AVP/TCP SETUP request, a server decides to accept this requested option, the 2xx reply MUST contain a Transport option that specifies RTP/AVP/TCP (without using the interleaved parameter, and with using the unicast parameter). The dest_addr parameter value MUST be echoed from the parameter value in the client request unless the destination address (only port) was not provided in which can the server MAY include the source address of the RTSP TCP connection with the port number unchanged.

In addition, the server reply MUST set the setup parameter on the Transport line, to indicate the role the server will play in the connection setup. Permissible values are "active" (if a client set "setup" to "passive" or "actpass") and "passive" (if a client set "setup" to "active" or "actpass").

If a server sets "setup" to "passive", the "src_addr" in the reply MUST indicate the ports the server is willing to receive an RTP connection and (if the client requested an RTCP connection by specifying two dest_addr ports or address/port pairs) and RTCP connection. If a server sets "setup" to "active", the ports specified in "src_addr" MUST be set to 9. The server MAY use the "ssrc" parameter, following the guidance in Section 16.52. Port ordering for src_addr follows the rules for RTP/AVP/UDP.

Servers MUST support taking the passive role and MAY support taking the active role. Servers with a public IP address takes the passive role, thus enabling clients behind NATs and Firewalls to better chance of succesful connect to the server by actively connecting outwards. Therefore the clients are RECOMMENDED to take the active role.

After sending (receiving) a 2xx reply for a SETUP method for a non-

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1549
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

interleaved RTP/AVP/TCP media stream, the active party SHOULD initiate the TCP connection as soon as possible. The client MUST NOT send a PLAY request prior to the establishment of all the TCP connections negotiated using SETUP for the session. In case the server receives a PLAY request in a session that has not yet established all the TCP connections, it MUST respond using the 464 "Data Transport Not Ready Yet" (Section 15.4.29) error code.

Once the PLAY request for a media resource transported over non-interleaved RTP/AVP/TCP occurs, media begins to flow from server to client over the RTP TCP connection, and RTCP packets flow bidirectionally over the RTCP TCP connection. As in the RTP/UDP case, client to server traffic on the TCP port is unspecified by this memo. The packets that travel on these connections MUST be framed using the protocol defined in [RFC4571], not by the framing defined for interleaving RTP over the RTSP control connection defined in Section 14.

A successful PAUSE request for a media being transported over RTP/AVP/TCP pauses the flow of packets over the connections, without closing the connections. A successful TEARDOWN request signals that the TCP connections for RTP and RTCP are to be closed as soon as possible.

Subsequent SETUP requests on an already-SETUP RTP/AVP/TCP URI may be ambiguous in the following way: does the client wish to open up new TCP RTP and RTCP connections for the URI, or does the client wish to continue using the existing TCP RTP and RTCP connections? The client SHOULD use the "connection" parameter (defined in Section 16.52) on the Transport line to make its intention clear in the regard (by setting "connection" to "new" if new connections are needed, and by setting "connection" to "existing" if the existing connections are to be used). After a 2xx reply for a SETUP request for a new connection, parties should close the pre-existing connections, after waiting a suitable period for any stray RTP or RTCP packets to arrive.

The usage of SRTP, i.e. either SAVP or SAVPF profiles requires that a security association is established. The default mechanism for establishing that security association is to use MIKEY[RFC3830] with RTSP as defined Appendix C.1.4.1.

Below, we rewrite part of the example media on demand example shown in Appendix A.1 to use RTP/AVP/TCP non-interleaved:

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1550
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
C->M: DESCRIBE rtsp://example.com/twister.3gp RTSP/2.0
      CSeq: 1
      User-Agent: PhonyClient/1.2

M->C: RTSP/2.0 200 OK
      CSeq: 1
      Server: PhonyServer/1.0
      Date: Thu, 23 Jan 1997 15:35:06 GMT
      Content-Type: application/sdp
      Content-Length: 227
      Content-Base: rtsp://example.com/twister.3gp/
      Expires: 24 Jan 1997 15:35:06 GMT

      v=0
      o=- 2890844256 2890842807 IN IP4 198.51.100.34
      s=RTSP Session
      i=An Example of RTSP Session Usage
      e=adm@example.com
      c=IN IP4 0.0.0.0
      a=control: *
      a=range: npt=0-0:10:34.10
      t=0 0
      m=audio 0 RTP/AVP 0
      a=control: trackID=1

C->M: SETUP rtsp://example.com/twister.3gp/trackID=1 RTSP/2.0
      CSeq: 2
      User-Agent: PhonyClient/1.2
      Require: play.basic
      Transport: RTP/AVP/TCP;unicast;dest_addr=":9"/":9";
                 setup=active;connection=new
      Accept-Ranges: NPT, SMPTE, UTC
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1551
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
M->C: RTSP/2.0 200 OK
      CSeq: 2
      Server: PhonyServer/1.0
      Transport: RTP/AVP/TCP;unicast;
                 dest_addr=":9"/":9";
                 src_addr="198.51.100.5:53478"/"198.51.100:54091";
                 setup=passive;connection=new;ssrc=93CB001E
      Session: 12345678
      Expires: 24 Jan 1997 15:35:12 GMT
      Date: 23 Jan 1997 15:35:12 GMT
      Accept-Ranges: NPT
      Media-Properties: Random-Access=0.8, Immutable, Unlimited

C->M: TCP Connection Establishment x2

C->M: PLAY rtsp://example.com/twister.3gp/ RTSP/2.0
      CSeq: 4
      User-Agent: PhonyClient/1.2
      Range: npt=30-
      Session: 12345678

M->C: RTSP/2.0 200 OK
      CSeq: 4
      Server: PhonyServer/1.0
      Date: 23 Jan 1997 15:35:14 GMT
      Session: 12345678
      Range: npt=30-623.10
      Seek-Style: First-Prior
      RTP-Info:  url="rtsp://example.com/twister.3gp/trackID=1"
         ssrc=4F312DD8:seq=54321;rtptime=2876889
```

C.3. Handling Media Clock Time Jumps in the RTP Media Layer

RTSP allows media clients to control selected, non-contiguous sections of media presentations, rendering those streams with an RTP media layer [RFC3550]. Two cases occur, the first is when a new PLAY request replaces an old ongoing request and the new request results in a jump in the media. This should produce in the RTP layer a continuous media stream. A client may also directly following a completed PLAY request perform a new PLAY request. This will result in some gap in the media layer. The below text will look into both cases.

A PLAY request that replaces a ongoing request allows the media layer rendering the RTP stream without being affected by jumps in media clock time. The RTP timestamps for the new media range is set so that they become continuous with the previous media range in the previous request. The RTP sequence number for the first packet in

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1552
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

the new range will be the next following the last packet in the previous range, i.e. monotonically increasing. The goal is to allow the media rendering layer to work without interruption or reconfiguration across the jumps in media clock. This should be possible in all cases of replaced PLAY requests for media that has random-access properties. In this case care is needed to align frames or similar media dependent structures.

In cases where jumps in media clock time are a result of RTSP signalling operations arriving after a completed PLAY operation, the request timing will result in that media becomes non-continuous. The server becomes unable to send the media so that it arrive timely and still carry timestamps to make the media stream continuous. In these cases the server will produce RTP streams where there are gaps in the RTP timeline for the media. In such cases, if the media has frame structure, aligning the timestamp for the next frame with the previous structure reduces the burden to render this media. The gap should represent the time the server hasn't been serving media, e.g. the time between the end of the media stream or a PAUSE request and the new PLAY request. In these cases the RTP sequence number would normally be monotonically increasing across the gap.

For RTSP sessions with media that lacks random access properties, like live streams, any media clock jump is commonly result of correspondingly long pause of delivery. The RTP timestamp will have increased in direct proportion to the duration of the paused delivery. Note also that in this case the RTP sequence number should be the next packet number. If not, the RTCP packet loss reporting will indicate as loss all packets not received between the point of pausing and later resuming. This may trigger congestion avoidance mechanisms. An allowed exception from the above recommendation on monotonically increasing RTP sequence number is live media streams, likely being relayed. In this case, when the client resumes delivery, it will get the media that is currently being delivered to the server itself. For this type of basic delivery of live streams to multiple users over unicast, individual rewriting of RTP sequence numbers becomes quite a burden. For solutions that anyway caches media, timeshifts, etc, the rewriting should be a minor issue.

The goal when handling jumps in media clock time is that the provided stream is continuous without gaps in RTP timestamp or sequence number. However, when delivery has been halted for some reason the RTP timestamp when resuming MUST represent the duration the delivery was halted. RTP sequence number MUST generally be the next number, i.e. monotonically increasing modulo 65536. For media resources with the properties Time-Progressing and Time-Duration=0.0 the server MAY create RTP media streams with RTP sequence number jumps in them due to client first halting delivery and later resuming it (PAUSE and

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1553
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

then later PLAY). However, servers utilizing this exception must take into consideration the resulting RTCP receiver reports that likely contains loss report for all the packets part of the discontinuity. A client can not rely on that a server will align when resuming playing even if it is RECOMMENDED. The RTP-Info header will provide information on how the server acts in each case.

We cannot assume that the RTSP client can communicate with the RTP media agent, as the two may be independent processes. If the RTP timestamp shows the same gap as the NPT, the media agent will assume that there is a pause in the presentation. If the jump in NPT is large enough, the RTP timestamp may roll over and the media agent may believe later packets to be duplicates of packets just played out. Having the RTP timestamp jump will also affect the RTCP measurements based on this.

As an example, assume a RTP timestamp frequency of 8000 Hz, a packetization interval of 100 ms and an initial sequence number and timestamp of zero.

```
C->S: PLAY rtsp://example.com/fizzle RTSP/2.0
  CSeq: 4
  Session: abcdefgh
  Range: npt=10-15
  User-Agent: PhonyClient/1.2

S->C: RTSP/2.0 200 OK
  CSeq: 4
  Session: abcdefgh
  Range: npt=10-15
  RTP-Info: url="rtsp://example.com/fizzle/audiotrack"
            ssrc=0D12F123:seq=0;rtptime=0
```

The ensuing RTP data stream is depicted below:

```
S -> C: RTP packet - seq = 0,  rtptime = 0,     NPT time = 10s
S -> C: RTP packet - seq = 1,  rtptime = 800,   NPT time = 10.1s
 . . .
S -> C: RTP packet - seq = 49, rtptime = 39200, NPT time = 14.9s
```

Upon the completion of the requested delivery the server sends a PLAY_NOTIFY

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1554
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
      S->C: PLAY_NOTIFY rtsp://example.com/fizzle RTSP/2.0
            CSeq: 5
            Notify-Reason: end-of-stream
            Request-Status: cseq=4 status=200 reason="OK"
            Range: npt=-15
            RTP-Info:url="rtsp://example.com/fizzle/audiotrack"
               ssrc=0D12F123:seq=49;rtptime=39200
            Session: abcdefgh

      C->S: RTSP/2.0 200 OK
            CSeq: 5
            User-Agent: PhonyClient/1.2
```

Upon the completion of the play range, the client follows up with a request to PLAY from a new NPT.

```
C->S: PLAY rtsp://example.com/fizzle RTSP/2.0
      CSeq: 6
      Session: abcdefg
      Range: npt=18-20
      User-Agent: PhonyClient/1.2

S->C: RTSP/2.0 200 OK
      CSeq: 6
      Session: abcdefg
      Range: npt=18-20
      RTP-Info: url="rtsp://example.com/fizzle/audiotrack"
                ssrc=0D12F123:seq=50;rtptime=40100
```

The ensuing RTP data stream is depicted below:

```
   S->C: RTP packet - seq = 50, rtptime = 40100, NPT time = 18s
   S->C: RTP packet - seq = 51, rtptime = 40900, NPT time = 18.1s
    . . .
   S->C: RTP packet - seq = 69, rtptime = 55300, NPT time = 19.9s
```

In this example, first, NPT 10 through 15 is played, then the client request the server to skip ahead and play NPT 18 through 20. The first segment is presented as RTP packets with sequence numbers 0 through 49 and timestamp 0 through 39,200. The second segment consists of RTP packets with sequence number 50 through 69, with timestamps 40,100 through 55,200. While there is a gap in the NPT, there is no gap in the sequence number space of the RTP data stream.

The RTP timestamp gap is present in the above example due to the time it takes to perform the second play request, in this case 12.5 ms (100/8000).

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1555
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

C.4. Handling RTP Timestamps after PAUSE

During a PAUSE / PLAY interaction in an RTSP session, the duration of time for which the RTP transmission was halted MUST be reflected in the RTP timestamp of each RTP stream. The duration can be calculated for each RTP stream as the time elapsed from when the last RTP packet was sent before the PAUSE request was received and when the first RTP packet was sent after the subsequent PLAY request was received. The duration includes all latency incurred and processing time required to complete the request.

> The RTP RFC [RFC3550] states that: The RTP timestamp for each unit [packet] would be related to the wallclock time at which the unit becomes current on the virtual presentation timeline.
>
> In order to satisfy the requirements of [RFC3550], the RTP timestamp space needs to increase continuously with real time. While this is not optimal for stored media, it is required for RTP and RTCP to function as intended. Using a continuous RTP timestamp space allows the same timestamp model for both stored and live media and allows better opportunity to integrate both types of media under a single control.

As an example, assume a clock frequency of 8000 Hz, a packetization interval of 100 ms and an initial sequence number and timestamp of zero.

```
C->S: PLAY rtsp://example.com/fizzle RTSP/2.0
      CSeq: 4
      Session: abcdefg
      Range: npt=10-15
      User-Agent: PhonyClient/1.2

S->C: RTSP/2.0 200 OK
      CSeq: 4
      Session: abcdefg
      Range: npt=10-15
      RTP-Info: url="rtsp://example.com/fizzle/audiotrack"
                ssrc=0D12F123:seq=0;rtptime=0
```

The ensuing RTP data stream is depicted below:

```
   S -> C: RTP packet - seq = 0, rtptime = 0,    NPT time = 10s
   S -> C: RTP packet - seq = 1, rtptime = 800,  NPT time = 10.1s
   S -> C: RTP packet - seq = 2, rtptime = 1600, NPT time = 10.2s
   S -> C: RTP packet - seq = 3, rtptime = 2400, NPT time = 10.3s
```

The client then sends a PAUSE request:

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1556
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
C->S: PAUSE rtsp://example.com/fizzle RTSP/2.0
      CSeq: 5
      Session: abcdefg
      User-Agent: PhonyClient/1.2

S->C: RTSP/2.0 200 OK
      CSeq: 5
      Session: abcdefg
      Range: npt=10.4-15
```

20 seconds elapse and then the client sends a PLAY request. In addition the server requires 15 ms to process the request:

```
C->S: PLAY rtsp://example.com/fizzle RTSP/2.0
      CSeq: 6
      Session: abcdefg
      User-Agent: PhonyClient/1.2

S->C: RTSP/2.0 200 OK
      CSeq: 6
      Session: abcdefg
      Range: npt=10.4-15
      RTP-Info: url="rtsp://example.com/fizzle/audiotrack"
                ssrc=0D12F123:seq=4;rtptime=164400
```

The ensuing RTP data stream is depicted below:

```
   S -> C: RTP packet - seq = 4, rtptime = 164400, NPT time = 10.4s
   S -> C: RTP packet - seq = 5, rtptime = 165200, NPT time = 10.5s
   S -> C: RTP packet - seq = 6, rtptime = 166000, NPT time = 10.6s
```

First, NPT 10 through 10.3 is played, then a PAUSE is received by the server. After 20 seconds a PLAY is received by the server which take 15ms to process. The duration of time for which the session was paused is reflected in the RTP timestamp of the RTP packets sent after this PLAY request.

A client can use the RTSP range header and RTP-Info header to map NPT time of a presentation with the RTP timestamp.

Note: In RFC 2326 [RFC2326], this matter was not clearly defined and was misunderstood commonly. However, for RTSP 2.0 it is expected that this will be handled correctly and no exception handling will be required.

Note Further: To ensure correct media decoding and usually jitter-buffer handling reseting some of the state when issuing a PLAY request is needed.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1557
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

C.5. RTSP / RTP Integration

For certain datatypes, tight integration between the RTSP layer and the RTP layer will be necessary. This by no means precludes the above restrictions. Combined RTSP/RTP media clients should use the RTP-Info field to determine whether incoming RTP packets were sent before or after a seek or before or after a PAUSE.

C.6. Scaling with RTP

For scaling (see Section 16.44), RTP timestamps should correspond to the rendering timing. For example, when playing video recorded at 30 frames/second at a scale of two and speed (Section 16.48) of one, the server would drop every second frame to maintain and deliver video packets with the normal timestamp spacing of 3,000 per frame, but NPT would increase by 1/15 second for each video frame.

> Note: The above scaling puts requirements on the media codec or a media stream to support it. For example motion JPEG or other non-predictive video coding can easier handle the above example.

C.7. Maintaining NPT synchronization with RTP timestamps

The client can maintain a correct display of NPT (Normal Play Time) by noting the RTP timestamp value of the first packet arriving after repositioning. The sequence parameter of the RTP-Info (Section 16.43) header provides the first sequence number of the next segment.

C.8. Continuous Audio

For continuous audio, the server SHOULD set the RTP marker bit at the beginning of serving a new PLAY request or at jumps in timeline. This allows the client to perform playout delay adaptation.

C.9. Multiple Sources in an RTP Session

Note that more than one SSRC MAY be sent in the media stream. If it happens all sources are expected to be rendered simultaneously.

C.10. Usage of SSRCs and the RTCP BYE Message During an RTSP Session

The RTCP BYE message indicates the end of use of a given SSRC. If all sources leave an RTP session, it can, in most cases, be assumed to have ended. Therefore, a client or server MUST NOT send a RTCP BYE message until it has finished using a SSRC. A server SHOULD keep using a SSRC until the RTP session is terminated. Prolonging the use of a SSRC allows the established synchronization context associated

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1558
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

with that SSRC to be used to synchronize subsequent PLAY requests even if the PLAY response is late.

An SSRC collision with the SSRC that transmits media does also have consequences, as it will normally force the media sender to change its SSRC in accordance with the RTP specification[RFC3550]. However, a RTSP server may wait and see if the client changes and thus resolve the conflict to minimize the impact. As media sender SSRC change will result in a loss of synchronization context, and require any receiver to wait for RTCP sender reports for all media requiring synchronization before being able to play out synchronized. Due to these reasons a client joining a session should take care to not select the same SSRC(s) as the server indicates in the ssrc Transport header parameter. Any SSRC signalled in the Transport header MUST be avoided. A client detecting a collision prior to sending any RTP or RTCP messages SHALL also select a new SSRC.

C.11. Future Additions

It is the intention that any future protocol or profile regarding both for media delivery and lower transport should be easy to add to RTSP. This section provides the necessary steps that needs to be meet.

The following things needs to be considered when adding a new protocol or profile for use with RTSP:

- The protocol or profile needs to define a name tag representing it. This tag is required to be a ABNF "token" to be possible to use in the Transport header specification.

- The useful combinations of protocol, profiles and lower layer transport for this extension needs to be defined. For each combination declare the necessary parameters to use in the Transport header.

- For new media protocols the interaction with RTSP needs to be addressed. One important factor will be the media synchronization. May need new headers similar to RTP info to carry information.

- Discuss congestion control for media, especially if transport without built in congestion control is used.

See the IANA section (Section 22) for information how to register new attributes.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1559
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix D. Use of SDP for RTSP Session Descriptions

The Session Description Protocol (SDP, [RFC4566]) may be used to describe streams or presentations in RTSP. This description is typically returned in reply to a DESCRIBE request on an URI from a server to a client, or received via HTTP from a server to a client.

This appendix describes how an SDP file determines the operation of an RTSP session. SDP as is provides no mechanism by which a client can distinguish, without human guidance, between several media streams to be rendered simultaneously and a set of alternatives (e.g., two audio streams spoken in different languages). The SDP extension "Grouping of Media Lines in the Session Description Protocol (SDP)" [RFC5888] provides such functionality to some degree. Appendix D.4 describes the usage of SDP media line grouping for RTSP.

D.1. Definitions

The terms "session-level", "media-level" and other key/attribute names and values used in this appendix are to be used as defined in SDP[RFC4566]:

D.1.1. Control URI

The "a=control:" attribute is used to convey the control URI. This attribute is used both for the session and media descriptions. If used for individual media, it indicates the URI to be used for controlling that particular media stream. If found at the session level, the attribute indicates the URI for aggregate control (presentation URI). The session level URI MUST be different from any media level URI. The presence of a session level control attribute MUST be interpreted as support for aggregated control. The control attribute MUST be present on media level unless the presentation only contains a single media stream, in which case the attribute MAY only be present on the session level and then also apply to that single media level.

ABNF for the attribute is defined in Section 20.3.

Example:

```
a=control:rtsp://example.com/foo
```

This attribute MAY contain either relative or absolute URIs, following the rules and conventions set out in RFC 3986 [RFC3986]. Implementations MUST look for a base URI in the following order:

1. the RTSP Content-Base field;

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1560
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

2. the RTSP Content-Location field;

3. the RTSP Request-URI.

If this attribute contains only an asterisk (*), then the URI MUST be treated as if it were an empty embedded URI, and thus inherit the entire base URI.

> Note, RFC 2326 was very unclear on the processing of relative URI and several RTSP 1.0 implementations at the point of publishing this document did not perform RFC 3986 processing to determine the resulting URI, instead simple concatenation is common. To avoid this issue completely it is recommended to use absolute URI in the SDP.

The URI handling for SDPs from container files need special consideration. For example lets assume that a container file has the URI: "rtsp://example.com/container.mp4". Lets further assume this URI is the base URI, and that there is a absolute media level URI: "rtsp://example.com/container.mp4/trackID=2". A relative media level URI that resolves in accordance with RFC 3986 [RFC3986] to the above given media URI is: "container.mp4/trackID=2". It is usually not desirable to need to include in or modify the SDP stored within the container file with the server local name of the container file. To avoid this, one can modify the base URI used to include a trailing slash, e.g. "rtsp://example.com/container.mp4/". In this case the relative URI for the media will only need to be: "trackID=2". However, this will also mean that using "*" in the SDP will result in control URI including the trailing slash, i.e. "rtsp://example.com/container.mp4/".

> Note: The usage of TrackID in the above is not an standardized form, but one example out of several similar strings such as TrackID, Track_ID, StreamID that is used by different server vendors to indicate a particular piece of media inside a container file.

D.1.2. Media Streams

The "m=" field is used to enumerate the streams. It is expected that all the specified streams will be rendered with appropriate synchronization. If the session is over multicast, the port number indicated SHOULD be used for reception. The client MAY try to override the destination port, through the Transport header. The servers MAY allow this, the response will indicate if allowed or not. If the session is unicast, the port numbers are the ones RECOMMENDED by the server to the client, about which receiver ports to use; the client MUST still include its receiver ports in its SETUP request.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1561
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The client MAY ignore this recommendation. If the server has no preference, it SHOULD set the port number value to zero.

The "m=" lines contain information about which transport protocol, profile, and possibly lower-layer is to be used for the media stream. The combination of transport, profile and lower layer, like RTP/AVP/UDP needs to be defined for how to be used with RTSP. The currently defined combinations are defined in Appendix C, further combinations MAY be specified.

Example:

```
m=audio 0 RTP/AVP 31
```

D.1.3. Payload Type(s)

The payload type(s) are specified in the "m=" line. In case the payload type is a static payload type from RFC 3551 [RFC3551], no other information may be required. In case it is a dynamic payload type, the media attribute "rtpmap" is used to specify what the media is. The "encoding name" within the "rtpmap" attribute may be one of those specified in RFC 3551 (Sections 5 and 6), or an MIME type registered with IANA, or an experimental encoding as specified in SDP (RFC 4566 [RFC4566]). Codec-specific parameters are not specified in this field, but rather in the "fmtp" attribute described below.

The selection of the RTP payload type numbers used may be required to consider RTP and RTCP Multiplexing [RFC5761] if that is to be supported by the server.

D.1.4. Format-Specific Parameters

Format-specific parameters are conveyed using the "fmtp" media attribute. The syntax of the "fmtp" attribute is specific to the encoding(s) that the attribute refers to. Note that some of the format specific parameters may be specified outside of the fmtp parameters, like for example the "ptime" attribute for most audio encodings.

D.1.5. Directionality of media stream

The SDP attributes "a=sendrecv", "a=recvonly" and "a=sendonly" provides instructions on which direction the media streams flow within a session. When using RTSP the SDP can be delivered to a client using either RTSP DESCRIBE or a number of RTSP external methods, like HTTP, FTP, and email. Based on this the SDP applies to how the RTSP client will see the complete session. Thus for media streams delivered from the RTSP server to the client would be given the "a=recvonly" attribute.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1562
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The direction attributes are not commonly used in SDPs for RTSP, but may occur. "a=recvonly" in a SDP provided to the RTSP client MUST indicate that media delivery will only occur in the direction from the RTSP server to the client. In SDP provided to the RTSP client that lacks any of the directionality attributes (a=recvonly, a=sendonly, a=sendrecv) MUST behave as if the "a=recvonly" attribute was received. Note that this overrules the normal default rule defined in SDP[RFC4566]. The usage of "a=sendonly" or "a=sendrecv" is not defined, nor is the interpretation of SDP by other entities than the RTSP client.

D.1.6. Range of Presentation

The "a=range" attribute defines the total time range of the stored session or an individual media. Non-seekable live sessions can be indicated as specified below, while the length of live sessions can be deduced from the "t" and "r" SDP parameters.

The attribute is both a session and a media level attribute. For presentations that contains media streams of the same durations, the range attribute SHOULD only be used at session-level. In case of different length the range attribute MUST be given at media level for all media, and SHOULD NOT be given at session level. If the attribute is present at both media level and session level the media level values MUST be used.

Note: Usually one will specify the same length for all media, even if there isn't media available for the full duration on all media. However, that requires that the server accepts PLAY requests within that range.

Servers MUST take care to provide RTSP Range (see Section 16.38) values that are consistent with what is presented in the SDP for the content. There is no reason for non dynamic content, like media clips provided on demand to have inconsistent values. Inconsistent values between the SDP and the actual values for the content handled by the server is likely to generate some failure, like 457 "Invalid Range", in case the client uses PLAY requests with a Range header. In case the content is dynamic in length and it is infeasible to provide a correct value in the SDP the server is recommended to describe this as non-seekable content (see below). The server MAY override that property in the response to a PLAY request using the correct values in the Range header.

The unit is specified first, followed by the value range. The units and their values are as defined in Section 4.4, Section 4.5 and Section 4.6 and MAY be extended with further formats. Any open ended range (start-), i.e. without stop range, is of unspecified duration

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1563
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

and MUST be considered as non-seekable content unless this property is overridden. Multiple instances carrying different clock formats MAY be included at either session or media level.

ABNF for the attribute is defined in Section 20.3.

Examples:

```
a=range:npt=0-34.4368
a=range:clock=19971113T211503Z-19971113T220300Z
Non seekable stream of unknown duration:
a=range:npt=0-
```

D.1.7. Time of Availability

The "t=" field defines when the SDP is valid. For on-demand content the server SHOULD indicate a stop time value for which it guarantees the description to be valid, and a start time that is equal to or before the time at which the DESCRIBE request was received. It MAY also indicate start and stop times of 0, meaning that the session is always available.

For sessions that are of live type, i.e. specific start time, unknown stop time, likely unseekable, the "t=" and "r=" field SHOULD be used to indicate the start time of the event. The stop time SHOULD be given so that the live event will have ended at that time, while still not be unnecessary long into the future.

D.1.8. Connection Information

In SDP, the "c=" field contains the destination address for the media stream. If a multicast address is specified the client SHOULD use this address in any SETUP request as destination address, including any additional parameters, such as TTL. For on-demand unicast streams and some multicast streams, the destination address MAY be specified by the client via the SETUP request, thus overriding any specified address. To identify streams without a fixed destination address, where the client is required to specify a destination address, the "c=" field SHOULD be set to a null value. For addresses of type "IP4", this value MUST be "0.0.0.0", and for type "IP6", this value MUST be "0:0:0:0:0:0:0:0" (can also be written as "::"), i.e. the unspecified address according to RFC 4291 [RFC4291].

D.1.9. Message Body Tag

The optional "a=mtag" attribute identifies a version of the session description. It is opaque to the client. SETUP requests may include this identifier in the If-Match field (see Section 16.23) to only allow session establishment if this attribute value still corresponds

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1564
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

to that of the current description. The attribute value is opaque and may contain any character allowed within SDP attribute values.

ABNF for the attribute is defined in Section 20.3.

Example:

```
a=mtag:"158bb3e7c7fd62ce67f12b533f06b83a"
```

One could argue that the "o=" field provides identical functionality. However, it does so in a manner that would put constraints on servers that need to support multiple session description types other than SDP for the same piece of media content.

D.2. Aggregate Control Not Available

If a presentation does not support aggregate control no session level "a=control:" attribute is specified. For a SDP with multiple media sections specified, each section will have its own control URI specified via the "a=control:" attribute.

Example:

```
v=0
o=- 2890844256 2890842807 IN IP4 192.0.2.56
s=I came from a web page
e=adm@example.com
c=IN IP4 0.0.0.0
t=0 0
m=video 8002 RTP/AVP 31
a=control:rtsp://audio.example.com/movie.aud
m=audio 8004 RTP/AVP 3
a=control:rtsp://video.example.com/movie.vid
```

Note that the position of the control URI in the description implies that the client establishes separate RTSP control sessions to the servers audio.example.com and video.example.com.

It is recommended that an SDP file contains the complete media initialization information even if it is delivered to the media client through non-RTSP means. This is necessary as there is no mechanism to indicate that the client should request more detailed media stream information via DESCRIBE.

D.3. Aggregate Control Available

In this scenario, the server has multiple streams that can be controlled as a whole. In this case, there are both a media-level "a=control:" attributes, which are used to specify the stream URIs,

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1565
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

and a session-level "a=control:" attribute which is used as the Request-URI for aggregate control. If the media-level URI is relative, it is resolved to absolute URIs according to Appendix D.1.1 above.

Example:

```
C->M: DESCRIBE rtsp://example.com/movie RTSP/2.0
      CSeq: 1
      User-Agent: PhonyClient/1.2

M->C: RTSP/2.0 200 OK
      CSeq: 1
      Date: Thu, 23 Jan 1997 15:35:06 GMT
      Expires: Thu, 23 Jan 1997 16:35:06 GMT
      Content-Type: application/sdp
      Content-Base: rtsp://example.com/movie/
      Content-Length: 227

      v=0
      o=- 2890844256 2890842807 IN IP4 192.0.2.211
      s=I contain
      i=<more info>
      e=adm@example.com
      c=IN IP4 0.0.0.0
      a=control:*
      t=0 0
      m=video 8002 RTP/AVP 31
      a=control:trackID=1
      m=audio 8004 RTP/AVP 3
      a=control:trackID=2
```

In this example, the client is recommended to establish a single RTSP session to the server, and uses the URIs rtsp://example.com/movie/trackID=1 and rtsp://example.com/movie/trackID=2 to set up the video and audio streams, respectively. The URI rtsp://example.com/movie/, which is resolved from the "*", controls the whole presentation (movie).

A client is not required to issues SETUP requests for all streams within an aggregate object. Servers should allow the client to ask for only a subset of the streams.

D.4. Grouping of Media Lines in SDP

For some types media it is desirable to express a relationship between various media components, for instance, for lip synchronization or Scalable Video Codec (SVC) [RFC5583]. This relationship is expressed on the SDP level by grouping of media

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1566
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

lines, as described in [RFC5888] and can be exposed to RTSP.

For RTSP it is mainly important to know how to handle grouped medias received by means of SDP, i.e., if the media are under aggregate control (see Appendix D.3) or if aggregate control is not available (see Appendix D.2).

It is RECOMMENDED that grouped medias are handled by aggregate control, to give the client the ability to control either the whole presentation or single medias.

Editor's note: how should the dependencies in [RFC5583] be handled in RTSP?

D.5. RTSP external SDP delivery

There are some considerations that need to be made when the session description is delivered to the client outside of RTSP, for example via HTTP or email.

First of all, the SDP needs to contain absolute URIs, since relative will in most cases not work as the delivery will not correctly forward the base URI.

The writing of the SDP session availability information, i.e. "t=" and "r=", needs to be carefully considered. When the SDP is fetched by the DESCRIBE method, the probability that it is valid is very high. However, the same are much less certain for SDPs distributed using other methods. Therefore the publisher of the SDP should take care to follow the recommendations about availability in the SDP specification [RFC4566].

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1567
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix E. RTSP Use Cases

This Appendix describes the most important and considered use cases for RTSP. They are listed in descending order of importance in regards to ensuring that all necessary functionality is present. This specification only fully supports usage of the two first. Also in these first two cases, there are special cases or exceptions that are not supported without extensions, e.g. the redirection of media delivery to another address than the controlling agent's (client's).

E.1. On-demand Playback of Stored Content

An RTSP capable server stores content suitable for being streamed to a client. A client desiring playback of any of the stored content uses RTSP to set up the media transport required to deliver the desired content. RTSP is then used to initiate, halt and manipulate the actual transmission (playout) of the content. RTSP is also required to provide necessary description and synchronization information for the content.

The above high level description can be broken down into a number of functions that RTSP needs to be capable of.

Presentation Description: Provide initialization information about the presentation (content); for example, which media codecs are needed for the content. Other information that is important includes the number of media stream the presentation contains, the transport protocols used for the media streams, and identifiers for these media streams. This information is required before setup of the content is possible and to determine if the client is even capable of using the content.

This information need not be sent using RTSP; other external protocols can be used to transmit the transport presentation descriptions. Two good examples are the use of HTTP [RFC2616] or email to fetch or receive presentation descriptions like SDP [RFC4566]

Setup: Set up some or all of the media streams in a presentation. The setup itself consist of selecting the protocol for media transport and the necessary parameters for the protocol, like addresses and ports.

Control of Transmission: After the necessary media streams have been established the client can request the server to start transmitting the content. The client must be allowed to start or stop the transmission of the content at arbitrary times. The client must also be able to start the transmission at any

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1568
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

point in the timeline of the presentation.

Synchronization: For media transport protocols like RTP [RFC3550] it might be beneficial to carry synchronization information within RTSP. This may be due to either the lack of inter-media synchronization within the protocol itself, or the potential delay before the synchronization is established (which is the case for RTP when using RTCP).

Termination: Terminate the established contexts.

For this use case there are a number of assumptions about how it works. These are:

On-Demand content: The content is stored at the server and can be accessed at any time during a time period when it is intended to be available.

Independent sessions: A server is capable of serving a number of clients simultaneously, including from the same piece of content at different points in that presentations time-line.

Unicast Transport: Content for each individual client is transmitted to them using unicast traffic.

It is also possible to redirect the media traffic to a different destination than that of the agent controlling the traffic. However, allowing this without appropriate mechanisms for checking that the destination approves of this allows for distributed denial of service attacks (DDoS).

E.2. Unicast Distribution of Live Content

This use case is similar to the above on-demand content case (see Appendix E.1) the difference is the nature of the content itself. Live content is continuously distributed as it becomes available from a source; i.e., the main difference from on-demand is that one starts distributing content before the end of it has become available to the server.

In many cases the consumer of live content is only interested in consuming what is actually happens "now"; i.e., very similar to broadcast TV. However, in this case it is assumed that there exist no broadcast or multicast channel to the users, and instead the server functions as a distribution node, sending the same content to multiple receivers, using unicast traffic between server and client. This unicast traffic and the transport parameters are individually negotiated for each receiving client.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1569
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Another aspect of live content is that it often has a very limited time of availability, as it is only is available for the duration of the event the content covers. An example of such a live content could be a music concert which lasts 2 hour and starts at a predetermined time. Thus there is need to announce when and for how long the live content is available.

In some cases, the server providing live content may be saving some or all of the content to allow clients to pause the stream and resume it from the paused point, or to "rewind" and play continuously from a point earlier than the live point. Hence, this use case does not necessarily exclude playing from other than the live point of the stream, playing with scales other than 1.0, etc.

E.3. On-demand Playback using Multicast

It is possible to use RTSP to request that media be delivered to a multicast group. The entity setting up the session (the controller) will then control when and what media is delivered to the group. This use case has some potential for denial of service attacks by flooding a multicast group. Therefore, a mechanism is needed to indicate that the group actually accepts the traffic from the RTSP server.

An open issue in this use case is how one ensures that all receivers listening to the multicast or broadcast receives the session presentation configuring the receivers. This specification has to rely on a external solution to solve this issue.

E.4. Inviting an RTSP server into a conference

If one has an established conference or group session, it is possible to have an RTSP server distribute media to the whole group. Transmission to the group is simplest when controlled by a single participant or leader of the conference. Shared control might be possible, but would require further investigation and possibly extensions.

This use case assumes that there exists either multicast or a conference focus that redistribute media to all participants.

This use case is intended to be able to handle the following scenario: A conference leader or participant (hereafter called the controller) has some pre-stored content on an RTSP server that he wants to share with the group. The controller sets up an RTSP session at the streaming server for this content and retrieves the session description for the content. The destination for the media content is set to the shared multicast group or conference focus.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1570
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

When desired by the controller, he/she can start and stop the transmission of the media to the conference group.

There are several issues with this use case that are not solved by this core specification for RTSP:

Denial of service: To avoid an RTSP server from being an unknowing participant in a denial of service attack the server needs to be able to verify the destination's acceptance of the media. Such a mechanism to verify the approval of received media does not yet exist; instead, only policies can be used, which can be made to work in controlled environments.

Distributing the presentation description to all participants in the group: To enable a media receiver to correctly decode the content the media configuration information needs to be distributed reliably to all participants. This will most likely require support from an external protocol.

Passing control of the session: If it is desired to pass control of the RTSP session between the participants, some support will be required by an external protocol to exchange state information and possibly floor control of who is controlling the RTSP session.

If there interest in this use case, further work is required on the necessary extensions.

E.5. Live Content using Multicast

This use case in its simplest form does not require any use of RTSP at all; this is what multicast conferences being announced with SAP [RFC2974] and SDP are intended to handle. However, in use cases where more advanced features like access control to the multicast session are desired, RTSP could be used for session establishment.

A client desiring to join a live multicasted media session with cryptographic (encryption) access control could use RTSP in the following way. The source of the session announces the session and gives all interested an RTSP URI. The client connects to the server and requests the presentation description, allowing configuration for reception of the media. In this step it is possible for the client to use secured transport and any desired level of authentication; for example, for billing or access control. An RTSP link also allows for load balancing between multiple servers.

If these were the only goals, they could be achieved by simply using HTTP. However, for cases where the sender likes to keep track of

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1571
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

each individual receiver of a session, and possibly use the session as a side channel for distributing key-updates or other information on a per-receiver basis, and the full set of receivers is not know prior to the session start, the state establishment that RTSP provides can be beneficial. In this case a client would establish an RTSP session for this multicast group with the RTSP server. The RTSP server will not transmit any media, but instead will point to the multicast group. The client and server will be able to keep the session alive for as long as the receiver participates in the session thus enabling, for example, the server to push updates to the client.

This use case will most likely not be able to be implemented without some extensions to the server-to-client push mechanism. Here the PLAY_NOTIFY method (see Section 13.5) with a suitable extension could provide clear benefits.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1572
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix F. Text format for Parameters

A resource of type "text/parameters" consists of either 1) a list of parameters (for a query) or 2) a list of parameters and associated values (for an response or setting of the parameter). Each entry of the list is a single line of text. Parameters are separated from values by a colon. The parameter name MUST only use US-ASCII visible characters while the values are UTF-8 text strings. The media type registration form is in Section 22.16.

There exist a potential interoperability issue for this format. It was named in RFC 2326 but never defined, even if used in examples that hint at the syntax. This format matches the purpose and its syntax supports the examples provided. However, it goes further by allowing UTF-8 in the value part, thus usage of UTF-8 strings may not be supported. However, as individual parameters are not defined, the using application anyway needs to have out-of-band agreement or using feature-tag to determine if the end-point supports the parameters.

The ABNF [RFC5234] grammar for "text/parameters" content is:

```
file              = *((parameter / parameter-value) CRLF)
parameter         = 1*visible-except-colon
parameter-value   = parameter *WSP ":" value
visible-except-colon = %x21-39 / %x3B-7E    ; VCHAR - ":"
value             = *(TEXT-UTF8char / WSP)
TEXT-UTF8char     =  %x21-7E / UTF8-NONASCII
UTF8-NONASCII     =  %xC0-DF 1UTF8-CONT
                  /  %xE0-EF 2UTF8-CONT
                  /  %xF0-F7 3UTF8-CONT
                  /  %xF8-FB 4UTF8-CONT
                  /  %xFC-FD 5UTF8-CONT
UTF8-CONT         =  %x80-BF
WSP               = <See RFC 5234> ; Space or HTAB
VCHAR             = <See RFC 5234>
CRLF              = <See RFC 5234>
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1573
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix G. Requirements for Unreliable Transport of RTSP

This section provides anyone intending to define how to transport of RTSP messages over a unreliable transport protocol with some information learned by the attempt in RFC 2326 [RFC2326]. RFC 2326 define both an URI scheme and some basic functionality for transport of RTSP messages over UDP, however, it was not sufficient for reliable usage and successful interoperability.

The RTSP scheme defined for unreliable transport of RTSP messages was "rtspu". It has been reserved by this specification as at least one commercial implementation exist, thus avoiding any collisions in the name space.

The following considerations should exist for operation of RTSP over an unreliable transport protocol:

- Request shall be acknowledged by the receiver. If there is no acknowledgement, the sender may resend the same message after a timeout of one round-trip time (RTT). Any retransmissions due to lack of acknowledgement must carry the same sequence number as the original request.

- The round-trip time can be estimated as in TCP (RFC 1123) [RFC1123], with an initial round-trip value of 500 ms. An implementation may cache the last RTT measurement as the initial value for future connections.

- If RTSP is used over a small-RTT LAN, standard procedures for optimizing initial TCP round trip estimates, such as those used in T/TCP (RFC 1644) [RFC1644], can be beneficial.

- The Timestamp header (Section 16.51) is used to avoid the retransmission ambiguity problem [Stevens98].

- The registered default port for RTSP over UDP for the server is 554.

- RTSP messages can be carried over any lower-layer transport protocol that is 8-bit clean.

- RTSP messages are vulnerable to bit errors and should not be subjected to them.

- Source authentication, or at least validation that RTSP messages comes from the same entity becomes extremely important, as session hijacking may be substantially easier for RTSP message transport using an unreliable protocol like UDP than for TCP.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1574
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

There exist two RTSP headers thats primarily are intended for being used by the unreliable handling of RTSP messages and which will be maintained:

- CSeq: See Section 16.19
- Timestamp: See Section 16.51

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1575
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix H. Backwards Compatibility Considerations

This section contains notes on issues about backwards compatibility with clients or servers being implemented according to RFC 2326 [RFC2326]. Note that there exists no requirement to implement RTSP 1.0, in fact we recommend against it as it is difficult to do in an interoperable way.

A server implementing RTSP/2.0 MUST include a RTSP-Version of RTSP/2.0 in all responses to requests containing RTSP-Version RTSP/2.0. If a server receives a RTSP/1.0 request, it MAY respond with a RTSP/1.0 response if it chooses to support RFC 2326. If the server chooses not to support RFC 2326, it MUST respond with a 505 (RTSP Version not supported) status code. A server MUST NOT respond to a RTSP-Version RTSP/1.0 request with a RTSP-Version RTSP/2.0 response.

Clients implementing RTSP/2.0 MAY use an OPTIONS request with a RTSP-Version of 2.0 to determine whether a server supports RTSP/2.0. If the server responds with either a RTSP-Version of 1.0 or a status code of 505 (RTSP Version not supported), the client will have to use RTSP/1.0 requests if it chooses to support RFC 2326.

H.1. Play Request in Play State

The behavior in the server when a Play is received in Play state has changed (Section 13.4). In RFC 2326, the new PLAY request would be queued until the current Play completed. Any new PLAY request now take effect immediately replacing the previous request.

H.2. Using Persistent Connections

Some server implementations of RFC 2326 maintain a one-to-one relationship between a connection and an RTSP session. Such implementations require clients to use a persistent connection to communicate with the server and when a client closes its connection, the server may remove the RTSP session. This is worth noting if a RTSP 2.0 client also supporting 1.0 connects to a 1.0 server.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1576
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix I. Open Issues

Open issues are filed and tracked in the bug and feature trackers at http://rtspspec.sourceforge.net. Open issues are discussed on MMUSIC list (mmusic@ietf.org).

Note to RFC-editor: Please remove this section before publication of this document as an RFC.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1577
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix J. Changes

This appendix briefly lists the differences between RTSP 1.0 [RFC2326] and RTSP 2.0 for an informational purpose. For implementers of RTSP 2.0 it is recommended to read carefully through this memo and not to rely on the list of changes below to adapt from RTSP 1.0 to RTSP 2.0, as RTSP 2.0 is not intended to be backwards compatible with RTSP 1.0 [RFC2326] other than the version negotiation mechanism.

J.1. Brief Overview

The following protocol elements were removed in RTSP 2.0 compared to RTSP 1.0:

- there is no section on minimal implementation anymore, but more the definition of RTSP 2.0 core;
- the RECORD and ANNOUNCE methods and all related functionality (including 201 (Created) and 250 (Low On Storage Space) status codes);
- the use of UDP for RTSP message transport was removed due to missing interest and to broken specification;
- the use of PLAY method for keep-alive in Play state.

The following protocol elements were added or changed in RTSP 2.0 compared to RTSP 1.0:

- RTSP session TEARDOWN from the server to the client;
- IPv6 support;
- extended IANA registries (e.g., transport headers parameters, transport-protocol, profile, lower-transport, and mode);
- request pipelining for quick session start-up;
- fully reworked state-machine;
- RTSP messages now uses URIs rather then URLs;
- incorporated much of related HTTP text ([RFC2616]) in this memo, compared to just referencing the sections in HTTP, to avoid ambiguities;

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1578
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

- the REDIRECT method was expanded and diversified for different situations;

- Includes a new section about how to setup different media transport alternatives and their profiles, and lower layer protocols. This resulted that the appendix on RTP interaction was moved there instead in the part describing RTP. The section also includes guidelines what to consider when writing usage guidelines for new protocols and profiles;

- added an asynchronous notification method PLAY_NOTIFY. This method is used by the RTSP server to asynchronously notify clients about session changes while in Play state. To a limited extend this is comparable with some implementations of ANNOUNCE in RTSP 1.0 not intended for Recording.

J.2. Detailed List of Changes

Compared to RTSP 1.0 (RFC 2326), the below changes has been made when defining RTSP 2.0. Note that this list does not reflect minor changes in wording or correction of typographical errors.

- The section on minimal implementation was deleted without substitution.

- The Transport header has been changed in the following way:

  * The ABNF has been changed to define that extensions are possible, and that unknown parameters results in that servers ignore the transport specification.

  * To prevent backwards compatibility issues, any extension or new parameter requires the usage of a feature-tag combined with the Require header.

  * Syntax unclarities with the Mode parameter has been resolved.

  * Syntax error with ";" for multicast and unicast has been resolved.

  * Two new addressing parameters has been defined, src_addr and dest_addr. These replaces the parameters "port", "client_port", "server_port", "destination", "source".

  * Support for IPv6 explicit addresses in all address fields has been included.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1579
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

  * To handle URI definitions that contain ";" or "," a quoted URI format has been introduced and is required.

  * Defined IANA registries for the transport headers parameters, transport-protocol, profile, lower-transport, and mode.

  * The transport headers interleaved parameter's text was made more strict and use formal requirements levels. It was also clarified that the interleaved channels are symmetric and that it is the server that sets the channel numbers.

  * It has been clarified that the client can't request of the server to use a certain RTP SSRC, using a request with the transport parameter SSRC.

  * Syntax definition for SSRC has been clarified to require 8HEX. It has also been extended to allow multiple values for clients supporting this version.

  * Clarified the text on the transport headers "dest_addr" parameters regarding what security precautions the server is required to perform.

- The Range formats has been changed in the following way:

  * The NPT format has been given a initial NPT identifier that must now be used.

  * All formats now support initial open ended formats of type "npt=-10" and also format only "Range: smpte" ranges for usage with GET_PARAMETER requests.

- RTSP message handling has been changed in the following way:

  * RTSP messages now uses URIs rather then URLs.

  * It has been clarified that a 4xx message due to missing CSeq header shall be returned without a CSeq header.

  * The 300 (Multiple Choices) response code has been removed.

  * Rules for how to handle timing out RTSP messages has been added.

  * Extended Pipelining rules allowing for quick session startup.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1580
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

- The HTTP references has been updated to RFC 2616 and RFC 2617. Most of text has been copied and then altered to fit RTSP into this specification. Public, and the Content-Base header has also been imported from RFC 2068 so that they are defined in the RTSP specification. Known effects on RTSP due to HTTP clarifications:
  - Content-Encoding header can include encoding of type "identity".
- The state machine section has completely been rewritten. It includes now more details and are also more clear about the model used.
- A IANA section has been included with contains a number of registries and their rules. This will allow us to use IANA to keep track of RTSP extensions.
- The transport of RTSP messages has seen the following changes:
  - The use of UDP for RTSP message transport has been deprecated due to missing interest and to broken specification.
  - The rules for how TCP connections is to be handled has been clarified. Now it is made clear that servers should not close the TCP connection unless they have been unused for significant time.
  - Strong recommendations why server and clients should use persistent connections has also been added.
  - There is now a requirement on the servers to handle non-persistent connections as this provides fault tolerance.
  - Added wording on the usage of Connection:Close for RTSP.
  - specified usage of TLS for RTSP messages, including a scheme to approve a proxies TLS connection to the next hop.
- The following header related changes have been made:
  - Accept-Ranges response header is added. This header clarifies which range formats that can be used for a resource.
  - Fixed the missing definitions for the Cache-Control header. Also added to the syntax definition the missing delta-seconds for max-stale and min-fresh parameters.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1581
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

* Put requirement on CSeq header that the value is increased by one for each new RTSP request. A Recommendation to start at 0 has also been added.

* Added requirement that the Date header must be used for all messages with message body and the Server should always include it.

* Removed possibility of using Range header with Scale header to indicate when it is to be activated, since it can't work as defined. Also added rule that lack of Scale header in response indicates lack of support for the header. Feature-tags for scaled playback has been defined.

* The Speed header must now be responded to indicate support and the actual speed going to be used. A feature-tag is defined. Notes on congestion control was also added.

* The Supported header was borrowed from SIP [RFC3261] to help with the feature negotiation in RTSP.

* Clarified that the Timestamp header can be used to resolve retransmission ambiguities.

* The Session header text has been expanded with a explanation on keep alive and which methods to use. SET_PARAMETER is now recommended to use if only keep-alive within RTSP is desired.

* It has been clarified how the Range header formats is used to indicate pause points in the PAUSE response.

* Clarified that RTP-Info URIs that are relative, use the Request-URI as base URI. Also clarified that the used URI must be the one that was used in the SETUP request. The URIs are now also required to be quoted. The header also expresses the SSRC for the provided RTP timestamp and sequence number values.

* Added text that requires the Range to always be present in PLAY responses. Clarified what should be sent in case of live streams.

* The headers table has been updated using a structured borrowed from SIP. Those tables carries much more information and should provide a good overview of the available headers.

* It has been clarified that any message with a message body is required to have a Content-Length header. This was the case in RFC 2326, but could be misinterpreted.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1582
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

  * ETag has changed name to MTag.

  * To resolve functionality around MTag. The MTag and If-None-Match header has been added from HTTP with necessary clarification in regards to RTSP operation.

  * Imported the Public header from HTTP RFC 2068 [RFC2068] since it has been removed from HTTP due to lack of use. Public is used quite frequently in RTSP.

  * Clarified rules for populating the Public header so that it is an intersection of the capabilities of all the RTSP agents in a chain.

  * Added the Media-Range header for listing the current availability of the media range.

  * Added the Notify-Reason header for giving the reason when sending PLAY_NOTIFY requests.

  * A new header Seek-Style has been defined to direct and inform how any seek operation should/have been performed.

- The Protocol Syntax has been changed in the following way:

  * All ABNF definitions are updated according to the rules defined in RFC 5234 [RFC5234] and has been gathered in a separate Section 20.

  * The ABNF for the User-Agent and Server headers has been corrected.

  * Some definitions in the introduction regarding the RTSP session has been changed.

  * The protocol has been made fully IPv6 capable.

  * Added a fragment part to the RTSP URI. This seem to be indicated by the note below the definition, however, it was not part of the ABNF.

  * The CHAR rule has been changed to exclude NULL.

- The Status codes have been changed in the following way:

  * The use of status code 303 "See Other" has been deprecated as it does not make sense to use in RTSP.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1583
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

  * When sending response 451 and 458 the response body should contain the offending parameters.

  * Clarification on when a 3rr redirect status code can be received has been added. This includes receiving 3rr as a result of request within a established session. This provides clarification to a previous unspecified behavior.

  * Removed the 201 (Created) and 250 (Low On Storage Space) status codes as they are only relevant to recording, which is deprecated.

  * Several new Status codes has been defined: 464 "Data Transport Not Ready Yet", 465 "Notification Reason Unknown", 470 "Connection Authorization Required", 471 "Connection Credentials not accepted", 472 "Failure to establish secure connection".

- The following functionality has been deprecated from the protocol:

  * The use of Queued Play.

  * The use of PLAY method for keep-alive in Play state.

  * The RECORD and ANNOUNCE methods and all related functionality. Some of the syntax has been removed.

  * The possibility to use timed execution of methods with the time parameter in the Range header.

  * The description on how rtspu works is not part of the core specification and will require external description. Only that it exist is defined here and some requirements for the transport is provided.

- The following changes has been made in relation to methods:

  * The OPTIONS method has been clarified with regards to the use of the Public and Allow headers.

  * Added text clarifying the usage of SET_PARAMETER for keep-alive and usage without any body.

  * PLAY method is now allowed to be pipelined with the pipelining of one or more SETUP requests following the initial that generates the session for aggregated control.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1584
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

    * REDIRECT has been expanded and diversified for different situations.

    * Added a new method PLAY_NOTIFY. This method is used by the RTSP server to asynchronously notify clients about session changes.

- Wrote a new section about how to setup different media transport alternatives and their profiles, and lower layer protocols. This resulted that the appendix on RTP interaction was moved there instead in the part describing RTP. The section also includes guidelines what to consider when writing usage guidelines for new protocols and profiles.

- Setup and usage of independent TCP connections for transport of RTP has been specified.

- Added a new section describing the available mechanisms to determine if functionality is supported, called "Capability Handling". Renamed option-tags to feature-tags.

- Added a contributors section with people who have contributed actual text to the specification.

- Added a section Use Cases that describes the major use cases for RTSP.

- Clarified the usage of a=range and how to indicate live content that are not seekable with this header.

- Text specifying the special behavior of PLAY for live content.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1585
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix K. Acknowledgements

This memorandum defines RTSP version 2.0 which is a revision of the Proposed Standard RTSP version 1.0 which is defined in [RFC2326]. The authors of RFC 2326 are Henning Schulzrinne, Anup Rao, and Robert Lanphier.

Both RTSP version 1.0 and RTSP version 2.0 borrow format and descriptions from HTTP/1.1.

This document has benefited greatly from the comments of all those participating in the MMUSIC-WG. In addition to those already mentioned, the following individuals have contributed to this specification:

Rahul Agarwal, Jeff Ayars, Milko Boic, Torsten Braun, Brent Browning, Bruce Butterfield, Steve Casner, Francisco Cortes, Kelly Djahandari, Martin Dunsmuir, Eric Fleischman, Jay Geagan, Andy Grignon, V. Guruprasad, Peter Haight, Mark Handley, Brad Hefta-Gaub, Volker Hilt, John K. Ho, Go Hori, Philipp Hoschka, Anne Jones, Ingemar Johansson, Anders Klemets, Ruth Lang, Stephanie Leif, Jonathan Lennox, Eduardo F. Llach, Thomas Marshall, Rob McCool, David Oran, Joerg Ott, Maria Papadopouli, Sujal Patel, Ema Patki, Alagu Periyannan, Colin Perkins, Igor Plotnikov, Jonathan Sergent, Pinaki Shah, David Singer, Lior Sion, Jeff Smith, Alexander Sokolsky, Dale Stammen, John Francis Stracke, Maureen Chesire, David Walker, Geetha Srikantan, Stephan Wenger, Pekka Pessi, Jae-Hwan Kim, Holger Schmidt, Stephen Farrell, Xavier Marjou, Joe Pallas, Martti Mela, Byungjo Yoon and Patrick Hoffman, Jinhang Choi.

K.1. Contributors

The following people have made written contributions that were included in the specification:

- Tom Marshall contributed text on the usage of 3rr status codes.
- Thomas Zheng contributed text on the usage of the Range in PLAY responses and proposed an earlier version of the PLAY_NOTIFY method.
- Sean Sheedy contributed text on the timeout behavior of RTSP messages and connections, the 463 status code, and proposed an earlier version of the PLAY_NOTIFY method.
- Greg Sherwood proposed an earlier version of the PLAY_NOTIFY method.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1586
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

- Fredrik Lindholm contributed text about the RTSP security framework.
- John Lazzaro contributed the text for RTP over Independent TCP.
- Aravind Narasimhan contributed by rewriting Media Transport Alternatives (Appendix C) and editorial improvements on a number of places in the specification.
- Torbjorn Einarsson has done some editorial approvements of the text.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1587
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Appendix L. RFC Editor Consideration

Please replace RFC XXXX with the RFC number this specification receives.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1588
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Authors' Addresses

Henning Schulzrinne
Columbia University
1214 Amsterdam Avenue
New York, NY 10027
USA

Email: schulzrinne@cs.columbia.edu

Anup Rao
Cisco
USA

Email: anrao@cisco.com

Rob Lanphier
Seattle, WA
USA

Email: robla@robla.net

Magnus Westerlund
Ericsson AB
Faeroegatan 6
STOCKHOLM, SE-164 80
SWEDEN

Email: magnus.westerlund@ericsson.com

Martin Stiemerling
NEC Laboratories Europe, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 (0) 6221 4342 113
Email: martin.stiemerling@neclab.eu
URI: http://ietf.stiemerling.org

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1589
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

TAXAN

Online Shop | Dealer Site | Forum | Sitemap | Contact

USA

Products

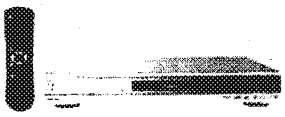
- Home
- Corporate
- Products
- News & Events
- Support
- Contact
- Links
- Japanese Customers

PRINTER FRIENDLY PAGE

Home > Products > Home Entertainment > AVeL LinkPlayer2 > AVeL LinkPlayer2 for Consumer

AVeL LinkPlayer2 for Consumer

AVLP2/DVDLA



BCN AWARD 2004-2006 AVeL LinkPlayer Japan's No.1 DMA

View

Available Stock Only

The AVLP2/DVDLA, a high def capable DVD player, is designed for the affordable home theater system. You can enjoy commertial DVD-Videos also HDNet® DVD titles on your big screen. The front USB port enables to play from your portable player or HDD, and the ethernet connects to your computers.

The LinkPlayer can play back Windows Media Video® 9 (WMV9), DivX® HD, and MPEG2-TS (HD 720p/1080i). HD mode allows you to enjoy pictures and movies in High-Definition (Video Up to 1080i / Photo Up to 2048x1532).

MANUAL Download PDF Manual (4538 Kb)

Download PDF Catalog (1490 Kb)

FAQ View FAQ

GALLERY View Gallery

DOWNLOAD Download available drivers

Featured Users

- High Def Home Theater
- Network Ditigal Home
- Remote View Anywhere
- DivX® Applications

User's Voice

- Review / Article
- Forum / Compare

General Information

Update Info 11/20/2006
- Setup Guide / Online
- Firmware Update **NEW**
- Beta Software Download
- Tips / Hints

Where to Buy
- USA / Canada

Latest Firmware Version
20-50-061026-04-IOD-234



Copyright © 2003 I-O DATA DEVICE USA, INC. All Rights Reserved.

Windows Media Center Extender for Xbox

Original Xbox: System Use

Console

View the Dashboard on HDTV

Listen in Surround Sound

Extend Your Warranty

Windows Media Center Extender for Xbox

Troubleshooting Your Hardware

Troubleshooting Network Connectivity

Power Cord Replacement

Xbox Live Features

Account Management

Backward Compatibility

Resources

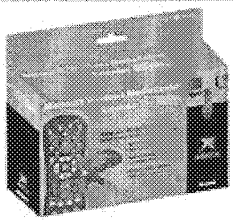
Related Links

- Set Up Your System
- Guide to Home Networking
- Join Xbox Live

Product Information

Xbox Windows Media Center Extender

Manufacturer:
Microsoft



Category:
Accessories

Console:
Xbox

$79.99

Related Links

- Hardware Manuals
- Product Web Site
- Xbox Support
- Xbox Wireless Network

Windows Media Center Extender for Xb

Bring your Media Center PC experience to your Xbox® and turn it ultimate entertainment center with the Windows® Media Center E Xbox! Enjoy digital music, photos, videos, and TV shows on televi throughout your home.

With a wired or wireless connection to the Media Center PC (sold the Xbox console now allows you to enjoy the digital entertainme your PC when and how you want. The included remote control an also support DVD movie playback. Just grab the remote, drop in t Center Extender DVD in your Xbox, and get ready for an entertair revolution! A whole new world for your Xbox awaits ...

- Watch and record television shows.
- Enjoy a free integrated TV Program Guide with no fees.
- Access your Media Center PC's digital media library music, vid pictures.
- Stay connected with instant messaging and Xbox Live®.
- Watch DVD movies.
- Listen to FM and Internet radio.

Entertain Yourself
Turn your Xbox into the ultimate entertainment center. Enjoy all t music, photos, videos and TV shows stored on your Media Center convenience of your Xbox console. Extend the experience through add-on services that offer music and video on demand, news, and information.

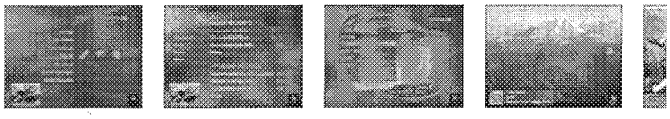
Take It Anywhere in the Home
Use the built-in Ethernet network support of your Xbox or optiona wireless hardware (sold separately) to bring the Media Center exp televisions all over your house. Just connect the Xbox console to t network, run the set-up software on the host Media Center PC, gr control, and you're in business.

Keep in Touch
Never miss an Xbox game invitation, thanks to built-in XboxLive A support. Send and receive Xbox *Live*game invitations while watch listening to Internet radio. Keep in touch with family with instant Windows® Messenger.

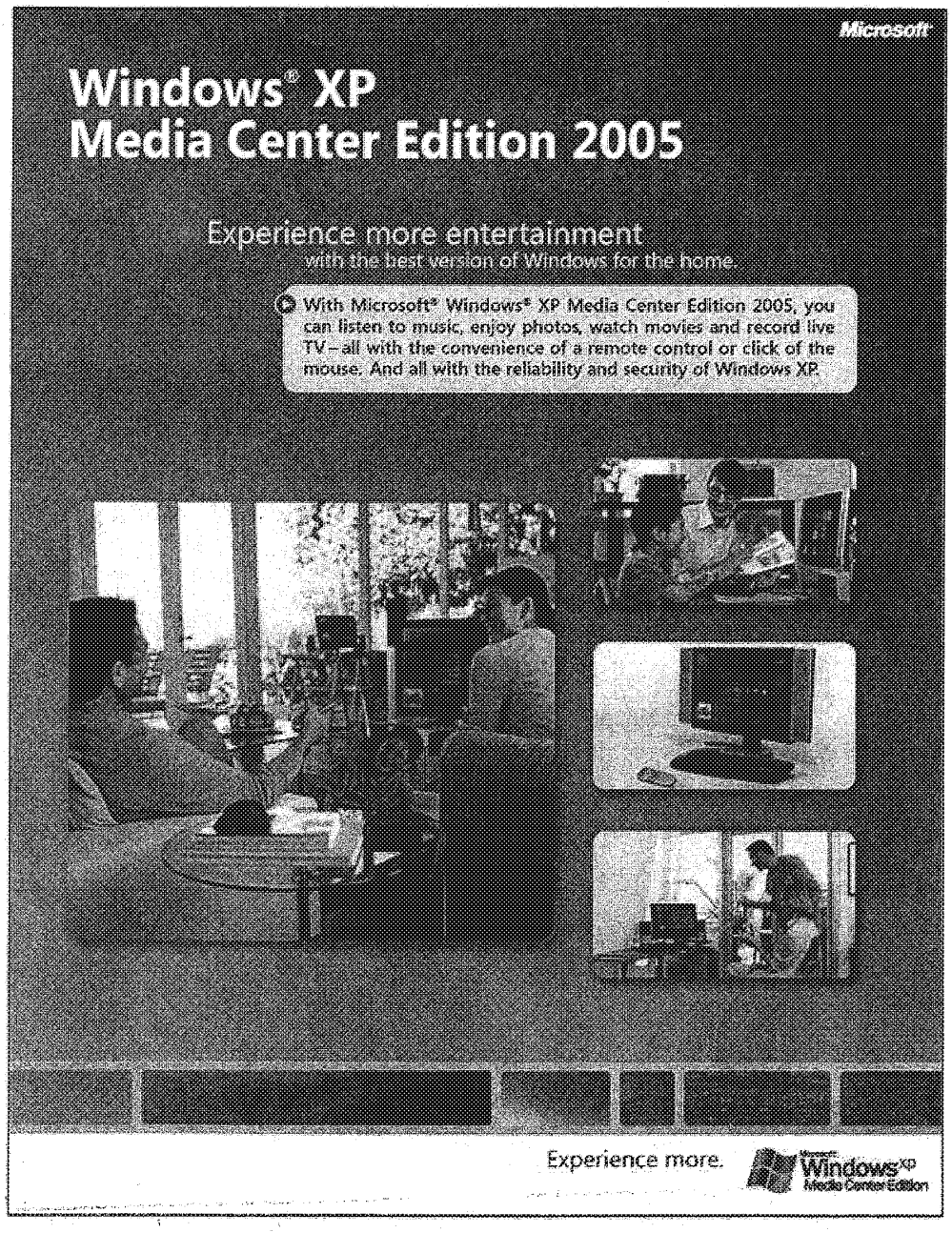
Enjoy DVD Movie Playback
The included remote control also supports DVD movie playback th Xbox console. Enjoy a night at the movies ... through your Xbox!

Screenshots

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1591
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1592
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)



http://download.microsoft.com/download/c/9/a/c9a7000a-66b3-455b-860b-1c16f2eecfec/MCE.pdf

Windows® XP Media Center Edition 2005 Microsoft

Experience more entertainment.

Windows XP Media Center Edition 2005.

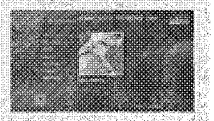
The Best Windows Ever Designed for the Home. Experience more of the photos, music, television, movies and games you want throughout your home with keyboard and mouse or a remote control.*

Complete and Easy Entertainment. Enjoy all of your digital entertainment all in one place.

More Connected. The widest range of ways to experience your digital entertainment around the home and on the go with Media Center Extenders, Portable Media Centers and more.

Do More With A Media Center PC.

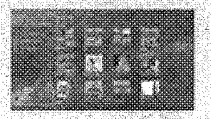
More Music. Listen to music in a whole new way—your way. Enjoy the ultimate digital jukebox and access online music services. Easily transfer music to a wide range of devices. Quickly and easily burn your favorites to CD.



More TV and Movies. Pause and record live TV.* Back up your favorite episodes to DVD. Easily find movies to record and enjoy with the new Movie Finder. Watch DVD movies and access movies-on-demand from online services.



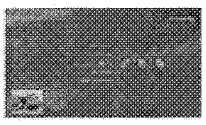
More Memories. A Media Center PC makes it easy to store and share your digital memories. Use the built-in tools to create photo albums, slide shows and home videos. Then share your creations on CD, DVD and the Web for everyone to enjoy.



Key Features.

- All-in-one organization of your photos, music, TV and movies
- Watch, pause and record live TV (and it's ready for HDTV* when you are)
- Burn photos, music, videos and TV to DVD
- Organize, view, enhance and share your digital photos
- Simple operation with keyboard and mouse or optional remote control*
- Record multiple live TV shows at the same time with your existing cable or satellite setup*
- Built-in Electronic Programming Guide** and Movie Finder with no monthly fees
- Windows Media Player 10
- Windows Movie Maker 2.1 with DVD burning
- Enjoy all your digital entertainment in any room in the home with Media Center Extender devices
- Full features and capabilities you expect with Windows XP





Windows XP Media Center Edition 2005 is available pre-installed on new PCs. Product availability may vary by geography and manufacturer.

Availability date: 10/12/2004
Distribution channel: Retail, Online
Geographical scope: Major Markets Worldwide
Product Web site: www.microsoft.com/windowsxp/mediacenter

* TV and DVD burning features vary by computer manufacturer.
** Note: To enjoy all the benefits of Windows XP Internet access is recommended for some features, such as the Electronic Programming Guide, Movie Finder and Online Spotlight, if it is required.

©2004 Microsoft Corporation. All rights reserved. Microsoft, Windows and the Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Experience more. Microsoft Windows XP Media Center Edition

Adaptive Streaming Comparison

Published on January 28, 2010 by chriskno
Updated on November 18, 2010 by chriskno
Tags: Adaptive Adobe Apple Dynamic Streaming Flash HTTP IIS Microsoft Smooth Streaming

IIS Smooth Streaming (http://www.iis.net/SmoothStreaming) provides users with a truly compelling way to watch the best-possible-quality streaming video. First proven in production at the Summer 2008 Olympics in Beijing, this live and on-demand adaptive streaming technology from Microsoft represents the latest generation in video streaming technologies.

Part of IIS Media Services (http://www.iis.net/media), IIS Smooth Streaming delivers up to true HD (1080p) video experiences that increase user engagement times. With integrated media delivery in IIS (http://www.iis.net/overview), you no longer need separate Web and streaming infrastructures to deliver compelling Web and rich media content to three screens (computer, TV, mobile). Using a single platform saves the cost of managing separate server infrastructures, and reduces the time to market and cost for deploying new applications, Web sites, and rich media experiences.

IIS Smooth Streaming is based on standard protocols, scales out using existing HTTP delivery and caching infrastructures, and ensures simplified file management by using as few as one contiguous video file to deliver multiple video quality levels. Combined with the Silverlight Media Framework (http://smf.codeplex.com/) and the upcoming Smooth Streaming client porting kit, IIS Smooth Streaming provides built-in sophisticated stream-switching, full DVR controls, and support for multiple camera angles. In addition, IIS Smooth Streaming 4.0 will also provide an option for streaming to iPhone (http://www.iis.net/overview/IntegratedMediaPlatform) and QuickTime X players.

The industry is excited by Microsoft's commitment to standards-based HTTP media delivery, and recent announcements from our partners and competitors are a validation of the Smooth Streaming approach to adaptive streaming. For example, as IIS Smooth Streaming has gained momentum, there have been new announcements from Adobe and Apple about methods each company now supports for varying the quality levels during video streaming. Adobe Flash Dynamic Streaming is a traditional multiple-bit-rate (MBR) streaming technology that is very similar to RealNetworks SureStream (released in 1998) and Windows Media Intelligent Streaming (released in 1999), and requires proprietary streaming servers to scale out to a large audience. Apple HTTP Adaptive Bitrate Streaming provides rudimentary stream-switching capabilities to devices running current Apple operating systems, and requires pre-segmenting your content into hundreds or thousands of small file chunks.

In the table below, we compare the latest adaptive streaming technologies from Microsoft, Adobe, and Apple.

	Microsoft IIS Smooth Streaming	**Adobe Flash Dynamic Streaming**	**Apple HTTP Adaptive Bitrate Streaming**
	Free download[1] (http://learn.iis.net/Con	$4500[3] (http://learn.iis.net/	Cost of Web server[5] (http://learn.iis.net/

Cost per Streaming Server	trolPanel/themes/iis/#1) + $469 for OS[2] (http://learn.iis.net/ControlPanel/themes/iis/#2)	ControlPanel/themes/iis/#3) + OS cost[4] (http://learn.iis.net/ControlPanel/themes/iis/#4)	ControlPanel/themes/iis/#5) + segmenter OS[6] (http://learn.iis.net/ControlPanel/themes/iis/#6)
On-demand & Live Streaming	✓	✓	✓
Live Streaming DVR[7] (http://learn.iis.net/page.aspx/792/adaptive-streaming-comparison/#7)	✓	Pause & Seek	
Streaming Protocol	HTTP	RTMP	HTTP
Scalability via HTTP Edge Caches	✓		✓
Stateless Server Connection[8] (http://learn.iis.net/page.aspx/792/adaptive-streaming-comparison/#8)	✓		✓
Supported Platforms	Silverlight, Xbox 360, other Smooth Streaming-compatible players, and iPhone OS 3.0[9] (http://learn.iis.net/page.aspx/792/adaptive-streaming-comparison/#9)	Flash Player 10, AIR	iPhone OS 3.0, devices running QuickTime X
DRM Support for Live, VOD	PlayReady	None[10] (http://learn.iis.net/page.aspx/792/adaptive-streaming-comparison/#10)	None
DRM ASP & Encoder ISV Support	✓		
Interoperable DRM (DECE Approved)[11] (http://learn.iis.net/page.aspx/792/adaptive-streaming-comparison/#11)	✓		

Real-time Client and Server Logging	✓[12] (http://learn.iis.net/page.aspx/792/adaptive-streaming-comparison/#12)		
Programmable Client Side Switching Logic	✓	✓	
Live In-Stream Ad Integration	✓		
Built-in Analytics Framework	✓		
Delivery to Mobile Devices	✓		✓
Native 64-bit Server Support	✓		✓
Media Container	MPEG 4 – Part 12 (Fragmented MP4)	MPEG 4 – Part 12 (MP4), FLV	MPEG-2 TS
Supported Video Codecs	Codec Agnostic (currently supports VC-1 Advanced Profile & H.264 Baseline, Main, and High)	H.264 Baseline, Main, and High; VP6	H.264 Baseline Level 3.0
Supported Audio Codecs	Codec Agnostic (currently supports WMA & AAC)	AAC, MP3	MP3, HE-AAC, AAC-LC
Maximum Bit Rate	No limit	No limit	1.6 Mbps
Default Fragment Length	2 seconds	n/a	10 seconds
End-To-End Latency	As low as 1.5 seconds (configurable)	6 seconds	30 seconds[13] (http://learn.iis.net/page.aspx/792/adaptive-streaming-comparison/#13)
File Type on Server	Contiguous	Contiguous	Fragmented
Client Programming Platform	Microsoft .NET Framework	Adobe ActionScript	Objective-C

Footnotes...	
	IIS Smooth Streaming is part of the free

()1	IIS Media Services (http://www.iis.net/media) download for Windows Server 2008 and Windows Server 2008 R2 .
()2	Runs on any edition of Windows Server® 2008 or Windows Server 2008 R2 including Windows® Web Server, which has a list price of $469.
()3	Assumes use of Adobe Flash Media Interactive Server to support Pause, Seek, Authentication, and higher scalability
()4	Requires Windows Server 2003 SP2, Windows Server 2008, Red Hat® Enterprise Linux® 4, or Red Hat Enterprise Linux 5.2.
()5	Runs on any Web server. Also requires the Apple streaming segmenter – see next note.
()6	The Apple stream segmenter is a utility that receives encoded MPEG2-TS and breaks it into 10 second "chunks" for delivery. This free download requires an Intel-based Mac, with a Mac Pro or an XServe having two Ethernet network interfaces recommended.
()7	Full DVR features include Pause, Seek, Fast Forward (e.g., 2x, 5x playback speeds), Fast Rewind, Go To Live, Instant Replay, and Slow Motion.
()8	A stateless (non-persistent) connection between server and client increases scalability and allows seamless failover or rollover between load-balanced servers.
()9	Support for adaptive streaming to the Apple iPhone from IIS Media Services 4 was announced (http://blogs.iis.net/chriskno/archive/2009/12/01/faqs-on-using-iis-smooth-streaming-with-the-apple-iphone.aspx) on October 18th, 2009.
()10	Future support was announced (http://www.adobe.com/products/flashmediastreaming/systemreqs/) Sept. 10, 2009. Proposed availability is H1 CY10 for delivery to a future version of Adobe® Flash® Player and Adobe AIR™.
	Digital Entertainment Content Ecosystem (DECE, LLC) is a consortium of major Hollywood studios, consumer electronics manufacturers and retailers, network

()11	hardware vendors, systems integrators and Digital Rights Management (DRM) vendors chartered to develop a set of standards for the digital distribution of premium Hollywood content.
()12	Real-time logging for Silverlight applications using the IIS Advanced Logging (http://www.iis.net/AdvancedLogging) extension.
()13	Encoded live streams are processed by the Apple stream segmenter. This intermediate step results in increased latency for live delivery.

Article Category

Products / Services Discussed

Tags

Best Practices For Multi-Device Transcoding

View current | Track | Permalink

Kaltura Edition:
Kaltura SaaS
Kaltura Cloud Edition
Kaltura Community Edition
Kaltura On Prem

First Supported Kaltura Version:
Kaltura Dragonfly

Audience / Tech Expertise:
Configuration
Content Administration
IT, Security and Server Administration

Introduction

Today video is everywhere. Every site. Every device. Everywhere you turn video is streaming. Consumer eyes are better trained than ever, expecting higher and higher quality with faster and faster service.

Content is king and no one wants their content interrupted because of video buffering, skipped frames or the dreaded crashed player.

The following document details the current Best Practices for Multi-Device Transcoding. Each section will detail the recommended encoding settings and resolutions for the current array of devices along with use cases and background information.

IT'S ALL ABOUT BALANCE

The challenge of streaming video is to find the right balance between bitrate and resolution as it relates to an end user's connection speed and system ability. There are far too many variable sthat could interrupt playback for us to account for them all, at least as of the time of this writing. Here is a short list of potential stream killing variables:

- Device and Screen size
- Internet Service Provider Connection Speed or Cap
- Bandwidth available through the wireless router due to distance or usage
- Network traffic to and from the sever hosting the video
- The CPU and GPU abilities on the playing device
- Browser brand and version
- Available plugins such as Flash and Silverlight
- Other programs running in the background

These are just the more obvious variables to consider and though we can not control them all we can take steps to avoid or account of most of them.

Streaming Content

Content flows to an end users system in a stream of data much like a river might flow from a mountain reservoir to a small town. All content that is viewed on the internet inside an application is considered streaming content. This document focuses on streaming video.

Goals Of Streaming Video

- Immediate Playback Start
- Uninterrupted Playback (no "buffering")
- Smooth Playback (no frame drops)
- Highest quality possible

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1600
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Adaptive Streaming

Adaptive streaming is a technique of detecting a users bandwidth capabilities in real time and then adjusting the quality of the video stream accordingly. This results in less buffering, fast start times and overall better experience for both high-speed and low-speed connections. Adaptive streaming works by having multiple bit rates available that the player or server (or CDN) can pull based on the users connection speed and ability. To exemplify this let's pretend we're streaming a movie called "The Big Movie". Though the end user would see a smoothly playing video, unknown to them multiple streams are actually available and may be seamlessly switched to if their connection drops lower or improves. The key here is "seamless". When adaptive is done correctly there should be no interruption of playback.

So for The Big Movie a player might be serving up the following all at once:

- The Big Movie @ 2612 kbps
- The Big Movie @ 1600 kbps
- The Big Movie @ 1200 kbps
- The Big Movie @ 800 kbps

USE CASE:

If a user's connection dropped from over 2.5 Mbps to say 1600 kbps the video player would switch, again seamlessly, from the 2612 bit rate down to the 1600. Often the player is designed to start on the lowest possible bit rate so that playback starts immediately. So for this example the 800 kbps stream might be served up first and then as the player realizes that the user can handle a higher bitrate it would then switch up.

Adaptive Set

An Adaptive Set is a package of transcodes for the same video that span multiple bit rates and are meant to find a balance between connection speed and resolution.

In order for Adaptive Streaming to provide the optimal viewing experience, all the streams in the Adaptive Set must be in some alignment. Typically, for Desktop and Set Box applications, this means that the frame rates, key frame intervals (GOP size), audio sample rates, and so on should be the same within a set. This is so that as the player switches between bit rates a smooth, seamless switch is achieved without any buffering, stuttering or noticeable audio pops.

This is not followed, however, when looking at adaptive sets for mobile. When users are viewing content on a mobile device they may be moving in between wireless or cell zones and their signal strength may fluctuate widely. Though we will get into this in more detail in the mobile section generally because of this variable you want your bit rates to not only span possible mobile connections but also be optimized for maintaining the stream -- this may mean hard shifts down in bit rates so that the end user does not have their mobile player crash and disengage them from your content and business. Because of this as bit rates go lower the resolution also decreases, as does the frame rate. Frame rates for mobile may go as low as 8 fps. Again, we will get into this in more detail in the Mobile Section.

Every device has different requirements currently for what they ideally want in an Adaptive Set. This can become overwhelming to video producers, editors and managers as it does, in reality, mean you need many versions of your video files for one video to be playable across multiple devices. This list could go as high as 40 different transcodes that need to take place however do not panic, you do not need to use all 40 if your budget, storage considerations and time do not permit it. You can compromise, but understand that those variables we discussed in the first section of this document may become a factor and cause poor playback. In a later section we will discuss specific device groups and the settings and use cases applicable to each.

General Settings And Concepts

Before we can dive into the specific device groups and their intended settings it is important to understand the essentials of transcoding. This section will detail the most commonly used settings in this process ranging from GOP size (or I-Frame interval) to resolutions and frame rates and so on.

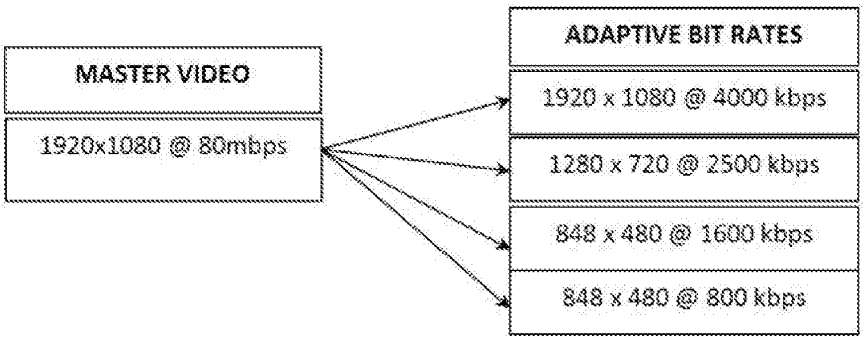
Transcoding

Let's start with that word "transcoding". A correctly formatted video master will NOT stream on the internet and must be converted, or transcoded, in order to be viewed. A transcode is made from taking an encoded piece of video and then converting it into one or more newly and more compressed streams that can then be played in a player on a computer or mobile device depending on the settings and methods used.

USE CASE:

A user wishes to take a piece of video shot in 1920 x 1080 and make it playable in an adaptive player on the internet. In order to do this the Master Video, which is of a high quality at 1080 in resolution, must be converted to at least 3 or more different bitrates that may even scale down in resolution.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1601
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)



Note that the 1920 x 1080 master video which was created at 80 mbps is converted to four different and more compressed bit rates. Though the first on the list is a 1920x1080 resolution the bit rate is only 4000 kbps, much lower than the 80,000 kbps (or 80 Mbps) stream. The next three bitrates that are produced scale down from the 1080 resolution to 720 and then 480 -- making this video playable for a wide range of users and connections.

Transcoding Methods

Different methods can be used to transcode video that trade quality for time. An Encoder can choose between at least 3 different encoding methods that effect bit rate and picture quality:

- CBR - Constant Bit Rate (CBR)
- VBR - Variable Bit Rate (VBR)
- 2-PASS VBR - 2-Pass Variable Bit Rate

CBR

As the name suggests this method will encode each frame of video with the same bitrate, no matter what his happening in the frame or what is changing from frame to frame. Encodes done in this method often have lower visual quality but also lower file sizes. This method is also the fastest in terms of transcode time.

USE CASE:

A news organization may choose to encode CBR so that the video is outputted quickly enough to make a news deadline. Time for them is more important than video quality. If they choose CBR (or even 1-Pass VBR) they may make the decision to encode it at a medium resolution (say 480) at a mid-level bit rate (say 1200 kbps) so that they can maintain some level of visual quality.

VBR

Unlike Constant, Variable Bit Rate adjusts the amount of bits assigned to a frame depending on what the encoder believes is happening in the frame. This means the bit rate fluctuates over time, going over the average when a complex frame is encountered. How far above the average bit rate is determined by setting the Max Bit rate appropriately. (discussed in a later section)

2-PASS VBR

This is the most recommended method for transcoding when picture quality is important. Like VBR, 2-Pass allows the bit rate to increase for complex scenes (say a rainy scene where every frame is different). Unlike straight VBR 2- Pass does a little more work. It does a first pass at the encode that creates a log file. This file is then used on a second pass to improve quality on difficult scenes. This results in a higher picture quality (fairly significant compared to 1-Pass) and a more consistent stream with fewer data/bit rate spikes.

Bandwidth

Bandwidth refers to the amount of data that a user may be able to receive and/or transmit. Often this is thought in terms of kilobits per second and is separated into both upload and download ability. Internet Service Providers tend to cap these data rates depending on the plan that the user has purchased which presents a challenge to content creators as they try to meet the needs of a broad range of users who have a varied range of bandwidths.

> NOTE: According to Cisco's report Mobile connections in the U.S. currently are at 1,059 kbps and by 2014 should be, per their forecast, 2902 kbps.

A U.S. customer may only have a connection as low as around 3 Mbps or, if they could afford a better plan, up to 50 Mbps -- that's an incredibly wide range of possible connections for which to account and presents a challenge. On one hand a content creator wants their guests to experience the highest level of quality, but on the other hand that guest may simply not be capable of handling that quality. So a balance, yet again, must be considered.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1602
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

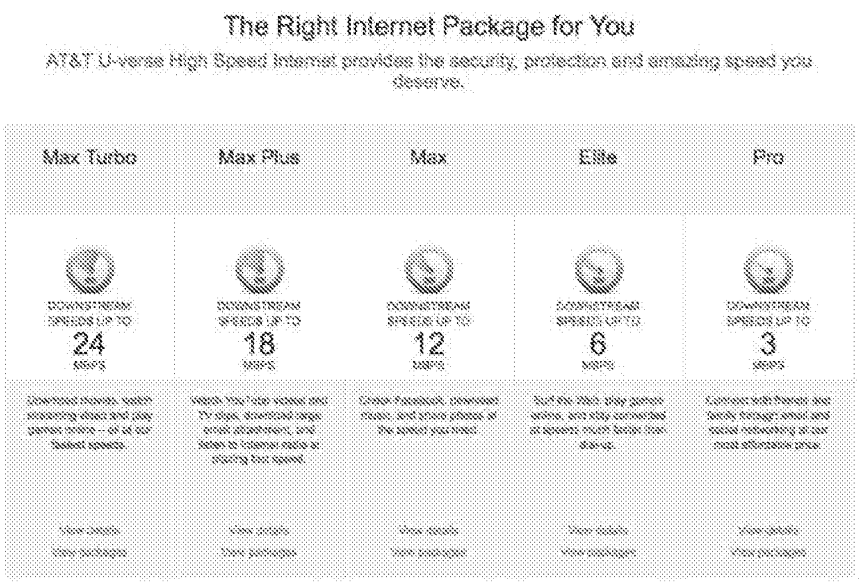
Bit Rate

A BitRate is a measurement of data speed across a network, often in Kilobits per second or kbps (1000 bits per second). This number correlates with potential bandwidth levels that a user may experience and should be in balance to the resolution of the stream.

USE CASE:

A household who's data plan is limited to 3 Mbps can not handle a bit rate that peaks to over 2500 kbps. You may think that because their data plan says they can handle up to 3000 kbps (that's 3 Mbps in kilobits) that they would be able to use all of that bandwidth for streaming. There a couple of reasons why this isn't the case. First -- an average data rate of around 2500 kbps may spike to at least 30% and at times up to and even over 80% of the average at various points in the video stream if the content creator has transcoded using variable bit rate, which is a common method. Secondly -- the user may have a CPU that cannot take advantage of the entire 3 Mbps, especially if other programs are active in the background. This could be because they have an older system and have yet to be able to upgrade.

Example: AT&T's ISP Connections By Data Plan (Source)



If you are using Akamai HD HTTP Streaming bit rate spikes are less of an issue as it uses client side caching allowing for a higher threshold -- however if network conditions worsen those spikes may still present a problem.

USE CASE: A Modern Family -

A household of four shares the same wireless internet connection but has many devices using that network all at the same time. It is entirely possible that this one household of four could have a father using the internet on his laptop, a mother using an iPhone streaming Pandora, one child on an Xbox and another streaming a movie in the living room on their HD television using a set top box -- all on the same wireless connection! Let's hope that the kid watching the movie in the living room is using adaptive streaming with a wide range of bit rates!

Again it is always about a balance between performance and visual quality.

Average vs. Max Bit Rate

The Video Bit rate has two main components: The Average and the Max.

Average Bit Rate

The average bit rate for video should coincide with the target bandwidth of the end user and should be in balance with the resolution)

Just changing a bit rate alone is not sufficient for dealing with bandwidth limitations and is not recommended. It is advised that low bitrates also scale down in resolution so that the end user has a good balance between image and playback quality.

USE CASE:

It is not advisable to send a mobile smart phone user a 1080 resolution at a low bitrate of 500kbps not only because the image quality will be unwatchable but also because the end user, though able to handle 500 kbps, does not possess a system that can comfortably handle 1080 lines of resolution. Potentially such a user would have their player crash or at the least a very poor playback experience if they tried to watch that 1080p video on a smart phone. Additionally low bitrates at high resolutions show larger digital artefacts/errors which make the image look undesirable. At such low bitrates it is better to reduce the resolution so that the resulting video is compressed enough to have a small file size, and

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1603
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

then you can rely on the end user's GPU to scale-up.

Max Bit Rate

A Max Bit Rate governs the ceiling that a variable bit rate may reach and should be within balance of the average and the target connection/device. Max Bit rates do not effect Progressive Download and some services like Akamai have some built in functionality that limit the impact of bit rate spikes. That being said, not everyone uses Akamai and Progressive download is not often ideal for streaming. (A high max bit rates essentially make the video bitrate unreliable complicating streaming applications.)

Potential Performance Issues Due to a High Max Bit Rate:

- Frequent or Unwanted Bit Rate Switching
- Playback Stuttering
- Frequent Buffering
- Player Crashing

Industry standards say you should calculate the Max bit rate by taking the Average and adding 50%. It is recommended to reduce this even to 30% in order to create a truly consistent stream but still taking advantage of a variable bit rate.

USE CASE:

If a transcoded stream has an Average of say 1400 kbps but spikes at 2600 kbps that user may experience one of the above performance issues. This is especially true if their bandwidth can only support between 1000 kbps and 2000 kbps, which is highly likely for many users who's ISP's cap their data plans. This would mean that when the stream spiked to 2600 it is outside the threshold that the user can handle and they will have a poor playback experience. Also – remember the earlier example of that family of four all using their devices on the same wireless network!

Adaptive Bit Rate Requirements

Bit rates should be at least 400 kbps apart in an adaptive set for the mid range to high range encodes. The lower encodes can be closer together and should scale down in resolution. The two settings, bit rate and resolution, should always be in alignment per industry practice.

There is no need to create a low bit rate for 720 and higher resolutions for two primary reasons – first image quality would be very poor as the loss of detail would become too great for a good viewing experience and secondly because a lower bitrate may be being consumed by a mobile device that can not handle a high resolution. For example - a 3GS iPhone can not stream 720 or higher resolution video without taxing the devices system and causing poor performance. Content creators want to reach the largest possible audience – it is always important to consider backwards compatibility when designing adaptive sets. More recent devices may have more latitude, especially newer 4G and Tablets such as the iPad 2. Later in this document we will detail potential adaptive sets for various devices.

> NOTE: Lower bitrates are NOT meant to look perfect. They are only meant to keep the signal from being interrupted and to minimize the risk of the player crashing. The hope is that these lower bitrates only appear briefly while the users bandwidth fluctuates. See the mobile section for use cases where this is exemplified.

Resolutions

The resolution of a video is measured in the height in pixels. A content creator could use any resolution she wanted however there are industry standards to consider.

Though larger and newer formats are emerging currently the preferred resolutions for streaming are:

- 480p – (SD) Standard Definition
- 720p – (hd) mini-High definition
- 1080p – (HD) True High Definition

HD – 1080p

True High Definition, HD, is defined as 1080 lines of horizontal progressive resolution. Think of your television as a series of very small lines – these lines make up the image of one frame and are scanned onto the screen in order, one line progressing after the other. Another way to imagine it is to take a painting and cut it into 1080 thin strips that are 1 pixel tall (very small) and then reassemble those strips one at a time in the order they would appear going from the top of the frame to the bottom. This form of High Definition is referred to as 1080p – where the 'p' stands for progressive and is identified by the HD acronym. The highest resolution possible for internet streaming is 1920 x 1080, or 1080p.

> NOTE: 1080i video also exists and uses an interlaced method similar to standard definition broadcast and along with 720p is the more common, modern broadcast format. 1080i is not used for internet streaming.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1604
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

hd – 720

In addition to 1080p and 1080i there is also a 720p resolution. This resolution is used for broadcast television and sometimes, independent filmmaking and home video. It is of a lower resolution than it's big brother 1080 which makes it easier to transmit and store.

This smaller version of High Definition can sometimes be referred to with the lower case acronym, hd. This is, by the way, not true high definition. High definition must be at least twice the resolution of Standard Definition Video, which is comprised of 525 lines of resolution.

HISTORY:

The precursor to the High Definition Television needed to carry a signal over a frequency wave – transmitted through the air from antennas to antennae. This method required bundling all the video and audio information into this signal so that when it reached a television set it was decoded properly. Standard Definition television's used an interlace method to deliver this signal – meaning odd lines where scanned onto the television screen followed then by the even lines. This was accomplished by having one frame of video comprised of two fields – field A would contain the odd scan lines and field B the even – combined, or interlaced, they presented a completed image. Each frame of video contained 525 lines of horizontal resolution. This standard also is referred to as NTSC which is the North American broadcast standard that set the television frame rate, at the time, to 29.97 frames per second. This is in contrast to the European, or PAL, method, which uses 625 lines of resolution at 25 frames per second.

It is important to understand this history when dealing with modern digital delivery as many users, especially at the Studio Level, may need to handle old footage that was created for analogue standard definition signals and will need to be converted it so that it can be streamed.

SD – 480p

Standard Definition is still used for Internet streaming as it's smaller file sizes and simpler resolution often meets the right balance for end users to have a positive playback experience. Standard Definition for streaming is not the same as broadcast standard definition – though qualitatively they may look similar. For Internet Streaming a common standard definition resolution is 480 progressive lines of resolution, or 480p.

Resolution Considerations (MOD-16)

When determining what resolution and/or frame size to use it is important to keep those dimensions divisible by 16. This is because encoders divide the image into 16 x 16 pixel macro blocks. If for some reason 16 is not a possible divisor for you it is also okay to use 8 though some loss in quality may occur.

Resolution vs. Dimension

It is easy to get resolution and dimension confused – primarily because they often refer to the same thing. However Resolution refers to the amount of detail the image has and Dimension only refers to the Aspect Ratio of that image. Both are measured in Width and Height.

Resolutions should scale according to the desired device and bitrate. It is advisable to not try to scale video up but rather present it in the original resolution of the master. Scaling up will result in noticeable degradation in picture quality. Likewise is it unadvisable to try to fit a larger dimension (a.k.a. frame size) onto a device that has a smaller resolution. (i.e. trying to fit a 1080 HD image onto an iPhone that can only handle 480 and below)

An Aspect Ratio is indicated by notating the relationship between the width and height in a numerical value such as 1.33 – now 1.33 is shorthand and means that for ever 1 unit of height there is 1.33 units of width. The correct way of notating 1.33 would be 1.33:1 or 1.33x1.

Here are some common Theatrical and Broadcast Aspect Ratios:

Common Aspect Ratios

1.33	Sometimes referred to as Academy Aspect Ratio and is 4x3 content
1.78	16x9 Television and the most common Film Aspect Ratio
1.66	An older aspect ratio popular with some filmmakers and animators
1.85	A very common aspect ratio used a great deal in 1980-1990's features
2.35	The next most used aspect ratio for theatrical films after 1.78
2.40	A variation on 2.35

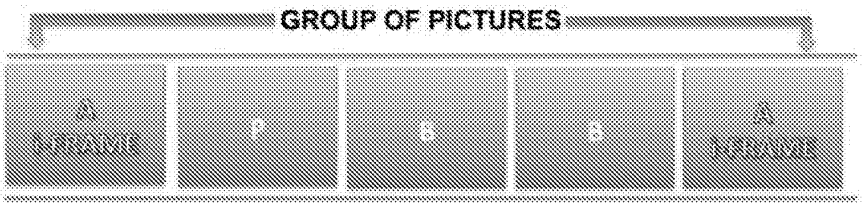
NOTE: An Aspect Ratio is calculated by dividing

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1605
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

the width by the height. 178 is actually 1.78 to 1 as a ratio of that width to height. A Resolution of 848x480 using this formula, for example, comes to 1.766 or rounded up, 1.78 -- tradition drops the decimal and so this is simply called 176. As you calculated your resolutions keep this formula and the need to have the resolutions be divisible by 16 in mind (as previously discussed).

Group Of Pictures And I-Frame Intervals

The key to video encoding is the use of compression -- taking a large resolution image as it might appear on 35mm film or HD video and making it viewable by a home user. Though bit rates and resolutions do much of the work to create a balanced viewing experience there are other factors that also contribute to this compression. The most vital of these is the I-Frame Interval, sometimes referred to as the Key Frame Interval or the Instantaneous Decoder Refresh (IDR). All refer to essentially the same thing -- the need for a reference frame from which guesses can be made about the subsequent frames. Every frame of video of a master once transcoded is not fully represented.



Every encoder has the ability to call out how far apart key frames are created -- this is referred to as the Key Frame Interval also known as the I-Frame Interval. The frames in between Key frames are called P-Frames (predictive coded picture) and B-Frames (bi-directionally predictive) . P frames make a guess as to what the picture looks like based on those Key Frames. B frames are additional reference frames that help improve the quality of P-Frames and the overall look of the stream.

Ideal Key Frame Intervals should be between 2 and 4 seconds. Meaning, for a 29.97 fps piece of video, the key frames should be between 60 frames and 120 frames.

B-Frame usage should be limited to 1 to 2 reference frames -- going over 3 reference frames may cause poor playback on some players (Quicktime for example). However, if you are confident that your users will be using players that support B-Frame decoding you can increase these B-Frames to increase picture quality. Understand though that you will be increasing file size which may slow down load times and cause some buffering.

IMPORTANT CONSIDERATIONS:

Key Frames are used for a variety of purposes and not just for the compression of the video. They are also Adaptive Switch and Seek Points -- an adaptive player will switch bitrates when needed at Key Frames and when seeking on a player's timeline -- those seek point are tied to I-Frames. If the Key Frame Interval is too short bit rate switching may happen too frequently and if too long a user's player may crash if a switch is required because the users' bandwidth has dropped. Also if a Key Frame Interval is too close together file sizes increase and the work needed to decode the picture increases causing a poor playback experience, especially for users with mid to low range bandwidths. For seeking considerations an interval that is very long means that a user will not be able to seek within the video between those points -- causing a poor experience.

Buffer

The player loads information from a video payload (your encoded video asset) ahead of your real time playback. This buffer should be in balance between the bit rate and connection speed. To calculate the Bitrate Buffer Size all you need to do is take the Average bit rate and add at least 50% so that the buffer would be 150% of Average.

Profiles

There are three primary encoding methods used -- these Profiles allow different levels of complexity in the video stream and are useful, if not required, to find that balance between connection and stream. The three levels are:

- **Baseline** -- This is the simplest method used and the most limiting. One would use this setting for mid to low end delivery on a mobile device or for videoconferencing. The image quality will be limited as neither VBR nor B-Frames are allowed.
- **Main** -- Used primarily for standard definition broadcast and streaming. (Level 3.1)
- **High** -- Used for high definition video streams and is the profile used for Blu-Ray authoring.

Video And Playback Artifacts

If the above settings are not followed a variety of playback artifacts or errors may appear. These artifacts may include:

- **Buffering** -- If there are huge bit rate spikes or if file sizes are large for various bit rates a player may stop playback and pause -- loading the video and waiting for system resources to free up. This also may be because the buffer is set too low during transcoding.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1606
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

- **Frame Skips/Drops** -- Frames of video may drop if signal strength is not sufficient or if the wrong frame rate was used during transcoding. Resolution may also be a culprit of frame skips. If a user is using an average laptop over a wireless connection in their home, 720 and 1080 resolutions will most likely skip frames, creating a choppy playback experience.
- **Stutter** -- Like frame skips, frames are dropping, but the perceived experience is that the video is stuttering. This may appear as if a frame is pausing for a split second before catching up to audio which normally does not stutter or skip. (audio typically plays back fine even if video artifacts are present as the two, though bundled in the same container, are treated separately)
- **Macro-Blocking (or just Blocking)** -- The image may appear as a mosaic of blocks -- this is often referred to as pixilation. This especially happens on fast motion scenes or scenes with heavy detail like rain.
- **Aliasing** -- Lower resolutions do not handle diagonal lines well -- these may appear as steps and be jagged rather than smooth diagonals.
- **Banding** -- Though on a video master an image may appear to have a consistent color, often after transcoding to mid to low bit rates/resolutions that same color appears as a gradient or a series of bands. This is referred to as banding. An encoder, especially if 1-Pass or CBR was used, may not be able to tell the difference between subtle color changes and instead presents them as big changes, rather than subtle shifts. This is most often in sky shots and in animation; the latter is most problematic and challenging to transcode -- especially modern CG based animation. Many transcoders have built in algorithms to deal with banding and over time this issue should dissipate.
- **Ghosting/Gibbs Effect/Ringing/Mosquito Noise** -- The image may seem doubled with the edges of an object seeming to flutter at a high rate. You would see a faint outline around an object or subject in the video that appears to fluctuate from frame to frame.

Audio

Audio tends to be simpler to transmit and is significantly lower than a Video bit rate. Audio Bit Rates range from 20kbps up to only 384 kbps (for 5.1 Audio Only) and typically use 44.1 khz per second or 22 khz for a sample rate.

The Best Practices Charts

Below you will find a series of charts detailing the best practice settings. These are meant as guidelines. You can move from these settings but always remember you are trying to achieve a balance and there are potential consequences for varying from recommendations.

The charts assume 23.98 fps which is fast becoming the standard for media content. If your content is 29.97 (or 30) fps the key frame interval would be all that would need adjustment from what is presented here. For example: The Key Frame interval for the SD-High_2612 at 23.98 fps is 96 (or roughly 4 seconds) If it was instead 29.97 fps the interval, for four seconds, would be 119.

Also, all these charts only reference 16x9, 178 material. Resolutions/Dimensions should be adjusted according to your source.

Video Masters

A transcode is only as good as the Master it was made from.

Transcoding always starts with a Video Master. **The transcode can never be of a better quality than the Master.**

A master file intended for transcoding is often referred to as a Mezzanine file -- as it's an intermediary step in the postproduction process. A true 4k uncompressed piece of video is far too large to even play back on most computers and certainly too big to transcode from as it would take a substantial amount of time to even move the files so that an encoder can pick it up and complete the transcode.

Ideally the video master should be at the native resolution of active picture -- in other words -- if the video has an aspect ratio of 178 than the master should be 178, if the video, however, was 235 the master should also be 235 with no burned in matting.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1607
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)



The above is an example of a 1.78 master that contains 2.40 active pictures - meaning that if you looked at this video master on a monitor you would see black mattes recorded, or burned, into the master. If this master was transcoded as 1.78 those black mattes would remain and take away from the image quality by consuming more bits. Mattes should be removed to maximize picture quality and the active picture should be reflected in the final output dimensions.

Here is a chart detailing possible Master settings:

Assumptions:
H.264/MPEG-4 AVC or ProRes 422

	Resolution	
Bit Rate	Width	Height
80 Mbps	1920	1080
50 Mbps	1920	1080
25 Mbps	1920	1080
15 Mbps	1920	1080
80 Mbps	1280	720
50 Mbps	1280	720
25 Mbps	1280	720
15 Mbps	1280	720

You may wish to create a sub-master that is easier to transmit to partners and can be used to transcode lesser quality streams. This chart details what a sub-master transcode could look like:

Intermediary Master aka Sub-Master Settings

Kaltura Flavor Name	A/V Bit Rate	Video Bit Rate	Audio Bit Rate	Width	Height	Type	Key Frame Interval	Frames Per Second
HD-Master	8128	8000	128	Auto	1080	High	96	23.98
hs-Master	8128	8000	128	Auto	720	High	96	23.98
SD-Master	8128	8000	128	Auto	480	High	96	23.98

Desktop Settings

Settings for Desktop applications differ greatly from those used on other devices primarily because a desktop may be able to handle a more complex set of transcodes albeit with a wider variety of potential stream killers. Though desktop can serve up even High Definition most users still do not have home desktop systems nor ISP connections that can stream HD in a manner that is acceptable. Currently HD streams cause skip frames and stuttering and may even crash the video player causing a poor user experience. As bandwidths and devices improve these problems will dissipate but it is important to understand this when designing your desktop bit rates as you must account for what consumers can actually handle, not what you wish they could handle.

The following chart details the Best Practices for Desktop:

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1608
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Desktop Best Practices

Kaltura Flavor Name	A/V Bit Rate	Video Bit Rate	Audio Bit Rate	Width	Height	Type	Key Frame Interval	Frames Per Second
HD-High_8128	8128	8000	128	Auto	1080	High	96	23.98
HD-High_5128	5128	5000	128	Auto	720	High	96	23.98
hs-High_4628	4628	4500	128	Auto	720	High	96	23.98
hd-High_3628	3628	3500	128	Auto	720	High	96	23.98
SD-High_2812	2812	2682	128	Auto	480	Main	96	23.98
SD-High_1616	1616	1488	128	Auto	480	Main	96	23.98
SD-Med_1028	1028	900	128	Auto	480	Main	96	23.98
SD-Med_816	816	688	128	Auto	480	Main	96	23.98
SD-Low_552	552	488	64	Auto	352	Main	96	23.98
SD-Low_448	448	384	64	Auto	352	Main	96	23.98

Mobile Settings

Mobile requirements are separated into two device platform sets -- iOS for Apple devices and Android/Other for non-Apple mobile devices.

IMPORTANT CONSIDERATIONS:

Though it may be tempting to try to fit your streaming video to the maximum resolution/dimension of the target device it is not recommended. Each device requires a small buffer to allow processing. This buffer is achieved partially by feeding video at a slightly smaller resolution/dimension than the max allowed for that device. This is improving and even recently Apple has begun adding full resolution to their recommendations -- though some playback problems may still occur. This is primarily true for older non-iOS devices.

iOS Best Practices (Mobile)

Apple has strict requirements for how video should stream. The following is based upon their recommendation.

iOS Best Practices

Kaltura Flavor Name	A/V Bit Rate	Video Bit Rate	Audio Bit Rate	Width	Height	Type	Key Frame Interval	Frames Per Second
iOS_WIFI_hd_4540	4540	4500	40	1024	576	Main	72	23.98
iOS_WIFI_hd_2540	2540	2500	40	1024	576	Main	72	23.98
iOS_WIFI_SD_1840	1800	1800	40	960	544	Main	72	23.98
iOS_WIFI_SD_1240	1240	1200	40	640	352	Base	72	23.98
iOS_WIFI_SD_640	640	600	40	640	352	Base	72	23.98
iOS_CELL_SD_440	440	400	40	416	234	Base	72	23.98
iOS_CELL_SD_240	240	200	40	416	234	Base	36	12
iOS_CELL_SD_110	150	110	40	416	234	Base	24	8

NOTE: Apple also recommends creating an audio only bit rate at 64 Kbps. This is so that if the user's connection drops very low between cell zones the stream will not be interrupted and will continue playing, keeping the user engaged, until the signal improves. This audio only bitrate would contain, ideally, a still image that appeared while the audio played back.

IOS USE CASE #1:

A Media Company has decided to only offer 3 mobile bit rates:

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1609
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

- 320 x 240 @ 128 kbps
- 480 x 320 @ 414 kbps
- 480 x 320 @ 564 kbps

This Media Company is trying to achieve a lot with just three bitrates. The 320x240 bit rate is an iPod dimension and they feel comfortable with this bitrate showing up on an iPhone or iPad with black bars on all sides. They also have two other bitrates at 320 resolution within 100 kbps of each other which are a bit too close together to create any benefit.

These bitrates will work but will result in potentially a poor user experience as the player may have a difficult time staying on one of the two upper bit rates - the 414 and 564 -- and may switch between them. Also if the end user is viewing these on an iPAD 2 with a 4G connection these bitrates would not be using the full potential of that device -- that is why it is recommended to have bit rates over 500 for those higher end uses.

IOS USE CASE #2:

Consider a real world scenario with a slightly different set of variants:

A single mother is driving from Los Angeles to San Diego with her daughter. Her daughter, age 7, is watching an iPAD 2 during the journey but she is watching a stream that only has 3 bit rates -- 1240 kbps, 640 and 440. The content creator likes the look of these on an iPad but doesn't want to go below 440 because of the image quality loss. The daughter is enjoying her film but because they are traveling down a highway they are moving from one zone to another and her signal fluctuates wildly. Because the content creator is not offering a bit rate below 440 the player crashes forcing the mother to either tell her daughter to do something else or she must decide to pull over and reload her child's iPAD and re-launch the player.

> NOTE: The way iOS works is that each of the above bit rates would be segmented into small chunks (10 seconds to maybe a minute) that are then called by the player one at t time in a progressive manner. This allows the amount of data flowing to a mobile device to be in small enough packets to allow smooth playback, limiting the drag on system resources.

Android/Other Best Practices (Mobile)

Android/Other Best Practices
***Dimensions/Resolutions reflect Horizontal View Only**

Kaltura Flavor Name	A/V Bit Rate	Video Bit Rate	Audio Bit Rate	Width	Height	Type	Key Frame Interval	Frames Per Second
Low_Edge_80	80	56	24	128	72	Base	48	23.98
Low_Edge_150	150	102	48	128	72	Base	48	23.98
Low_Edge_250	250	202	48	256	144	Base	48	23.98
Low_Edge_400	400	352	48	256	144	Base	48	23.98
Low_3G4G_600	600	552	48	512	288	Base	48	23.98
Low_3G4G_800	800	736	48	512	288	Base	48	23.98
Med_4GWIFI_1000	1000	936	64	512	288	Base	48	23.98
Med_4GWIFI_1300	1300	1236	64	512	288	Base	48	23.98
Med_4GWIFI_1600	1600	1536	64	512	288	Base	48	23.98
High_4GWIFI_1900	1900	1836	64	512	288	Base	48	23.98
High_4GWIFI_2200	2200	2136	64	512	288	Base	48	23.98
High_4GWIFI_2500	2500	2436	64	512	288	Base	48	23.98
Adv_4GWIFI_1200	1200	1136	64	768	432	Main	48	23.98
Adv_4GWIFI_1600	1600	1536	64	768	432	Main	48	23.98
Adv_4GWIFI_1900	1900	1836	64	768	432	Main	48	23.98

The above bit rates may appear at first to be daunting but remember -- you can always pair down a set of bitrates as long as you include a wide enough range. The downside of having too few bit rates spanning several bandwidth levels is that the shift in quality becomes more obvious as the player switches between them. One reason for a larger number is

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1610
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

simply to give the end user the smoothest experience. Also for non-iOS devices there is a wider range of possible devices to account for -- thus the long list of potential encodes. The above is based on Adobe's recommendation, combining their Variant 8 and 9 charts.

Connected Devices Best Practices (Roku, Boxee, TV, etc.)

Connected Devices Best Practices (Roku, Boxee, TV, etc.)

Kaltura Flavor Name	A/V Bit Rate	Video Bit Rate	Audio Bit Rate	Width	Height	Type	Key Frame Interval	Frames Per Second
HD-High_5128	5128	5000	128	Auto	1080	High	96	23.98
hd-High_3628	3628	3500	128	Auto	720	High	96	23.98
SD-High_2612	2612	2484	128	Auto	480	Main	96	23.98
SD-Med_1618	1618	1488	128	Auto	480	Main	96	23.98
SD-Med_1028	1028	900	128	Auto	480	Main	96	23.98
SD-Low_816	816	688	128	Auto	480	Main	96	23.98
SD-Low_552	552	488	64	Auto	480	Main	96	23.98

* The above dimensions and bit rates are based on the requirements for the Roku player.

Kaltura Transcoding Flavors Best Practices Master Chart

Download the above charts combined as PDF.

HTML5 Video Formats Browser Compatibility

HTML 5 VIDEO FORMAT BROWSER COMPATIBILITY CHART

	VIDEO FORMAT	AUDIO FORMAT	BROWSER							
			Internet Explorer	Mozilla Firefox	Safari	Google Chrome	Chromium	Opera	Konqueror	Epiphany
HMTL5	Ogg Theora	Vorbis	No	Yes	Yes*	Yes	Yes	Yes	Yes	Yes
	VP8 (WebM)	Vorbis	Yes*	Yes	Yes*	Yes	Yes	Yes	Yes	Yes
	H.264	AAC	Yes	No**	Yes*	Yes***	Yes*	No	Yes*	Yes*
FLASH	VP6	MP3/Speex/PCM	Yes	Yes	Yes	No		Yes		
	H264/Mpeg-4	AAC/Vorbis	Yes	Yes	Yes	No		Yes		

* Denotes Manual Install - meaning user will have to manually install a plugin or application.
** Possible on some systems if the HTML5 extension for Windows Media Player Firefox plug-in is installed [28] - Note taken from Wikipedia.
*** Support for H.264 may be removed shortly as they are trying to strictly use VP8.

Above chart references:

1. http://html5video.org/wiki/Kaltura_Video_Library_Compatibility_Chart
2. http://en.wikipedia.org/wiki/HTML5_video#Table
3. http://en.wikipedia.org/wiki/Comparison_of_container_formats
4. http://kb2.adobe.com/cps/141/tn_14156.html

STREAMING CONTAINER SUMMARY

Most Common Formats in 2011/2012

CONTAINER	VIDEO FORMAT	AUDIO FORMAT	BACKWARDS COMPATIBLE	CHAPTERS	SUBTITLES /CLOSED CAPTIONS
MP4	H264, MPEG2,MPEG4	AAC, AC-3, Vorbis	YES*	YES	YES
WebM	VP8	Vorbis	NO	NO	NO
FLV	VP6, Sorenson, H264	MP3, AAC, PCM	YES	NO	NO
F4V	H264	MP3, AAC, HE-AAC	YES	-	-
3GP	H264, H263	AAC, HE-AAC	YES	-	YES

* Will not function properly on older smart phones - for those older phones, 3GP is recommended.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1611
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

I found this useful | 25 other people found this useful.

(38775 reads)

Related URL:

Recommended Video Source Formats and Specifications (Kaltura)
Bandwidth Caps for Residential High-Speed Internet in the U.S. and Japan
Dynamic stream switching with Flash Media Server 3 - Understanding the multi-bit rate solution
VBV - Video Buffering Verifier (Wikipedia)
VBV - Video Buffering Verifier
Best Practices for Creating and Deploying HTTP Live Streaming Media for the iPhone and iPad
X264 Encoding Suggestions
Encoding.com H.264 (AVC) FAQ
Transforming the HD Video Experience With Akamai Flash Streaming Solutions
Video encoding and transcoding recommendations for HTTP Dynamic Streaming on the Flash Platform

Market Industry:

All

Resources for download:

Best Practices Master Chart.pdf

DOWNLOAD & INSTALL KALTURA CE | START FREE TRIAL | CONTACT SALES

Kaltura is the world's first Open Source Online Video Platform, providing both enterprise level commercial software and services, fully supported and maintained by Kaltura, as well as free open-source community supported solutions, for video publishing, management, syndication and monetization.

Stay in Touch

Search

Products
Features
Applications
Deployments
Kaltura Exchange
What's New?
Kaltura API

Solutions
Solutions Overview
Media and Entertainment
Education
Enterprise
Service Providers

Partners
Channel Partners
Integrators
Partners Overview
Technology Partners
Partner Portal

Why Kaltura
Free Trial
Contact Sales

Customers
Case Studies

Community
Kaltura.org
Community Blog
Events & Meetups
HTML5Video.org
OpenTec Blog

Company
About Kaltura
News
Careers
Contact Us
Giving Back
Kaltura Team
Webinars
Events
Login

Copyright © 2013 Kaltura Inc.
All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this web site constitutes acceptance of the Terms of Use and Privacy Policy. User submitted media on this site is licensed under: Creative Commons Attribution-Share Alike 3.0 Unported License.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1612
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Container format (digital)

From Wikipedia, the free encyclopedia

A **container** or **wrapper format** is a file format, or often a stream format (the stream need not be stored as a file) whose specifications regard only the way data are stored (but *not* coded) within the file, and how many metadata could or are effectively stored, whereas no specific codification of the data themselves is implied or specified. A wrapper format is, in fact, a meta-format, because it stores the real data *and* the information about how such data are stored within the file itself. As a consequence of that, a program which is able to correctly identify and open a file (read a stream) written in such a format might not be able to subsequently decode the actual data stored within, because either the metadata in the wrapper file are not sufficient *or* the software lacks that specific decoding algorithm enroled in the metadata to interpret the actual data the file "wraps" around.

Stated like that, a container format could, in theory, wrap any kind of data. Although there exist a few examples of such file formats (Microsoft Windows' DLL files are just an implicit one), most wrappers exist for particular data groups. This is due to the specific requirements of the desired information. The most relevant family of wrappers is, in fact, to be found among multimedia file formats, where the audio and/or video streams can effectively be coded with hundreds of different alternative algorithms, whereas they are stored in fewer file formats. In this case the algorithm (or algorithms, as in the case of mixed audio and video contents in a single video file format) used to actually store the data is called a *codec* (shorthand for coder/decoder).

Contents

- 1 Multimedia container formats
- 2 Single coding formats
- 3 Issues
- 4 See also
  - 4.1 Similar packaging structures
- 5 References

Multimedia container formats

The container file is used to identify and interleave the different data types. Simpler container formats can contain different types of audio codecs, while more advanced container formats can support multiple audio and video streams, subtitles, chapter-information, and meta-data (tags) — along with the synchronization information needed to play back the various streams together. In most cases, the file header, most of the metadata and the synchro chunks are specified by the container format (for example container formats exist for optimized, low-quality, internet video streaming which, for example, differs from high-quality DVD streaming requirements).

The constituent parts of a container format have various names; they are often called "chunks", as in RIFF and PNG, while they are called "packets" in MPEG-TS (from the communications term), and they are called "segments" in JPEG. The main content of a chunk is called the "data" or the "payload". Most container formats have chunks in sequence, each with a header, while TIFF unusually instead stores offsets, which results in difficulties in properly preserving information – notably, Exif photo data is often discarded. Modular chunks make it easy to recover other chunks in case of file corruption or

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1613
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

dropped frames or bit slip, while offsets result in framing errors in cases of bit slip.

Some containers are exclusive to audio:

- AIFF (IFF file format, widely used on Mac OS platform)
- WAV (RIFF file format, widely used on Windows platform)
- XMF (Extensible Music Format)

Other containers are exclusive to still images:

- FITS (Flexible Image Transport System) is a wrapper file format for still images, raw data, and associated metadata.
- TIFF (Tagged Image File Format) is a wrapper file format for still images and associated metadata.

Other flexible containers can hold many types of audio and video, as well as other media. The most popular multi-media containers are:

- 3GP (used by many mobile phones; based on the ISO base media file format)
- ANIM Standard multimedia file for digital animations on classical Commodore Amiga line of computers. It follows IFF ILBM main specifications, and it is the first animation format adopted officially by an Operating System.
- ASF (standard container for Microsoft WMA and WMV)
- AVI (the standard Microsoft Windows container, also based on RIFF)
- CDXL (another name of ANIM format), it was the animation file format standard of CDTV and Amiga CD32
- DVR-MS ("Microsoft Digital Video Recording", proprietary video container format developed by Microsoft based on ASF)
- IFF (first platform-independent container format)
- Matroska (MKV) (not standard for any codec or system, as it can hold virtually anything. It is an open standard and open source container format).
- MJ2 - Motion JPEG 2000 file format, based on the ISO base media file format which is defined in MPEG-4 Part 12 and JPEG 2000 Part 12
- MOV (standard QuickTime video container from Apple Inc.)
- MPEG program stream (standard container for MPEG-1 and MPEG-2 elementary streams on reasonably reliable media such as disks; used also on DVD-Video discs)
- MPEG-2 transport stream (TS) (a.k.a. MPEG-TS) (standard container for digital broadcasting and for transportation over unreliable media; used also on Blu-ray Disc Video; typically contains multiple video and audio streams, and an electronic program guide)
- MP4 (standard audio and video container for the MPEG-4 multimedia portfolio, based on the ISO base media file format defined in MPEG-4 Part 12 and JPEG 2000 Part 12)
- Ogg (standard container for Xiph.org audio codec Vorbis and video codec Theora)
- OGM ("Ogg Media", video container for Xiph.org codecs, no longer supported and formally discouraged by Xiph.org[1].)
- RealMedia (standard container for RealVideo and RealAudio)

There are many other container formats, such as NUT, MXF, GXF, ratDVD, SVI, VOB and DivX Media Format

See the Comparison of container formats for details regarding these formats.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1614
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Single coding formats

In addition to pure container formats, which specify *only* the wrapper but not the coding, a number of file formats specify *both* a storage layer and the coding, as part of modular design and forward compatibility.

Examples include JPEG File Interchange Format (JFIF) for containing JPEG data, and Portable Network Graphics (PNG).

In principle, coding can be changed while the storage layer is retained; for example, Multiple-image Network Graphics (MNG) uses the PNG container format but provides animation, while JPEG Network Graphics (JNG) puts JPEG encoded data in a PNG container; in both cases however, the different formats have different magic numbers – the format specifies the coding, though a MNG can contain both PNG-encoded images and JPEG-encoded images.

Issues

The differences between various container formats arise from five main issues:

1. Popularity; how widely supported a container is.
2. Overhead. This is the difference in file-size between two files with the same content in a different container.
3. Support for advanced codec functionality. Older formats such as AVI do not support new codec features like B-frames, VBR audio, VFR natively. The format may be "hacked" to add support, but this creates compatibility problems.
4. Support for advanced content, such as chapters, subtitles, meta-tags, user-data.
5. Support of streaming media

See also

- Comparison of (audio/video) container formats
- Open source codecs and containers
  - FFmpeg, cross-platform, open source, audio and video codec suite/library
- List of multimedia (audio/video) codecs
  - Comparison of video codecs
  - Comparison of audio codecs

Similar packaging structures

- Archive format
- Metafile

References

1. ^ Xiph.org does not support OGM

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1615
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Retrieved from "http://en.wikipedia.org/wiki/Container_format_(digital)"
Categories: Computer and telecommunication standards | Computer file formats | Container formats | Files | Lists of file formats | Film and video technology

- This page was last modified on 17 August 2009 at 15:49.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of Use for details.
  Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1616
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

DVD - MPeg differences

DVD uses a subset of MPeg standards ISO/IEC 11172 (MPeg-1) and ISO/IEC 13818 (MPeg-2).

Restrictions

- Streams - 1 video, up to 8 audio
- Coding - MPeg-1 vbr/cbr, MPeg-2 MP@ML or SP@ML vbr/cbr
- Data rate
  - MPeg-1 maximum video 1856Kbps, maximum audio 384Kbps
  - Mpeg-2 maximum video 9800Kbps, maximum audio 912Kbps

  note: audio options other than MPeg exist, see extensions
- Image size
  - Mpeg-1 352x240 (NTSC) or 352x288 (PAL/SECAM)
  - Mpeg-2 720x480, 704x480, 352x480 (NTSC)
    720x576, 704x576, 352x576 (PAL/SECAM)
- Frame rate
  - coded - 24fps progressive, 29.97fps interlaced (NTSC only), 25fps interlaced (PAL/SECAM only)
  - displayed - 29.97 (NTSC) and 25fps (PAL/SECAM)
- Color space
  - color primaries and transfer characteristics
    NTSC - 4 (ITU-R BT.470 M) or 6 (SMPTE 170 M)
    PAL - 5 (ITU-R BT.470 B or G)
  - matrix coefficients (RGB to YCbCr)
    5 (ITU-R BT.470 B or G) or 6 (SMPTE 170 M)
- Audio sampling rate - 48KHz (MPeg audio only, other audio formats have their own rates)
- Aspect ratio - either 4:3 or 16:9
- GOP max
  - MPeg-1 18 frames (NTSC), 15 frames (PAL/SECAM)
  - MPeg-2 36 fields (NTSC), 30 fields (PAL/SECAM)

  note: typical rate is .5 seconds, with one GOP per VOBU
  note: GOP's are NOT optional
- pack size - fixed at 2048 bytes, system stream level (one DVD sector)
- Low delay - not permitted

Extensions

- Private stream 1 (Audio and subpictures)
  - sub-stream 0x20 to 0x3f are subpictures
  - sub-stream 0x80 to 0x87 are audio (AC3, DTS, SDDS)
  - sub-stream 0xA0 to 0xA7 are LPCM audio
- Private stream 2 (Navigation) - see PCI packet and DSI packet

DVD-Video home

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1617
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

about me
homepage
blog
gpg, pgp
CV (french)
CV (english)
projects
Debian
VideoLAN
FLESSD
Elk Scheme
libcaca
zzuf
genethumb.sh
PWNtcha
MonsterZ
LMOS
git-bigfiles
more...
MPEG & DVD
DVD subtitles
MPEG-4 lecture
DeCSS distribution*
doc
SVN & CVS
gprof & pthreads
Debian project lecture
shorter version
Verisign dickheads
doubleclick filter
WTFPL / LPRAB
leisure
artwork
photos
music
DTC logos
pom
goatse
La Classe américaine
#LinuxFr fortunes*
links

DVD subtitles

As of January 9th, 2001, the latest version of this document can be found here :
http://sam.zoy.org/doc/dvd/subtitles/

Preamble

One of the last things we missed in DVD decoding under my system was the decoding of subtitles. I found i information on the web or Usenet about them, apart from a few words on them being run-length encoded in the DV FAQ.

So we decided to reverse-engineer their format (it's completely legal in France, since we did it on interoperabil purposes), and managed to get almost all of it.

Basics

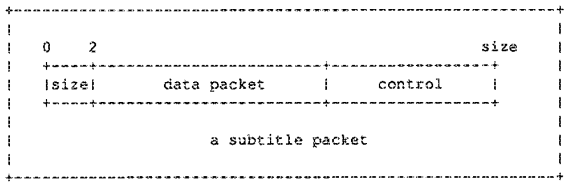
DVD subtitles are hidden in private PS packets (`0x000001ba`), just like AC3 streams are.

Within the PS packet, there are PES packets, and like AC3, the header for the ones containing subtitles have `0x000001bd` header. As for AC3, where there's an ID like `0x80+x`), there's a subtitle ID equal to `0x20+x`, where `x` is tl subtitle ID. There are 32 possible different subtitles on a DVD.

I'll suppose you know how to extract AC3 from a DVD, and jump to the interesting part of this documentation. Anyw you're unlikely to have understood what I said without already being familiar with MPEG2.

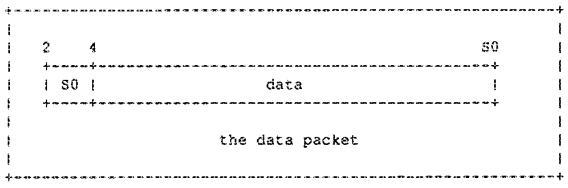
The data structure

A subtitle packet, after its parts have been collected and appended, looks like this :



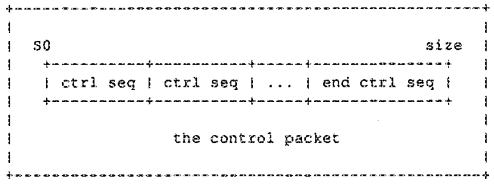
`size` is a 2 bytes word.

Here is the structure of the data packet :



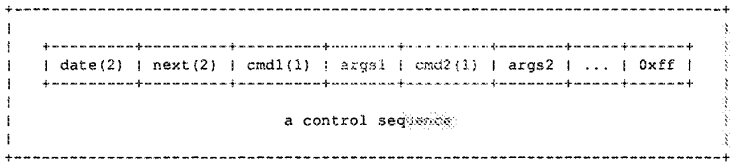
`S0`, the data packet size, is a 2 bytes word.

Here's the structure of the control packet :



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1618
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Here is the structure of a control sequence :



A control sequence starts with a date coded on 2 bytes, then an offset to the next control sequence coded on 2 byte If the offset to the next control sequence equals the offset of the current control sequence, it means we are on the la control sequence.

The data in a control sequence after the offset consists of one byte long commands followed by argumer depending on the command. The last byte is always `0xff`, which is actually a command without arguments telling v reached the end of the control sequence.

Control sequence commands

Here are the control sequences I know of. I know there are many more, to control subtitle fading and such things, bu didn't find information on them and couldn't reverse-engineer any because I actually didn't find any.

- `0x00` (*force displaying*) :
  this command takes no argument and is used to tell the decoder it has to display the subtitle. I suppose son other subtitles have commands which tell them to only display subtitles when the user clicks on the mouse, f instance.
- `0x01` (*start date*) :
  this command does not need an argument, since there is already a date information in the control sequence tells the decoder the delay before it has to display the subtitle (the decoder already knows the PES pack date from its PTS, the delay is in 100th of a second).
- `0x02` (*stop date*) :
  see the explanations for the start date. This command tells the decoder when to stop displaying the subtitle.
- `0x03****` (*palette*) :
  this command has four one nibble-long arguments, giving the palette information. Subtitles are encoded ir colours, but the palette is 16 colours-wide.
- `0x04****` (*alpha channel*) :
  this command has four one nibble-long arguments, giving the alpha channel information for each colour.
- `0x05************` (*coordinates*) :
  this command has four three nibble-long arguments, giving the coordinates of the subtitle on the screen : x1, x y1, y2. x1 is the first column, x2 is the last column, y1 is the first line, y2 is the last line. Thus the subtitle's size (x1-x2+1) x (y1-y2+1)
- `0x06********` (*RLE offsets*) :
  this command has 2 two-bytes-long arguments, respectively the offset of the first graphic line, and the offset the second one in the RLE data (the graphics are interlaced, so it helps a lot).
- `0xff` (*end command*) :
  this command has no argument and tells the decoder it reached the end of the command sequence.

Control packet decoding example

```
00000a0c0103023104ff0050002cf00223e00000604e9ff00830a0c02ff
```

Let's decode this sample control packet.

The first control sequence is :

```
(0000) (0a0c) (01) (03 0231) (04 0ff0) (05 0002cf00223e) (06 000604e9) (ff)
```

We can deduce from this that the effect date is zero 100th of a second after the PES packet's time (`0000`) and th the next control sequence will be at offset `0a0c`.

Then we learn that the sequence is a display sequence (`01`), that colour 0 is 0, colour 1 is 2, colour 2 is 3, and colo 3 is 1 (`03 0231`), that colours 0 and 3 are transparent, and colours 2 and 3 are opaque (`04 0ff0`). Then the `0002cf00223e` sequence tells us that the first column is 0x000, the last one is 0x2cf, the first line is 0x002, and the la line is 0x23e. Thus the subtitle's size is 0x2d0 x 0x23d. The `06 000604e9` means that the first encoded image sta at offset 0x006, and the second one starts at 0x04e9. Eventually, the `ff` stuff tells us we reached the end of t control sequence.

The second control sequence is :

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1619
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
(0093) (0a0c) (02) (ff)
```

We can deduce from this that the effect date is 1.47 seconds after the PES packet's time (`0x0093` = 147) and th the next control sequence will be at offset `0a0c`. But since this is precisely offset `0a0c`, we know it's the last conti sequence.

This control sequence just tell us it is a stop display sequence (`01`), and finishes with the `ff` byte.

Decoding the graphics

The graphics are rather easy to decode (at least, when you know how to do it).

The picture is interlaced, for instance for a 40 lines picture :

```
line 0  ----------------#----------
line 2  ------#--------------------
                  ...
line 38 ------------#--------------
line 1  ------------------#-------
line 3  --------#-----------------
                  ...
line 39 ------------#------------
```

When decoding you should get :

```
line 0  ----------------#----------
line 1  ------------------#-------
line 2  ------#--------------------
line 3  --------#-----------------
                  ...
line 38 ------------#--------------
line 39 ------------#------------
```

If the displaying resolution is low, you can choose to only display even lines, for instance.

The pixels are run-length encoded. The one byte values are :

```
one byte values : 0xf 0xe 0xd 0xc 0xb 0xa 0x9 0x8 0x7 0x6 0x5 0x4
two byte values : 0x3* 0x2* 0x1*
 3 bytes values : 0x0f* 0x0e* 0x0d* 0x0c* 0x0b* 0x0a* 0x09* 0x08* 0x07* 0x06* 0x05* 0x04*
 4 bytes values : 0x03** 0x02** 0x01** 0x000*
```

`*` stands for any nibble. Once a sequence X of this alphabet has been read, the pixels can be displayed : (X>>2) the number of pixels to display, and (X & 0x3) is the colour of the pixel. For instance, `0x23` means `0x23>>2` pixels colour `0x23&0x3`, which is actually 8 pixels of colour 3.

`000*` has a special meaning : it's a fill line / carriage return command. The decoder should do a carriage return wh reaching the end of the line, or when encountering this "000-" sequence. But with this sequence, it should also fill t rest of the line with the appropriate colour (X & 0x3). After a carriage return, the parser should be byte-aligned, one nibble might have to be skipped.

After a carriage return, the parser should be byte-aligned, so one nibble might have to be skipped, and it should re a line on the other interlaced picture, and swap like this after each carriage return.

Code

Misc information

There is no colour information stored within the subtitle packet, they are defined elsewhere in the IFO file. I will p some information about it here when I have some time.

I don't know what are the other control sequences.

Credits

Thanks to Michel Lespinasse <walken at via dot ecp dot fr> for his great help on understanding the RLE stuff, and all the ideas he had.

Thanks to mass (David Waite) and taaz (David I. Lehn) from irc at openprojects.net for sending me their subtitles.

Other contributors include Bob Ives" <bob at rebelact dot com> for pointing a small error.

Changes

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1620
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

- January 16th, 2000 : changed "x0" and "x1" to "S0" and "S1" to make it less confusing.
- January 16th, 2000 : added David Waite's and David I. Lehn's name.
- January 16th, 2000 : added the 'changes' section.
- January 9th, 2000 : fixed a small error in the SSO* explanation.

Sam Hocevar $Id: index.html 555 2004-09-17 13:03:02Z sam $Host: 98.189.197.2

valid XHTML 1.1* & CSS*

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1621
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

© ISO/IEC 2004 — All rights reserved

INTERNATIONAL ORGANISATION FOR STANDARDISATION

ORGANISATION INTERNATIONALE DE NORMALISATION

ISO/IEC JTC1/SC29/WG11

CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 **N6214**

February 2004

Systems

Title: Final Committee Draft of MPEG-4 streaming text format

INFORMATION TECHNOLOGY

Coding of audio-visual objects — Part 17: Streaming text format

Document type: International Standard
Document subtype:
Document stage: (40) Enquiry
Document language: E

STD Version 2.1

ISO/IEC JTC 1/SC 29

Date: 2004-02-18

ISO/IEC FCD 14496-17

ISO/IEC JTC 1/SC 29/WG 11

Secretariat:

Information technology — Coding of audio-visual objects — Part 17: Streaming text format

Élément introductif — Élément central — Partie 17: Titre de la partie

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

© ISO/IEC 2004 — All rights reserved

Contents

Page

Foreword v
Introduction v
1 Scope ..... Error! Bookmark not defined.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1625
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14496-17 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 29, *Coding of Audio, Picture, Multimedia and Hypermedia Information.*

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

— *Part 1: Systems*
— *Part 2: Visual*
— *Part 3: Audio*
— *Part 4: Conformance testing*
— *Part 5: Reference sofware*
— *Part 6: Delivery Multimedia Integration Framework (DMIF)*
— *Part 7: Optimized software for MPEG-4 tools*
— *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*
— *Part 9: Reference hardware description*
— *Part 10: Advanced Video Coding (AVC)*
— *Part 11: Scene description and application engine*
— *Part 12: ISO media file format*
— *Part 13: IPMP extensions*
— *Part 14: MP4 file format*
— *Part 15: AVC file format*
— *Part 16: Animation Framework eXtension (AFX)*
— *Part 17: Streaming text format*

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1626
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Introduction

This Specifications defines text streams as a concatenation of text access units. The format of text streams and text access units is specified, as well as signaling and decoding of text streams.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1627
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Information technology — Coding of audio-visual objects — Part 17: Streaming text format

1. Scope

This document specifies ISO/IEC International Standard ISO/IEC 14496-17 Streaming Text.

2. Normative references

The following Standards contain provisions that, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All Standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent edition of the Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

All references in this subclause were correct at the time of approval of this International Standard. The provisions of the referenced specifications, as identified in this subclause, are valid within the context of this IS. The reference to a specification within this IS does not give it any further status within ISO/IEC; in particular, it does not give the referenced specification the status of an International Standard.

- 3GPP TS 26.245: 2003, *Timed text format (Release 6)*

3. Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

3.1 **Text access unit**: an individually accessible portion of data within a text stream. Each text access unit contains the coded representation of text data corresponding to one text sample. For presentation, the text access unit can be associated with a single time stamp.

3.2 **Text modifier**: data within a text sample, specifying a modification to the presentation of the text string within that text sample.

3.3 **Text sample**: text data of a specified format, consisting of a text string, optionally followed by one or more text modifiers

3.4 **Text string**: data within a text sample, representing a string of characters encoded using UTF-8 or UTF-16.

3.5 **TTU[j]**: a Timed Text Unit is a syntactical sub-structure of a text access unit for carriage of text access units across networks, whereby by its index j identifies which type of data from a text access unit is carried.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1628
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4. Text stream format

The text stream format defined in this Specification is derived from the timed text format defined by 3GPP for timed text downloaded in a 3GP file, as specified in 3GPP TS 26.245. The 3GPP timed text format defines text samples and sample descriptions. This Specification defines how to partition such text samples and sample descriptions into so-called TTUs, Timed Text Units, and how to concatenate them into text access units. TTUs provide a flexible structure that allows for easy and convenient adaptation to the various transport layers. For each transport layer the most suitable TTU structure can be chosen. If so desired, text access units can be re-partitioned into TTUs for most effective adaptation to other transport systems. Partitioning in TTUs is defined so, that 3GPP extensions are possible, but also extensions defined in future versions of this Specification. See figure 1.

A text stream is a concatenation of text access units. During transport typically a time stamp is assigned to each text access unit to indicate the intended time to present the text access unit on the display. The text is presented within a text box, the position of which can be controlled by parameters within the text stream.

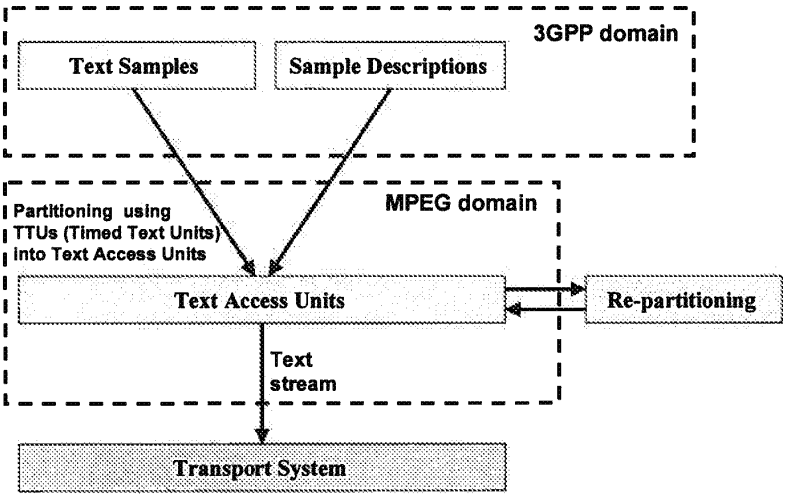


Figure 1. 3GPP defined text samples and sample descriptions and MPEG defined text access units.

4.1 Text samples

Each text sample represents text data and consists of a text string, optionally followed by one or more text modifiers, fully consistent with the TextSample format specified in 3GPP TS26.245. The text string represents the text to be displayed during a certain period, while the modifiers carry the changes that are to be applied to the text string during the time that the text is displayed, such as text colour changes synchronized with a song for a Karaoke application.

Each text string starts with 16 bit length field that specifies the length of the text string in bytes; note that this does not give the character count, as a character can vary in length from, in general, 1 to 6 bytes. The length of the text modifiers is not specified, as it is assumed that this information is derived from the total text sample length provided by means external to the text sample.

Global information about a text sample is provided by a sample description, for example about font(s) to be used, about the positioning of the text within the text box, the background colour of that text box, etc. Multiple sample descriptions are allowed; to each sample description (SD) an index is assigned and to each text sample the index of the applicable sample description is associated. A sample description will typically apply to multiple text samples; however, to each text sample exactly one sample description applies.

The relationship between sample descriptions, text samples, a text string and text modifiers is depicted in figure 2.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1629
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

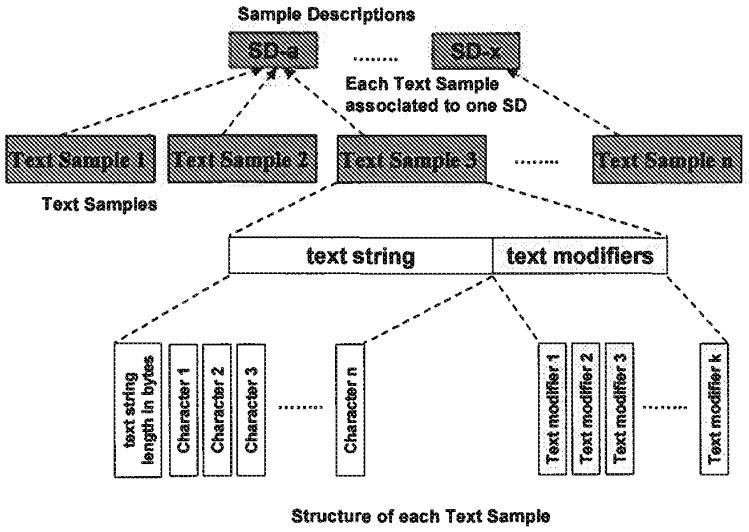


Figure 2. Sample Descriptions, Text Samples and their structure

4.2 Text access units

Each text access unit carries a text sample. Each text sample is to be presented during a certain period of time, specified by duration information contained in the text access unit that carries the text sample. At the instant in time that the presentation of a text sample ends, the next text sample starts being presented. To allow for periods of time without a text sample being presented, for instance between two subtitles, an "empty" text sample is defined, which is a text sample without data. If an "empty" text sample is immediately preceded by a non-"empty" text sample, then the "empty" text sample may be included in the same text access unit as the non-"empty" text sample.

Typically, a text sample is small, around 100 – 200 bytes, and often much smaller than the size of the packets that carry the text data across a transport network. It is therefore expected that transport systems will often aggregate multiple text samples into one such packet. On the other hand, text samples can also be large, for example when scrolling horizontal text at the bottom of the screen, in which case fragmentation of text samples may be required prior to transport. So as to achieve maximum commonality among text transport systems, a flexible and transport generic data structure is defined for containing a text sample within a text access unit. This data structure allows for a simple and generic network adaptation of text access units, and is implemented by means of so-called Timed Text Units, TTUs, of several types, each referred to as TTU[j]. The format of TTUs is defined in this document, as well as some general requirements for their use, while the actual usage of TTUs is left to the specifications for the applied transport means. However, as TTUs support text access unit fragmentation in a generic and error resilient manner, transport systems should use TTUs for fragmentation instead of transport specific aggregation and fragmentation tools.

Each text access unit is defined to consist of one or more TTUs. Five different type of TTUs are defined, one for carriage of a complete text sample, three for carriage of text sample fragments, and one for carriage of a complete sample description, while three types are reserved for future use. Because sample descriptions are small, there is no support for carriage of sample description fragments. Each TTU type is identified by its index j and referred to as TTU[j]. The following TTU types are defined:

- TTU[0]; this TTU type is reserved for future use;
- TTU[1]; this TTU type is capable of carrying one complete text access unit;
- TTU[2]; this TTU type is capable of carrying one fragment of a text string from a text sample;
- TTU[3]; this TTU type is capable of carrying the first text modifier fragment from a text sample;
- TTU[4]; this TTU type is capable of carrying a non-first text modifier fragment from a text sample;
- TTU[5]; this TTU type is capable of carrying one complete sample description;

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1630
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

- TTU[6] and TTU[7]; these TTU types are reserved for future use.

Figure 3 depicts the options to construct TTUs from text samples and sample descriptions. Each complete sample description is contained in one TTU[5]. Each text sample that is not fragmented, is contained in one TTU[1]. When a text sample is fragmented, then the text string is separated from the text modifiers for carriage in different TTUs. Each text string fragment is contained in one TTU[2], so that there are as many TTU[2]s as text string fragments. Note however that the text string itself may not be fragmented, in which case there is only a single text string fragment carried in one TTU[3]. The first text modifier fragment is contained in one TTU[3]. Hence, TTU[3] signals the start of the modifiers. Each subsequent text modifier fragment, if any, is contained in one TTU[4].

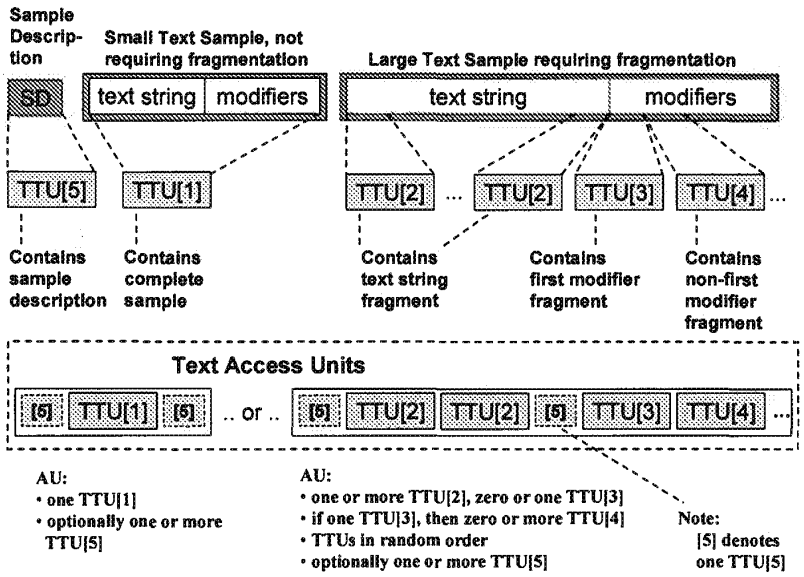


Figure 3. Carriage of Sample Descriptions and Text Samples in TTU[j]s, and construction of Text Access Units by means of TTUs.

In figure 3, also the basics of the construction of text access units by means of TTUs are depicted. If a text sample is not fragmented, then a text access unit contains the TTU[1] that carries the entire text sample. If the text sample is fragmented, then the text access unit contains one or more TTU[2]s, each with a fragment of the text string. If the text sample also carries modifiers, then the text access unit carries one TTU[3] and, depending on the size of the modifiers, zero or more TTU[4]s. In addition, each text access unit contains optionally one or more TTU[5]s, each carrying a sample description that may apply to this or future text samples in the text stream.

An empty text sample, that is a text sample without data (hence without a text string and modifiers), may be used to terminate the display of a text string prior to the start of the display of the next text sample in the stream. Such empty text sample is carried in a TTU[1], and may be included in the same access unit as the terminated text sample. Note that figure 3 does not illustrate how to include an empty text sample.

The order of TTUs within a text access unit is random within some constraints. When text samples are fragmented, it is allowed to repeat a fragment that is considered more important than others, and to transmit modifiers prior to text string fragments, which may be useful for large text strings. For ordering TTUs within a text access unit, the following constraints apply:

- When a sample description is provided in-band, then a sample description shall be included in the stream prior to any text sample that utilizes that sample description. This means that an in-band sample description, carried in a TTU[5], shall precede any TTU[1], TTU[2], TTU[3] and TTU[4] that references that sample description. Note that such TTU[5] may be carried in the same text access unit, or in one or more preceding text access unit(s).
- If an empty text sample is used to terminate the display of the text sample in a text access unit, then any other TTU[1], TTU[2], TTU[3] and TTU[4] in this text access unit shall precede the TTU[1] that contains the empty text sample.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1631
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.2.1.1 Text Sample fragmentation constraints

A number of constraints shall be applied when fragmenting text samples for carriage in TTU[2], TTU[3] and TTU[4]. The following constraints shall be applied:

- The first fragment of a text string shall commence with the 16 bit field that specifies the length in bytes of the entire non-fragmented text string;
- Each text string shall be fragmented on character boundary, so that each fragment contains complete character codes only, so as to ensure that each received text fragment can be decoded even in the case that a previous fragment gets lost. As a consequence, text string fragmentation requires knowledge of UTF-8/-16 formats to determine character boundaries.

It is not required to fragment text modifiers on modifier boundary; consequently, when a modifier fragment is lost, receivers will typically not be capable of decoding any remaining fragments.

4.2.2 TTU structure

Each TTU starts with an 8 bit TTU header; this header carries the type of TTU, signals whether characters in the text string of the text sample are encoded using UTF-8 or UTF-16, and signals whether the TTU header is followed by a field that indicates the length (size) of the TTU in bytes. This Specification defines some requirements for the presence and usage of the TTU length field; in addition, transport specific requirements not defined in this Specification may apply. The remaining fields after the TTU header are TTU type specific and described in the following sub-clauses.

4.2.2.1 TTU[1] signaling

A TTU[1] carries a complete text sample; the TTU header and the TTU length field, if present, are followed by fields that provide the following information:

- the index of the applicable sample description; and
- the duration of the text sample.

Carriage of an empty text sample in a TTU[1] is signaled by the presence of the TTU length field encoded with a specific value (see clause 4.3).

4.2.2.2 TTU[2] signaling

A TTU[2] carries a text string fragment from a text sample. The TTU header and the TTU length field, if present, are followed by fields that signal:

- the index of the applicable sample description;
- the duration of the text sample;
- the total length of the original non-fragmented text sample; this information allows the receiver to allocate buffer space for the decoding of the text sample; this information is provided for each text string fragment so as to also allow for buffer space allocation after losing one or more text string fragments;
- the total number of text sample fragments; this number shall include each text string fragment and each modifier fragment.
- the sequence number of the fragment contained in this TTU[2]; the sequence number of the first text sample fragment is equal to 0, and for each subsequent fragment the sequence number is incremented by one, so that in case of N fragments, the sequence number of the last fragment is equal to N-1. A text sample can be fragmented in at most 16 different fragments.

4.2.2.3 TTU[3] and TTU[4] signaling

In a TTU[3] and TTU[4], the first and a non-first modifiers fragment is carried, respectively; the TTU header and the TTU length field, if present, are followed by fields that signal:

- the total number of text sample fragments (see clause 4.2.2.2);
- the sequence number of the fragment contained in this TTU[2] (see clause 4.2.2.2);.

Note that the index of the applicable sample description, the duration of the text sample, and the total length of the fragmented text sample are not signaled, as modifiers fragments apply to text strings for which this information is already signaled.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1632
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.2.2.4 TTU[5] signaling

A TTU[5] carries a sample description. The TTU header and the TTU length field, if present, are followed by a field that signals:

- the index of the applicable sample description.

4.2.3 Constraints for using Sample Descriptions

Within text streams 256 index values are available to unambiguously identify a sample description. The values 0 and 255 are reserved. Sample descriptions can be provided either in-band or out-of-band. The index values in the inclusive range between 1 and 127 are available for in-band usage, while values between 128 and 254 (inclusive) can be used to provide sample descriptions out-of-band. See table 1.

Table 1 Index values for Sample Descriptions.

Index value	Assignment
0	Reserved
1 – 127	Available for in-band usage
128 – 254	Available for out-of- band usage
255	Reserved

When sample descriptions are provided in-band, there may be a need to regularly update sample descriptions. For this purpose a wrap-up mechanism is defined for in-band sample descriptions, that allows to continuously update such sample descriptions from a range of 64 valid index values. Hence, at any point in time at most 64 in-band sample descriptions can be in use simultaneously. Each sample description with an index value not in the valid range is discarded from further use. Valid sample descriptions can be transmitted in-band as often as desirable for random access, without any impact on the range of invalid index values. However, when receiving a sample description with an index value from the invalid range, then the valid range is changed, so that the received sample description becomes valid. When a sample description is received with index value i not in the valid range, then the receiver shall discard each sample description with an index value in the range between (i+1)mod128 and (i+64)mod128, inclusive. For example, assume that the valid range is between 41 and 104 inclusive, and hence, that the values 105 to 127 and 1 to 40 are invalid values. When now the sample description with index value 114 is received, the receiver shall invalidate the index values 115 to 127 and 1 to 50, inclusive, so as to change the valid range to values between 51 and 114.

4.3 Syntax and Semantics

4.3.1 Text_access_unit()

```
text_access_unit(){
      for (i=0; i<N;i++){
            TTU[j]()
            }
      }
```

4.3.2 TTU[j]()

```
TTU[j](){
      reserved                                    3     bslbf
      UTF_16_flag                                 1     bslbf
      TTU_length_flag                             1     bslbf
      TTU_type                                    3     uimsbf
      if (TTU_data_length_flag==1){
            TTU_data_length          16     uimsbf
            {
      TTU_data[j]()
      }
```


© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1633
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

`UTF_16_flag` - a one bit flag, indicating in TTU[1] and TTU[2] the encoding used for the characters in the text string; when set to '1', UTF-16 is applied, when set to '0', UTF-8 is applied. For TTU[3], TTU[4] and TTU[5] this bit has no meaning and shall be ignored.

`TTU_data_length_flag` - a one bit flag, that indicates, when set to '1', the presence in this `TTU[j]` of the `TTU_data_length` field. This field shall be set to '1' in each TTU[5], and in each TTU[1] that carries an empty text sample.

`TTU_type` - a three bit integer that specifies the type of TTU; the TTU-type field is encoded with a value that is equal to the index j in TTU[j], according to the following table.

Table 2 TTU types

TTU_type value	TTU type	TTU usage
0	TTU[0]	Reserved for future use
1	TTU[1]	Carriage of complete text sample
2	TTU[2]	Carriage of text string fragment
3	TTU[3]	Carriage of first modifiers fragment
4	TTU[4]	Carriage of non-first modifiers fragment
5	TTU[5]	Carriage of a complete sample description
6	TTU[6]	Reserved for future use
7	TTU[7]	Reserved for future use

`TTU_data_length` - an optional 16 bit integer that specifies the total number of bytes of the TTU_data_length field and the immediately following `TTU_data[j]()`. In a TTU[1], carriage of an empty text sample is signaled by a `TTU_data_length` value of 6. See clause 4.3.4.

4.3.3 TTU_data[0]()

```
TTU_data[0](){
      for (i=0; i<N;i++){
            reserved                                  8      bslbf
            }
      }
```

4.3.4 TTU_data[1]()

```
TTU_data[1](){
      sample_index                                    8      uimsbf
      sample_duration                                 24     uimsbf
      whole_text_sample()
      }
```

`sample_index` - an 8 bit unsigned integer that specifies the index of the sample description that applies to the text sample contained in this TTU. The sample_index value shall unambiguously refer to one particular sample description that shall be present in the text stream prior to its use. The sample_index values 0 and 255 are reserved.

`sample_duration` - a 24 bit unsigned integer that specifies the intended duration of the presentation of the text sample in this TTU in units of the durationClock, specified in textConfig; see Clause 7.

`whole_text_sample()` - TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see Clause 7).

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1634
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.3.5 TTU_data[2]()

```
TTU_data[2](){
      sample_index                                8      uimsbf
      sample_length                               16     uimsbf
      sample_duration                             24     uimsbf
      total_number_of_sample_fragments            4      uimsbf
      number_of_this_sample_fragment              4      uimsbf
      text_string_fragment()
      }
```

sample_index - See sample_index semantics in clause 4.3.4

sample_length - a 16 bit unsigned integer that specifies the total number of bytes in the TextSample of which a fragment is carried in this TTU.

sample_duration - See sample_duration semantics in clause 4.3.4

total_number_of_sample_fragments - a 4 bit unsigned integer that specifies for the TextSample of which a fragment is carried in this TTU the total number of fragments in which that TextSample is split.

number_of_this_sample_fragment - a 4 bit unsigned integer that specifies the sequence number of the TextSample fragment that is carried in this TTU. When the TextSample is split into N fragments, then the sequence number reflects the logical order of the fragments, that is the first fragment has sequence number 0, and the last fragment sequence number N-1, while for each subsequent fragment the sequence number is increased by 1.

text_string_fragment() - a fragment of the text string from the TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see Clause 7).

4.3.6 TTU_data[3]()

```
TTU_data[3](){
      total_number_of_sample_fragments            4      uimsbf
      number_of_this_sample_fragment              4      uimsbf
      first_modifier_box_fragment()
      }
```

total_number_of_sample_fragments - See total_number_of_sample_fragments semantics in clause 4.3.5.

number_of_this_sample_fragment - See number_of_this_sample_fragment semantics in clause 4.3.5.

first_modifier_box_fragment() – the first fragment of the text modifiers from the TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see Clause 7).

4.3.7 TTU_data[4]()

```
TTU_data[4](){
      total_number_of_sample_fragments            4      uimsbf
      number_of_this_sample_fragment              4      uimsbf
      non_first_modifier_box_fragment()
      }
```


© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1635
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

total_number_of_sample_fragments - See total_number_of_sample_fragments semantics in clause 4.3.5.

number_of_this_sample_fragment - See number_of_this_sample_fragment semantics in clause 4.3.5.

non_first_modifier_box_fragment() - a non-first fragment of the text modifiers from the TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in the DecoderConfiguration (see Clause 7).

4.3.8 TTU_data[5]()

```
TTU_data[5](){
      sample_index                              8      uimsbf
      sample_description()
      }
```

sample_index - the index of the sample description that applies to the text sample contained in this TTU. The sample_index value shall unambiguously refer to one particular sample description that shall be present in the text stream prior to its use. The sample_index values 0 and 255 are reserved. The sample_index values between 128 and 254 inclusive are assigned to out-of-band provided sample descriptions and shall not be used for in-band provided sample descriptions. Similarly, the sample_index values between 1 and 127 inclusive are assigned to in-band provided sample descriptions and shall not be used for out-of-band provided sample descriptions.

The following shall apply to in-band provided sample descriptions. At any instant in time, at most 64 different sample_index values shall be in use for in-band sample descriptions, identified by a range of valid sample_index values. When a receiver starts the decoding of a text stream, the receiver has no knowledge of the valid range, but upon receiving the first sample description the receiver shall define the range of valid sample_index values as follows. If the first received sample has the sample_index value j, then the receiver shall define the sample_index values in the range between (j+1)mod128 and (j+64)mod128, inclusive, invalid and to shall consider all other sample_index values between 1 and 127 (inclusive) valid. When subsequent sample descriptions are received with a sample_index value in the valid range, then the valid range shall not be modified. However, when a sample description is received with index value i not in the valid range, then the receiver shall invalidate sample_index values in the range between (i+1)mod128 and (i+64)mod128, inclusive.Each sample description with an sample_index value assigned as invalid shall be discarded. See also table 3. Text samples in the text stream shall not reference any invalidated sample description. A sample description is valid for each subsequent text sample in the text stream until invalidated by another sample description.

Table 3 Sample Descriptions that are invalidated by the Sample Description with sample_index value i

Received sample_index value i in currently invalid range	New invalid range (inclusive)	New valid range (inclusive)
$1 \le i \le 63$	(i+1) – (i+64)	1 – i (i+65) – (127)
$64 \le i \le 127$	(i+1) – 127 1 – (i-64)	(i-63) – i

sample_description() - a sample description for a TextSample, referred to as TextDescription, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in the DecoderConfiguration (see Clause 7).

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1636
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.3.9 TTU_data[6]()

```
TTU_data[6]{
      for (i=0; i<N;i++){
            reserved                          8      bslbf
            }
      }
```

4.3.10 TTU_data[7]()

```
TTU_data[7]{
      for (i=0; i<N;i++){
            reserved                          8      bslbf
            }
      }
```

5. Signaling of a text stream

A text stream shall be signaled by a streamType value 0x0B and by an objectTypeIndication value of 0x08.

6. Text rendering

Text from a decoded text stream is rendered within a Text Box. The Text Box can be positioned within a Text Track; see 3GPP TS 26.245. How to render the text in the Text Box and how to position the Text Box within the Text Track is specified within the text stream by means of sample descriptions and/or text modifiers.

The width and the height of the Text Track shall be defined in the decoderConfiguration, textConfig, in units of square pixels. In textConfig it is also specified whether or not a text stream is intended to be positioned relative to a video stream. Note that a Text Track can also be positioned with respect to a video picture by means of information that is provided by external means, such as information contained in an MP4 file, or by an application. External information, if provided, takes priority over information provided within textConfig.

If the text stream is associated with a video picture, then textConfig specifies the video-width and video-height of that video picture. The video-width and video-height, as well as the position of the Text Track relative to the video picture are specified in units of the same square pixels as the width and the height of the Text Track, so as to enable scaling and positioning of the text with respect to video in a manner independent of the video resolution. This allows that the same text stream can be used with video streams coded at different resolutions and bitrates. The relative position of the Text Track with respect to the video picture is specified by the horizontal-offset and vertical-offset. These two parameters specify the horizontal and vertical distance in pixels between the top left pixel of the video picture and the top left pixel of the Text Box, respectively; see figure 4. It is allowed that the Text Box is partly or completely outside of the video picture. A positive value of horizontal-offset and vertical-offset indicate that top left pixel of the Text Box is right and below, respectively, of the top left pixel of the video picture. For example, in figure 4, both horizontal-offset and vertical-offset have a positive value. Accordingly, a negative value of horizontal-offset and vertical-offset indicates that the top left pixel of the Text Box is left and above, respectively, of the top left pixel of the video picture. Note that the square pixels units used in textConfig to specify the sizes of the Text Track and the video picture and their relative positions will typically have a different dimension than the actually coded video pixels. Hence if the video-width and video-height specify a picture with a size of 640*480, then the picture may actually be coded at 720*480 or 320*240 or any other resolution.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1637
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

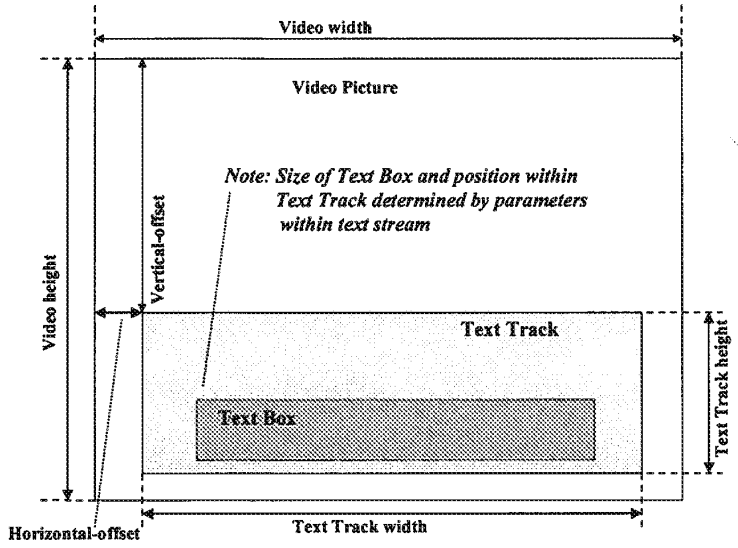


Figure 4. Size of the Text Track and the positioning of the Text Track with respect to a video picture.

7. DecoderConfiguration for Timed Text decoders

To decode a text stream, a receiver needs the decoderConfiguration for the text stream, as defined by TextConfig. The TextConfig provides information such as:

- the 3GPP format that the text stream is compatible with
- whether an MPEG defined extension is applied
- the profile and level of the text stream
- clock units used to encode text sample durations
- whether in-band sample descriptions are applied
- whether out-of-band sample descriptions are applied
- whether sample descriptions are carried in the textConfig
- optional information on the positioning of the text track relative to a video picture

7.1 Syntax

```
class TextConfig extends DecoderSpecificInfo : bit(8) tag=DecSpecificInfoTag {
   bit(8)  3GPPBaseFormat;
   bit(8)  MPEGExtendedFormat;
   bit(8)  profileLevel;
   bit(24) durationClock;
   bit(1)  contains-list-of-compatible-3GPPFormats-flag;
   bit(2)  sampleDescriptionFlags;
   bit(1)  SampleDescription-carriage-flag;
   bit(1)  contains-associated-video-information;
   bit(3)  reserved;
   bit(8)  layer;
   bit(16) text-track-width;
   bit(16) text-track-height;
   if (contains-list-of-compatible-3GPPFormats-flag) {
      bit(8) number-of-formats;
```

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1638
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
      bit(8) Compatible-3GPPFormat(number-of-formats);
      }
   if (SampleDescription-carriage-flag) {
      bit(8) number-of-SampleDescriptions;
      SampleDescription()(number-of-SampleDescriptions);  //as many SampleDescriptions,
      }                        each carried in a TTU[5], as number-of-SampleDescriptions
   if (contains-associated-video-information) {
      bit(16) video-width;
      bit(16) video-height;
      bit(16) horizontal-offset;
      bit(16) vertical-offset;
      }
   bit(8)  remaining-config-length;
   bit(8)  reserved(remaining-config-length);
}
```

7.2 Semantics

`3GPPBaseFormat` – one byte signaling the latest 3GPP format that this text stream is compatible with, coded as specified in table 4. If the text stream is also compatible with older versions of 3GPP specifications, then these formats can be listed in the textConfig by setting the `contains-list-of-compatible-3GPPFormats-flag`.

Table 4 – 3GPPBaseFormat

0x00 – 0x0F	Reserved
0x10	Timed Text as specified in 3GPP TS 26.245
0x11 – 0xFF	Reserved

`MPEGExtendedFormat` – one byte signaling the MPEG defined compatible extension of the 3GPPBaseFormat applied for this stream, coded as specified in table 5.

Table 5 – MPEGExtendedFormat

0x00 – 0x0F	Reserved
0x10	No extension applied
0x11 – 0xFF	Reserved

`profileLevel` – an 8 bit unsigned integer that specifies the profile and level of the text stream encoded as specified in table 6. See Clause 9 for profile and level constraints.

Table 6 profileLevel

0x00 – 0x0F	Reserved
0x10	Base profile, base level
0x11 – 0xFF	Reserved

`durationClock` – a 24 bit unsigned integer that specifies the frequency in Herz of the clock used to encode the `sample-duration` field in TTU[1] and TTU[2]. A value equal to 1000 signals a 1kHz clock, and that `sample-duration` field is encoded in units of 1 msec. Transport systems should assign time stamps to text access units using the same frequency as signaled by `durationClock` or an integer multiple thereof.

`contains-list-of-compatible-3GPPFormats-flag` – one bit that signals the presence of a list of one or more 3GPP formats older than the signaled format by `3GPPBaseFormat`, that the text stream is also compatible with.

`sampleDescriptionFlags` – two bits that signals whether SampleDescriptions are provided in-band, out-of-band or both, as specified in table 7.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1639
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Table 7 – sampleDescriptionFlags

00	Forbidden
01	No in-band, out-of band only
10	In-band only, no out-of band
11	Both in-band and out-of band

`SampleDescription-carriage-flag` – one bit that signals the presence of a list of SampleDescriptions in this textConfig. This bit shall be set to '0', if `sampleDescriptionFlags` is coded with the value '10'.

`contains-associated-video-information` – one bit that signals the presence of the `video-width, video-height, horizontal-offset, and vertical-offset` fields in this TextConfig. If some or all of this information is also provided by external means, then the externally provided information takes priority.

`layer` – unsigned integer specifying the proximity of the text track to the viewer. A higher value means closer to the user. A value of '0' indicates an undefined proximity. If the proximity is also provided by external means, then the externally provided information takes priority.

`text-track-width` – unsigned integer specifying the width of the text track in pixels.

`text-track-height` – unsigned integer specifying the height of the text track in pixels.

`number-of-formats` – unsigned integer specifying the number of immediately following `Compatible-3GPPFormat` fields. The value '0' is forbidden.

`Compatible-3GPPFormat` – one byte that signals a 3GPP formats older than the signaled format by `3GPPBaseFormat`, that the text stream is also compatible with, coded in the same way as the `3GPPBaseFormat` field, specified in table 4.

`number-of-SampleDescriptions` – unsigned integer that specifies the number of immediately following `SampleDescriptions()`.

`SampleDescription()` – one sample description carried in one TTU[5].

`video-width` – unsigned integer specifying the width of the video picture in pixels[1]

`video-height` – unsigned integer specifying the height of the video picture in pixels; see footnote 1.

`horizontal-offset` – signed integer specifying the horizontal distance in pixels (see footnote 1) between the left border of the video picture and the left border of the Text Box; a positive distance value indicates that the left border of the Text Box is right of the left border of the video picture

`vertical-offset` – signed integer specifying the vertical distance in pixels (see footnote 1) between the top border of the video picture and the top border of the Text Box; a positive distance value indicates that the top border of the Text Box is below the top border of the video picture

`remaining-config-data-length` – unsigned integer, specifying the number of immediately following `reserved` bytes in this textConfig.

[1] These are the square pixels used to specify the Text Box dimensions, not coded video pixels.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1640
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

8. Text stream decoding

A hypothetical decoder has been defined for decoding of text streams. The Hypothetical Text Decoder (HTD) consists of a TTU Decoder, a textSampleBuffer, an in-band-sampleDescriptionBuffer, an out-of-band-sampleDescriptionBuffer and a textDecoder; see figure 5. Input to the HTD is a text stream with a bitrate of R kb/s. The value of R, as well as the textSampleBuffer size, the in-band-sampleDescriptionBuffer size and the out-of-band-sampleDescriptionBuffer size is specified per profile and level; see Clause 9.

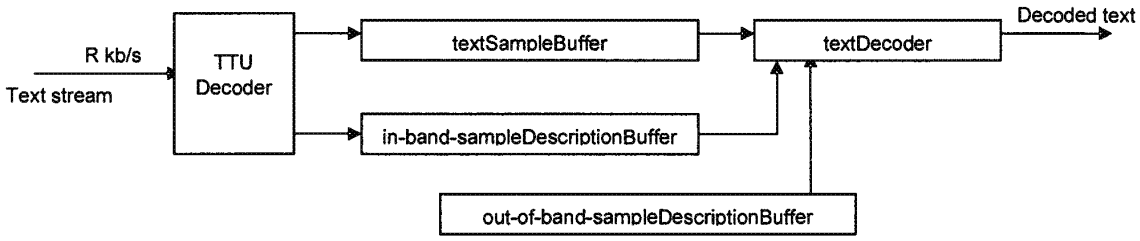


Figure 5 Hypothetical Text Decoder

In the HTD, each out-of-band sample description utilized by the text stream is provided prior to its usage by the text stream to the out-of-band-sampleDescriptionBuffer by means not defined by this Specification. The total size in bytes of all out-of-band sample descriptions used in the text stream shall not exceed the specified size of the out-of-band-sampleDescriptionBuffer.

At the input of the HTD, the TTU decoder processes each received TTU byte instantaneously. Contained data from text samples are transferred to the textSampleBuffer, while in-band provided sample descriptions are transferred to the in-band-sampleDescriptionBuffer. At any point in time, the total size in bytes of all valid in-band sample descriptions shall not exceed the specified size of the in-band-sampleDescriptionBuffer. Text streams shall be constructed so that each in-band sample description is completely present in the in-band-sampleDescriptionBuffer at the instant in time that a text sample that references that sample description is decoded.

If the textSampleBuffer is not full, then the bytes of the text stream enter the TTU decoder at a bitrate R; if the textSampleBuffer is full, no bytes enter the TTU decoder. The byte in the text stream subsequent to the text sample byte that causes the textSampleBuffer to be full does not enter the TTU decoder as long as the textSampleBuffer remains full. At the instant in time that the presentation of a text sample is to start, the text sample is instantaneously removed from the textSampleBuffer, decoded and presented. Underflow of the textSampleBuffer occurs if a text sample has not (yet) fully entered the textSampleBuffer at the instant in time that the text sample is to be presented. Text streams shall be constructed so, that in the hypothetical text decoder no underflow occurs:

- The size of each text sample shall be smaller than or equal to the specified size of the textSampleBuffer;
- Taking the fullness of the textSampleBuffer into account, there shall be sufficient time to deliver the complete text sample to the textSampleBuffer at the specified bitrate prior to its decoding.

Note: In the extreme case that the size of the previous text sample is equal to the size of textSampleBuffer, then the next text sample is to be delivered during the duration of the previous one. To avoid underflow in this extreme case, the encoder has to ensure that the duration between the previous non-empty sample and the next non-empty sample N (note that there may be one or more "empty" samples in-between) is larger than the size of the non-empty sample N divided by the applicable bitrate R.

9. Profiles and Levels for text streams

For text streams this specification defines only a single profile and a single level: the base profile and the base level. Other profile and levels may be defined in future. The profile and level are signaled in textConfig, see Clause 7 and table 6.

In each text stream that conforms to the base profile, each tool specified by 3GPP in 3GPP TS 26.245 may be used. In each text stream that complies with the base level, the HTD parameter values defined in table 8 shall apply. Each decoder that complies with the base profile and the base level shall be capable of decoding each text stream that conforms to the base profile and the base level.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1641
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Table 8 HTD parameter values for the Base Level

HTD Parameter	Value
Bitrate R	10 kb/s
Size of textSampleBuffer	8192 Byte
Size of in-band-sampleDescriptionBuffer	4096 Byte
Size of out-of-band-sampleDescriptionBuffer	4096 Byte

Editorial note: the HTD model and the above numbers need further consideration

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1642
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Annex A

Font Referencing

A.1 Font references in text streams

Timed Text streams may reference fonts in the receiver. These fonts may be residently available in the receiver, or downloaded or streamed to the receiver. To ensure that the desired fonts in the receiver are referenced correctly, the font name identification in textData() has to uniquely reference the name of the desired font in the receiver.

In an MPEG-4 systems context, when a font stream is used to supply fonts for a text stream, that text stream shall indicate that dependency by using the dependsonES_ID field in the ES_Descriptor of the text stream, containing the ES_ID of the font stream. Other systems contexts should indicate that dependency in an appropriate way.

In text streams which have an associated font stream, when a specific font instance in the font stream is to be used for a text access unit, the font name shall use the appropriate naming structure, which for MPEG-4 font streams is defined in 14496-18 section 5.5.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1643
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

THE FREE LIBRARY
23,424,620 articles and books

Periodicals Literature
Search
Keyword Title Author Topic

IBM Spearheading Intellectual Property Protection Technology for Information on the Internet; Cryptolope Containers Have Arrived.

How To Patent Your Idea
www.inventionhome.com
3 Easy Steps To Patent & Make Money Get A Free Patent & Invention Kit

Intellectual Property MS
iit.edu/Intellectual-Property
Online. Earn your Master's Degree in Intellectual Property Management

Link/Page Citation

SAN JOSE, Calif.--(BUSINESS WIRE)--May 1, 1996--IBM today announced the commercial availability of Cryptolope(i) containers, secure packaging for digital information, enabling Internet users to buy and sell content securely over the Internet.

Cryptolope technology is a fundamental part of IBM's new infoMarket service (http://www.infomarket.ibm.com), an important implementation of IBM's network computing strategy. IBM infoMarket combines a sophisticated search engine with an unmatched body of content, secure container technology and IBM's expertise in transaction and billing management, to provide a framework for secure electronic commerce.

"The Internet is a natural conduit for the transmission of information, but publishers previously have been afraid to put their valuable content on the 'Net for fear that it may be tampered with or that they will not receive compensation for their works," said Jeff Crigler, vice president, IBM infoMarket. "The IBM infoMarket service provides the technology to address those needs."

IBM infoMarket also announced a number of new content providers today. With a growing number of content and technology providers now aligned with the service, including Reuters New Media, Jupiter Communications and CMP Publications, IBM infoMarket is a leading provider of content on the Internet. Users benefit from IBM infoMarket's ability to search through commercial content – some of which was previously unavailable on the World Wide Web – as well as USENET newsgroups and other leading content sources, a capability unequaled by any other search engine in the market today.

Tyco Integrated Security
tycois.com
Over 100 Years of Experience in Commercial Security. Learn More !

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1644
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Once targeted information is found, commercial content will be delivered in Cryptolope containers, accompanied by a content abstract that provides users with the essential product knowledge they need to make a buying decision. The content abstract may also include the content source, summary, author, last update, size, and price, as well as any unique terms of sale. Once the user has decided to open the contents of a Cryptolope container, a transparent digital key is issued unlocking the material contained within.

To view a free document, the user clicks on the article and the information appears on the desktop. To view priced content, the user agrees to any unique terms of the Cryptolope container that are prepared by the rights holder and contained in the content abstract.

Users of IBM infoMarket will be able to retrieve and download Cryptolope containers from the infoMarket service Web site using an IBM infoMarket helper application, initially for Netscape(ii) 1.1 or greater, and Windows(ii) 3.1 or Windows 3.1.1. Windows 95, OS/2 and Mac/OS versions are planned.

Superdistribution Can Provide Incremental Revenue

A user can forward a Cryptolope container to others allowing content providers to take advantage of "superdistribution", a new business model that, if permitted by the publisher, allows the customer to send the Cryptolope to others who may be interested in its contents, thus facilitating the first phase of superdistribution. Publishers can now take advantage of the Internet as a valuable business medium by which to sell their information, over and over again.

The IBM infoMarket service has created a clearinghouse for commercial content providers to track the payment, delivery and authenticity of contents held within a Cryptolope.

The content provider sets marketing and pricing guidelines for the use of commercial content or technology services. This allows the owners of this valuable information to have greater control over distribution, appearance and price of their content.

Wide-Area Search of Distributed Data

The search feature of the IBM infoMarket service can take a single search request and simultaneously search multiple, disparate, and distributed information sources returning relevance ranked results. IBM infoMarket's powerful search capabilities allow users to simultaneously investigate the broad scope of the Internet as well as authorized, private commercial content. Keyword and Boolean search requests allow for more complex information queries using OR, AND or NEAR connectors. These search capabilities allow for manual selection of sources that will best contain the desired information to meet users needs.

How to Patent My Idea?
www.patentlybrilliant.com
Patent Before Someone Else Does! Get Your Free Inventor Kit Today.

IBM infoMarket also provides a feature which enables users to search by source. Users can pick an individual source, all or a group of sources in a category, or select all sources. Additionally, search results can be received in brief or with more details. Detailed results provide the user with the content abstract, which may include source, author, excerpt, date, size, and pricing information.

Rich News and Business Information

New content and technology providers for IBM infoMarket include the following:

-- CMP Publications(ii) provides publishing, marketing, and information services to the high-technology market

-- Excalibur Technologies Corporation(ii) provides the first set of tools for creating retrieval solutions for text, images, and other forms of digital data;

-- International Services, Inc.(ii) provides international trade leads, language translation services, and international credit

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1645
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

reports

-- Jupiter Communications, LLC(ii) research, consulting, and publishing information emerging consumer online and interactive technologies

-- M.A.I.D(ii) established Profound, Inc., the New York-based subsidiary of Market Analysis and Information Database, Inc.

-- Market Guide(ii) provides high quality, fundamental information on over 8,000 publicly traded companies to the professional brokerage, institutional research and individual investor marketplaces.

-- Online Inc.(ii) publishes how-to magazines and books aimed at users of online databases, CD-ROMs, multimedia, and the Internet

-- Thunderstone Software(ii) provides intelligent concept searching and retrieval technologies

-- Vickers Stock Research Corporation, Inc.(ii) provides information on exactly what stocks insiders are buying and selling

IBM

IBM has been a pioneer in the development of the Internet. It designed special hardware and software, and provided operations support for key networks that make up today's Internet, including the Internet's primary backbone, the NSFnet. IBM also helped form Advanced Network & Services (ANS), the organization which had been responsible for managing and servicing the NSFnet from its inception until the beginning of 1995.

IBM offers hardware, software, consulting and services to help customers take advantage of the potential of the Internet. Whether addressing an individual need or creating a total enterprise solution, IBM provides the expertise required to plan, design, implement and operate an Internet solution for businesses.

To learn more about IBM's Internet activities, visit the IBM Internet home page at: http://www.internet.ibm.com or the IBM Corporation home page at http://www.ibm.com. -0-

Note to Editors:

(i) Indicates trademark or registered trademark of International Business Machines Corporation.

(ii) Indicates trademark or registered trademark of respective company.

Windows is a trademark of Microsoft Corporation

CONTACT: IBM

Mike King, 914/766-1119

mikeking@gemini.ibm.com

or

TSI for IBM

Ellen Cooper, 212/696-2000, ext. 258

ecooper@tsipr.com

COPYRIGHT 1996 Business Wire
No portion of this article can be reproduced without the express written permission from the copyright holder.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1646
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Copyright 1996, Gale Group. All rights reserved. Gale Group is a Thomson Corporation Company.

Please bookmark with social media, your votes are noticed and appreciated:

+1 0

Like Share 0

Reader ratings: **0 [0 vote(s)]**

Reader Opinion

Title:

Comment:

Submit

Article Details

Printer friendly Cite/link Email Feedback

Publication: Business Wire

Date: May 1, 1996

Words: 1082

Previous Article: Toshiba Expands Strategic Accounts, Sales.

Next Article: Magic Software Enterprises announces alliance with IBM Corp.'s AS/400 Division: Magic to develop AS/400 server product; IBM-Magic Solution to rapidly...

Topics: Computer software industry
Product introduction
DBMS software
Software industry
Product introduction

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1647
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Related Articles

The superdistribution model.

August Wilson's call: notes from editors.

The patent files.

Webscapades.

It's all mine: protect your intellectual property. (On The Shelf).

PACT COULD HURT DAIRY FARMERS.

SDDC works with the Department of Homeland Security in Korea.

Sneak attack: plan to protect U.S. ports homes in on contraband.

The Free Library > Business and Industry > Business > Business Wire > May 1, 1996
The Free Library > Business and Industry > Business, International > Business Wire > May 1, 1996
The Free Library > Date > 1996 > May > 1 > Business Wire

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1648
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Terms of use | Copyright © 2014 Farlex, Inc. | Feedback | For webmasters

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1649
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

© ISO/IEC 2004 — All rights reserved

INTERNATIONAL ORGANISATION FOR STANDARDISATION

ORGANISATION INTERNATIONALE DE NORMALISATION

ISO/IEC JTC1/SC29/WG11

CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 **N6214**

February 2004

Systems

Title: Final Committee Draft of MPEG-4 streaming text format

INFORMATION TECHNOLOGY

Coding of audio-visual objects — Part 17: Streaming text format

Document type: International Standard
Document subtype:
Document stage: (40) Enquiry
Document language: E

STD Version 2.1

ISO/IEC JTC 1/SC 29

Date: 2004-02-18

ISO/IEC FCD 14496-17

ISO/IEC JTC 1/SC 29/WG 11

Secretariat:

Information technology — Coding of audio-visual objects — Part 17: Streaming text format

Élément introductif — Élément central — Partie 17: Titre de la partie

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1652
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Contents

Page

Foreword v
Introduction v
1 Scope ..... Error! Bookmark not defined.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1653
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14496-17 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 29, *Coding of Audio, Picture, Multimedia and Hypermedia Information*.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

- *Part 1: Systems*
- *Part 2: Visual*
- *Part 3: Audio*
- *Part 4: Conformance testing*
- *Part 5: Reference sofware*
- *Part 6: Delivery Multimedia Integration Framework (DMIF)*
- *Part 7: Optimized software for MPEG-4 tools*
- *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*
- *Part 9: Reference hardware description*
- *Part 10: Advanced Video Coding (AVC)*
- *Part 11: Scene description and application engine*
- *Part 12: ISO media file format*
- *Part 13: IPMP extensions*
- *Part 14: MP4 file format*
- *Part 15: AVC file format*
- *Part 16: Animation Framework eXtension (AFX)*
- *Part 17: Streaming text format*

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1654
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Introduction

This Specifications defines text streams as a concatenation of text access units. The format of text streams and text access units is specified, as well as signaling and decoding of text streams.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1655
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Information technology — Coding of audio-visual objects — Part 17: Streaming text format

1. Scope

This document specifies ISO/IEC International Standard ISO/IEC 14496-17 Streaming Text.

2. Normative references

The following Standards contain provisions that, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All Standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent edition of the Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

All references in this subclause were correct at the time of approval of this International Standard. The provisions of the referenced specifications, as identified in this subclause, are valid within the context of this IS. The reference to a specification within this IS does not give it any further status within ISO/IEC; in particular, it does not give the referenced specification the status of an International Standard.

- 3GPP TS 26.245: 2003, *Timed text format (Release 6)*

3. Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

3.1 **Text access unit**: an individually accessible portion of data within a text stream. Each text access unit contains the coded representation of text data corresponding to one text sample. For presentation, the text access unit can be associated with a single time stamp.

3.2 **Text modifier**: data within a text sample, specifying a modification to the presentation of the text string within that text sample.

3.3 **Text sample**: text data of a specified format, consisting of a text string, optionally followed by one or more text modifiers

3.4 **Text string**: data within a text sample, representing a string of characters encoded using UTF-8 or UTF-16.

3.5 **TTU[j]**: a Timed Text Unit is a syntactical sub-structure of a text access unit for carriage of text access units across networks, whereby by its index j identifies which type of data from a text access unit is carried.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1656
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4. Text stream format

The text stream format defined in this Specification is derived from the timed text format defined by 3GPP for timed text downloaded in a 3GP file, as specified in 3GPP TS 26.245. The 3GPP timed text format defines text samples and sample descriptions. This Specification defines how to partition such text samples and sample descriptions into so-called TTUs, Timed Text Units, and how to concatenate them into text access units. TTUs provide a flexible structure that allows for easy and convenient adaptation to the various transport layers. For each transport layer the most suitable TTU structure can be chosen. If so desired, text access units can be re-partitioned into TTUs for most effective adaptation to other transport systems. Partitioning in TTUs is defined so, that 3GPP extensions are possible, but also extensions defined in future versions of this Specification. See figure 1.

A text stream is a concatenation of text access units. During transport typically a time stamp is assigned to each text access unit to indicate the intended time to present the text access unit on the display. The text is presented within a text box, the position of which can be controlled by parameters within the text stream.

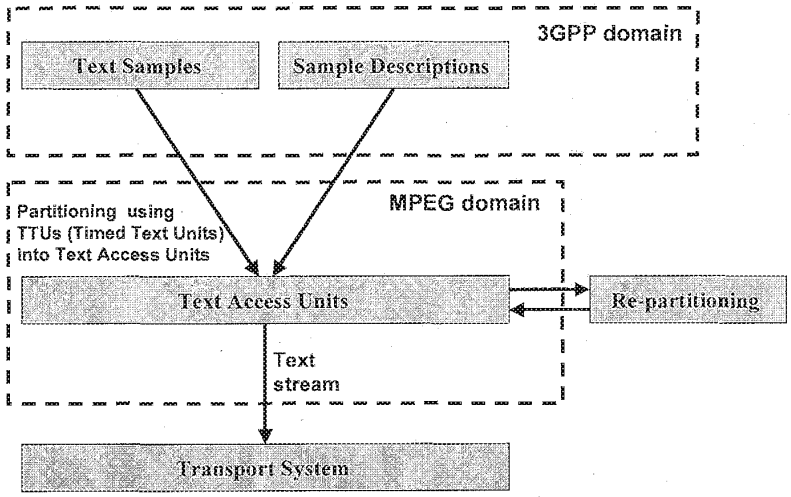


Figure 1. 3GPP defined text samples and sample descriptions and MPEG defined text access units.

4.1 Text samples

Each text sample represents text data and consists of a text string, optionally followed by one or more text modifiers, fully consistent with the TextSample format specified in 3GPP TS26.245. The text string represents the text to be displayed during a certain period, while the modifiers carry the changes that are to be applied to the text string during the time that the text is displayed, such as text colour changes synchronized with a song for a Karaoke application.

Each text string starts with 16 bit length field that specifies the length of the text string in bytes; note that this does not give the character count, as a character can vary in length from, in general, 1 to 6 bytes. The length of the text modifiers is not specified, as it is assumed that this information is derived from the total text sample length provided by means external to the text sample.

Global information about a text sample is provided by a sample description, for example about font(s) to be used, about the positioning of the text within the text box, the background colour of that text box, etc. Multiple sample descriptions are allowed; to each sample description (SD) an index is assigned and to each text sample the index of the applicable sample description is associated. A sample description will typically apply to multiple text samples; however, to each text sample exactly one sample description applies.

The relationship between sample descriptions, text samples, a text string and text modifiers is depicted in figure 2.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1657
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

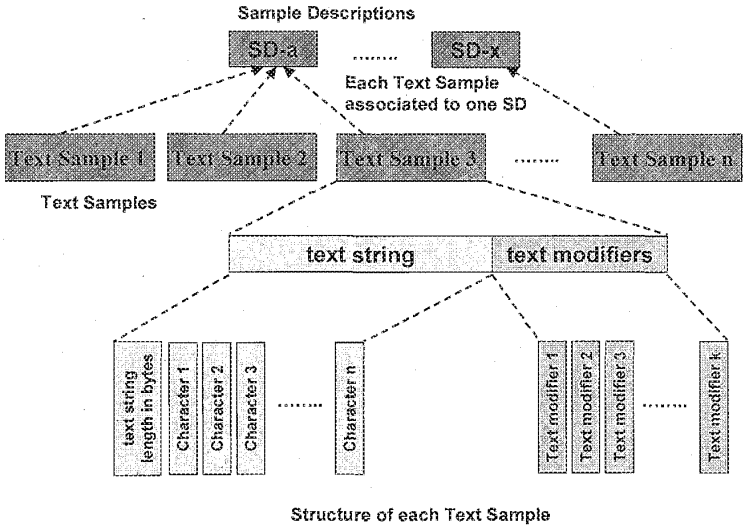


Figure 2. Sample Descriptions, Text Samples and their structure

4.2 Text access units

Each text access unit carries a text sample. Each text sample is to be presented during a certain period of time, specified by duration information contained in the text access unit that carries the text sample. At the instant in time that the presentation of a text sample ends, the next text sample starts being presented. To allow for periods of time without a text sample being presented, for instance between two subtitles, an "empty" text sample is defined, which is a text sample without data. If an "empty" text sample is immediately preceded by a non-"empty" text sample, then the "empty" text sample may be included in the same text access unit as the non-"empty" text sample.

Typically, a text sample is small, around 100 – 200 bytes, and often much smaller than the size of the packets that carry the text data across a transport network. It is therefore expected that transport systems will often aggregate multiple text samples into one such packet. On the other hand, text samples can also be large, for example when scrolling horizontal text at the bottom of the screen, in which case fragmentation of text samples may be required prior to transport. So as to achieve maximum commonality among text transport systems, a flexible and transport generic data structure is defined for containing a text sample within a text access unit. This data structure allows for a simple and generic network adaptation of text access units, and is implemented by means of so-called Timed Text Units, TTUs, of several types, each referred to as TTU[j]. The format of TTUs is defined in this document, as well as some general requirements for their use, while the actual usage of TTUs is left to the specifications for the applied transport means. However, as TTUs support text access unit fragmentation in a generic and error resilient manner, transport systems should use TTUs for fragmentation instead of transport specific aggregation and fragmentation tools.

Each text access unit is defined to consist of one or more TTUs. Five different type of TTUs are defined, one for carriage of a complete text sample, three for carriage of text sample fragments, and one for carriage of a complete sample description, while three types are reserved for future use. Because sample descriptions are small, there is no support for carriage of sample description fragments. Each TTU type is identified by its index j and referred to as TTU[j]. The following TTU types are defined:

- TTU[0]; this TTU type is reserved for future use;
- TTU[1]; this TTU type is capable of carrying one complete text access unit;
- TTU[2]; this TTU type is capable of carrying one fragment of a text string from a text sample;
- TTU[3]; this TTU type is capable of carrying the first text modifier fragment from a text sample;
- TTU[4]; this TTU type is capable of carrying a non-first text modifier fragment from a text sample;
- TTU[5]; this TTU type is capable of carrying one complete sample description;

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1658
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

- TTU[6] and TTU[7]; these TTU types are reserved for future use.

Figure 3 depicts the options to construct TTUs from text samples and sample descriptions. Each complete sample description is contained in one TTU[5]. Each text sample that is not fragmented, is contained in one TTU[1]. When a text sample is fragmented, then the text string is separated from the text modifiers for carriage in different TTUs. Each text string fragment is contained in one TTU[2], so that there are as many TTU[2]s as text string fragments. Note however that the text string itself may not be fragmented, in which case there is only a single text string fragment carried in one TTU[3]. The first text modifier fragment is contained in one TTU[3]. Hence, TTU[3] signals the start of the modifiers. Each subsequent text modifier fragment, if any, is contained in one TTU[4].

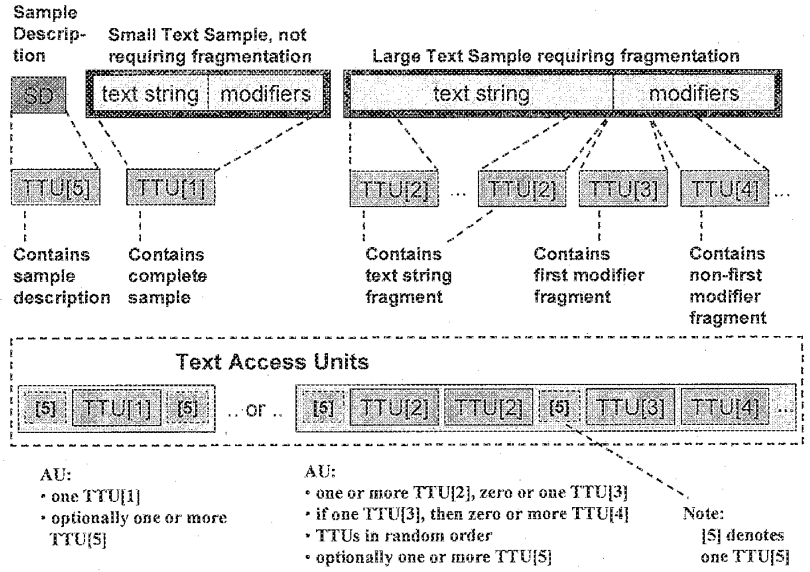


Figure 3. Carriage of Sample Descriptions and Text Samples in TTU[j]s, and construction of Text Access Units by means of TTUs.

In figure 3, also the basics of the construction of text access units by means of TTUs are depicted. If a text sample is not fragmented, then a text access unit contains the TTU[1] that carries the entire text sample. If the text sample is fragmented, then the text access unit contains one or more TTU[2]s, each with a fragment of the text string. If the text sample also carries modifiers, then the text access unit carries one TTU[3] and, depending on the size of the modifiers, zero or more TTU[4]s. In addition, each text access unit contains optionally one or more TTU[5]s, each carrying a sample description that may apply to this or future text samples in the text stream.

An empty text sample, that is a text sample without data (hence without a text string and modifiers), may be used to terminate the display of a text string prior to the start of the display of the next text sample in the stream. Such empty text sample is carried in a TTU[1], and may be included in the same access unit as the terminated text sample. Note that figure 3 does not illustrate how to include an empty text sample.

The order of TTUs within a text access unit is random within some constraints. When text samples are fragmented, it is allowed to repeat a fragment that is considered more important than others, and to transmit modifiers prior to text string fragments, which may be useful for large text strings. For ordering TTUs within a text access unit, the following constraints apply:

- When a sample description is provided in-band, then a sample description shall be included in the stream prior to any text sample that utilizes that sample description. This means that an in-band sample description, carried in a TTU[5], shall precede any TTU[1], TTU[2], TTU[3] and TTU[4] that references that sample description. Note that such TTU[5] may be carried in the same text access unit, or in one or more preceding text access unit(s).
- If an empty text sample is used to terminate the display of the text sample in a text access unit, then any other TTU[1], TTU[2], TTU[3] and TTU[4] in this text access unit shall precede the TTU[1] that contains the empty text sample.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1659
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.2.1.1 Text Sample fragmentation constraints

A number of constraints shall be applied when fragmenting text samples for carriage in TTU[2], TTU[3] and TTU[4]. The following constraints shall be applied:

- The first fragment of a text string shall commence with the 16 bit field that specifies the length in bytes of the entire non-fragmented text string;
- Each text string shall be fragmented on character boundary, so that each fragment contains complete character codes only, so as to ensure that each received text fragment can be decoded even in the case that a previous fragment gets lost. As a consequence, text string fragmentation requires knowledge of UTF-8/-16 formats to determine character boundaries.

It is not required to fragment text modifiers on modifier boundary; consequently, when a modifier fragment is lost, receivers will typically not be capable of decoding any remaining fragments.

4.2.2 TTU structure

Each TTU starts with an 8 bit TTU header; this header carries the type of TTU, signals whether characters in the text string of the text sample are encoded using UTF-8 or UTF-16, and signals whether the TTU header is followed by a field that indicates the length (size) of the TTU in bytes. This Specification defines some requirements for the presence and usage of the TTU length field; in addition, transport specific requirements not defined in this Specification may apply. The remaining fields after the TTU header are TTU type specific and described in the following sub-clauses.

4.2.2.1 TTU[1] signaling

A TTU[1] carries a complete text sample; the TTU header and the TTU length field, if present, are followed by fields that provide the following information:

- the index of the applicable sample description; and
- the duration of the text sample.

Carriage of an empty text sample in a TTU[1] is signaled by the presence of the TTU length field encoded with a specific value (see clause 4.3).

4.2.2.2 TTU[2] signaling

A TTU[2] carries a text string fragment from a text sample. The TTU header and the TTU length field, if present, are followed by fields that signal:

- the index of the applicable sample description;
- the duration of the text sample;
- the total length of the original non-fragmented text sample; this information allows the receiver to allocate buffer space for the decoding of the text sample; this information is provided for each text string fragment so as to also allow for buffer space allocation after losing one or more text string fragments;
- the total number of text sample fragments; this number shall include each text string fragment and each modifier fragment.
- the sequence number of the fragment contained in this TTU[2]; the sequence number of the first text sample fragment is equal to 0, and for each subsequent fragment the sequence number is incremented by one, so that in case of N fragments, the sequence number of the last fragment is equal to N-1. A text sample can be fragmented in at most 16 different fragments.

4.2.2.3 TTU[3] and TTU[4] signaling

In a TTU[3] and TTU[4], the first and a non-first modifiers fragment is carried, respectively; the TTU header and the TTU length field, if present, are followed by fields that signal:

- the total number of text sample fragments (see clause 4.2.2.2);
- the sequence number of the fragment contained in this TTU[2] (see clause 4.2.2.2);.

Note that the index of the applicable sample description, the duration of the text sample, and the total length of the fragmented text sample are not signaled, as modifiers fragments apply to text strings for which this information is already signaled.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1660
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.2.2.4 TTU[5] signaling

A TTU[5] carries a sample description. The TTU header and the TTU length field, if present, are followed by a field that signals:

- the index of the applicable sample description.

4.2.3 Constraints for using Sample Descriptions

Within text streams 256 index values are available to unambiguously identify a sample description. The values 0 and 255 are reserved. Sample descriptions can be provided either in-band or out-of-band. The index values in the inclusive range between 1 and 127 are available for in-band usage, while values between 128 and 254 (inclusive) can be used to provide sample descriptions out-of-band. See table 1.

Table 1 Index values for Sample Descriptions.

Index value	Assignment
0	Reserved
1 – 127	Available for in-band usage
128 – 254	Available for out-of- band usage
255	Reserved

When sample descriptions are provided in-band, there may be a need to regularly update sample descriptions. For this purpose a wrap-up mechanism is defined for in-band sample descriptions, that allows to continuously update such sample descriptions from a range of 64 valid index values. Hence, at any point in time at most 64 in-band sample descriptions can be in use simultaneously. Each sample description with an index value not in the valid range is discarded from further use. Valid sample descriptions can be transmitted in-band as often as desirable for random access, without any impact on the range of invalid index values. However, when receiving a sample description with an index value from the invalid range, then the valid range is changed, so that the received sample description becomes valid. When a sample description is received with index value i not in the valid range, then the receiver shall discard each sample description with an index value in the range between (i+1)mod128 and (i+64)mod128, inclusive. For example, assume that the valid range is between 41 and 104 inclusive, and hence, that the values 105 to 127 and 1 to 40 are invalid values. When now the sample description with index value 114 is received, the receiver shall invalidate the index values 115 to 127 and 1 to 50, inclusive, so as to change the valid range to values between 51 and 114.

4.3 Syntax and Semantics

4.3.1 Text_access_unit()

```
text_access_unit(){
      for (i=0; i<N;i++){
            TTU[j]()
            }
      }
```

4.3.2 TTU[j]()

```
TTU[j](){
      reserved                                    3     bslbf
      UTF_16_flag                                 1     bslbf
      TTU_length_flag                             1     bslbf
      TTU_type                                    3     uimsbf
      if (TTU_data_length_flag==1){
            TTU_data_length          16     uimsbf
            {
      TTU_data[j]()
      }
```


© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1661
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

`UTF_16_flag` – a one bit flag, indicating in TTU[1] and TTU[2] the encoding used for the characters in the text string; when set to '1', UTF-16 is applied, when set to '0', UTF-8 is applied. For TTU[3], TTU[4] and TTU[5] this bit has no meaning and shall be ignored.

`TTU_data_length_flag` – a one bit flag, that indicates, when set to '1', the presence in this `TTU[j]` of the `TTU_data_length` field. This field shall be set to '1' in each TTU[5], and in each TTU[1] that carries an empty text sample.

`TTU_type` – a three bit integer that specifies the type of TTU; the TTU-type field is encoded with a value that is equal to the index j in TTU[j], according to the following table.

Table 2 TTU types

TTU_type value	TTU type	TTU usage
0	TTU[0]	Reserved for future use
1	TTU[1]	Carriage of complete text sample
2	TTU[2]	Carriage of text string fragment
3	TTU[3]	Carriage of first modifiers fragment
4	TTU[4]	Carriage of non-first modifiers fragment
5	TTU[5]	Carriage of a complete sample description
6	TTU[6]	Reserved for future use
7	TTU[7]	Reserved for future use

`TTU_data_length` – an optional 16 bit integer that specifies the total number of bytes of the TTU_data_length field and the immediately following `TTU_data[j]()`. In a TTU[1], carriage of an empty text sample is signaled by a `TTU_data_length` value of 6. See clause 4.3.4.

4.3.3 TTU_data[0]()

```
TTU_data[0](){
      for (i=0; i<N;i++){
            reserved                                  8      bslbf
            }
      }
```

4.3.4 TTU_data[1]()

```
TTU_data[1](){
      sample_index                                    8      uimsbf
      sample_duration                                 24     uimsbf
      whole_text_sample()
      }
```

`sample_index` – an 8 bit unsigned integer that specifies the index of the sample description that applies to the text sample contained in this TTU. The sample_index value shall unambiguously refer to one particular sample description that shall be present in the text stream prior to its use. The sample_index values 0 and 255 are reserved.

`sample_duration` – a 24 bit unsigned integer that specifies the intended duration of the presentation of the text sample in this TTU in units of the durationClock, specified in textConfig; see Clause 7.

`whole_text_sample()` – TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see Clause 7).

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1662
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.3.5 TTU_data[2]()

```
TTU_data[2](){
      sample_index                                8      uimsbf
      sample_length                               16     uimsbf
      sample_duration                             24     uimsbf
      total_number_of_sample_fragments            4      uimsbf
      number_of_this_sample_fragment              4      uimsbf
      text_string_fragment()
      }
```

sample_index – See sample_index semantics in clause 4.3.4

sample_length – a 16 bit unsigned integer that specifies the total number of bytes in the TextSample of which a fragment is carried in this TTU.

sample_duration – See sample_duration semantics in clause 4.3.4

total_number_of_sample_fragments – a 4 bit unsigned integer that specifies for the TextSample of which a fragment is carried in this TTU the total number of fragments in which that TextSample is split.

number_of_this_sample_fragment – a 4 bit unsigned integer that specifies the sequence number of the TextSample fragment that is carried in this TTU. When the TextSample is split into N fragments, then the sequence number reflects the logical order of the fragments, that is the first fragment has sequence number 0, and the last fragment sequence number N-1, while for each subsequent fragment the sequence number is increased by 1.

text_string_fragment() – a fragment of the text string from the TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see Clause 7).

4.3.6 TTU_data[3]()

```
TTU_data[3](){
      total_number_of_sample_fragments            4      uimsbf
      number_of_this_sample_fragment              4      uimsbf
      first_modifier_box_fragment()
      }
```

total_number_of_sample_fragments – See total_number_of_sample_fragments semantics in clause 4.3.5.

number_of_this_sample_fragment – See number_of_this_sample_fragment semantics in clause 4.3.5.

first_modifier_box_fragment() – the first fragment of the text modifiers from the TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see Clause 7).

4.3.7 TTU_data[4]()

```
TTU_data[4](){
      total_number_of_sample_fragments            4      uimsbf
      number_of_this_sample_fragment              4      uimsbf
      non_first_modifier_box_fragment()
      }
```


© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1663
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

total_number_of_sample_fragments - See total_number_of_sample_fragments semantics in clause 4.3.5.

number_of_this_sample_fragment - See number_of_this_sample_fragment semantics in clause 4.3.5.

non_first_modifier_box_fragment() - a non-first fragment of the text modifiers from the TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in the DecoderConfiguration (see Clause 7).

4.3.8 TTU_data[5]()

```
TTU_data[5](){
      sample_index                              8      uimsbf
      sample_description()
      }
```

sample_index - the index of the sample description that applies to the text sample contained in this TTU. The sample_index value shall unambiguously refer to one particular sample description that shall be present in the text stream prior to its use. The sample_index values 0 and 255 are reserved. The sample_index values between 128 and 254 inclusive are assigned to out-of-band provided sample descriptions and shall not be used for in-band provided sample descriptions. Similarly, the sample_index values between 1 and 127 inclusive are assigned to in-band provided sample descriptions and shall not be used for out-of-band provided sample descriptions.

The following shall apply to in-band provided sample descriptions. At any instant in time, at most 64 different sample_index values shall be in use for in-band sample descriptions, identified by a range of valid sample_index values. When a receiver starts the decoding of a text stream, the receiver has no knowledge of the valid range, but upon receiving the first sample description the receiver shall define the range of valid sample_index values as follows. If the first received sample has the sample_index value j, then the receiver shall define the sample_index values in the range between (j+1)mod128 and (j+64)mod128, inclusive, invalid and to shall consider all other sample_index values between 1 and 127 (inclusive) valid. When subsequent sample descriptions are received with a sample_index value in the valid range, then the valid range shall not be modified. However, when a sample description is received with index value i not in the valid range, then the receiver shall invalidate sample_index values in the range between (i+1)mod128 and (i+64)mod128, inclusive.Each sample description with an sample_index value assigned as invalid shall be discarded. See also table 3. Text samples in the text stream shall not reference any invalidated sample description. A sample description is valid for each subsequent text sample in the text stream until invalidated by another sample description.

Table 3 Sample Descriptions that are invalidated by the Sample Description with sample_index value i

Received sample_index value i in currently invalid range	New invalid range (inclusive)	New valid range (inclusive)
$1 \le i \le 63$	(i+1) – (i+64)	1 – i (i+65) – (127)
$64 \le i \le 127$	(i+1) – 127 1 – (i-64)	(i-63) – i

sample_description() - a sample description for a TextSample, referred to as TextDescription, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in the DecoderConfiguration (see Clause 7).

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1664
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.3.9 TTU_data[6]()

```
TTU_data[6]{
      for (i=0; i<N;i++){
            reserved                        8      bslbf
            }
      }
```

4.3.10 TTU_data[7]()

```
TTU_data[7]{
      for (i=0; i<N;i++){
            reserved                        8      bslbf
            }
      }
```

5. Signaling of a text stream

A text stream shall be signaled by a streamType value 0x0B and by an objectTypeIndication value of 0x08.

6. Text rendering

Text from a decoded text stream is rendered within a Text Box. The Text Box can be positioned within a Text Track; see 3GPP TS 26.245. How to render the text in the Text Box and how to position the Text Box within the Text Track is specified within the text stream by means of sample descriptions and/or text modifiers.

The width and the height of the Text Track shall be defined in the decoderConfiguration, textConfig, in units of square pixels. In textConfig it is also specified whether or not a text stream is intended to be positioned relative to a video stream. Note that a Text Track can also be positioned with respect to a video picture by means of information that is provided by external means, such as information contained in an MP4 file, or by an application. External information, if provided, takes priority over information provided within textConfig.

If the text stream is associated with a video picture, then textConfig specifies the video-width and video-height of that video picture. The video-width and video-height, as well as the position of the Text Track relative to the video picture are specified in units of the same square pixels as the width and the height of the Text Track, so as to enable scaling and positioning of the text with respect to video in a manner independent of the video resolution. This allows that the same text stream can be used with video streams coded at different resolutions and bitrates. The relative position of the Text Track with respect to the video picture is specified by the horizontal-offset and vertical-offset. These two parameters specify the horizontal and vertical distance in pixels between the top left pixel of the video picture and the top left pixel of the Text Box, respectively; see figure 4. It is allowed that the Text Box is partly or completely outside of the video picture. A positive value of horizontal-offset and vertical-offset indicate that top left pixel of the Text Box is right and below, respectively, of the top left pixel of the video picture. For example, in figure 4, both horizontal-offset and vertical-offset have a positive value. Accordingly, a negative value of horizontal-offset and vertical-offset indicates that the top left pixel of the Text Box is left and above, respectively, of the top left pixel of the video picture. Note that the square pixels units used in textConfig to specify the sizes of the Text Track and the video picture and their relative positions will typically have a different dimension than the actually coded video pixels. Hence if the video-width and video-height specify a picture with a size of 640*480, then the picture may actually be coded at 720*480 or 320*240 or any other resolution.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1665
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

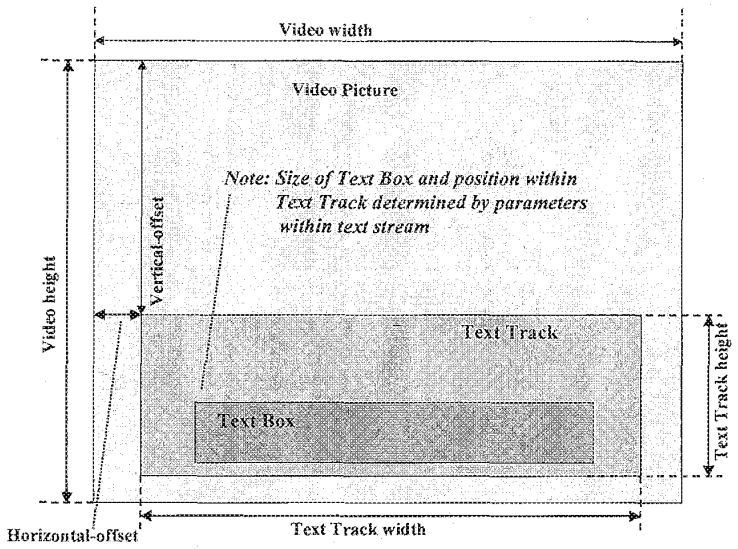


Figure 4. Size of the Text Track and the positioning of the Text Track with respect to a video picture.

7. DecoderConfiguration for Timed Text decoders

To decode a text stream, a receiver needs the decoderConfiguration for the text stream, as defined by TextConfig. The TextConfig provides information such as:

- the 3GPP format that the text stream is compatible with
- whether an MPEG defined extension is applied
- the profile and level of the text stream
- clock units used to encode text sample durations
- whether in-band sample descriptions are applied
- whether out-of-band sample descriptions are applied
- whether sample descriptions are carried in the textConfig
- optional information on the positioning of the text track relative to a video picture

7.1 Syntax

```
class TextConfig extends DecoderSpecificInfo : bit(8) tag=DecSpecificInfoTag {
   bit(8)  3GPPBaseFormat;
   bit(8)  MPEGExtendedFormat;
   bit(8)  profileLevel;
   bit(24) durationClock;
   bit(1)  contains-list-of-compatible-3GPPFormats-flag;
   bit(2)  sampleDescriptionFlags;
   bit(1)  SampleDescription-carriage-flag;
   bit(1)  contains-associated-video-information;
   bit(3)  reserved;
   bit(8)  layer;
   bit(16) text-track-width;
   bit(16) text-track-height;
   if (contains-list-of-compatible-3GPPFormats-flag) {
      bit(8) number-of-formats;
```

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1666
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
      bit(8) Compatible-3GPPFormat(number-of-formats);
      }
   if (SampleDescription-carriage-flag) {
      bit(8) number-of-SampleDescriptions;
      SampleDescription()(number-of-SampleDescriptions);  //as many SampleDescriptions,
      }                          each carried in a TTU[5], as number-of-SampleDescriptions
   if (contains-associated-video-information) {
      bit(16) video-width;
      bit(16) video-height;
      bit(16) horizontal-offset;
      bit(16) vertical-offset;
      }
   bit(8)  remaining-config-length;
   bit(8)  reserved(remaining-config-length);
}
```

7.2 Semantics

`3GPPBaseFormat` – one byte signaling the latest 3GPP format that this text stream is compatible with, coded as specified in table 4. If the text stream is also compatible with older versions of 3GPP specifications, then these formats can be listed in the textConfig by setting the `contains-list-of-compatible-3GPPFormats-flag`.

Table 4 – 3GPPBaseFormat

0x00 – 0x0F	Reserved
0x10	Timed Text as specified in 3GPP TS 26.245
0x11 – 0xFF	Reserved

`MPEGExtendedFormat` – one byte signaling the MPEG defined compatible extension of the 3GPPBaseFormat applied for this stream, coded as specified in table 5.

Table 5 – MPEGExtendedFormat

0x00 – 0x0F	Reserved
0x10	No extension applied
0x11 – 0xFF	Reserved

`profileLevel` – an 8 bit unsigned integer that specifies the profile and level of the text stream encoded as specified in table 6. See Clause 9 for profile and level constraints.

Table 6 profileLevel

0x00 – 0x0F	Reserved
0x10	Base profile, base level
0x11 – 0xFF	Reserved

`durationClock` – a 24 bit unsigned integer that specifies the frequency in Herz of the clock used to encode the `sample-duration` field in TTU[1] and TTU[2]. A value equal to 1000 signals a 1kHz clock, and that `sample-duration` field is encoded in units of 1 msec. Transport systems should assign time stamps to text access units using the same frequency as signaled by `durationClock` or an integer multiple thereof.

`contains-list-of-compatible-3GPPFormats-flag` – one bit that signals the presence of a list of one or more 3GPP formats older than the signaled format by `3GPPBaseFormat`, that the text stream is also compatible with.

`sampleDescriptionFlags` – two bits that signals whether SampleDescriptions are provided in-band, out-of-band or both, as specified in table 7.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1667
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Table 7 – sampleDescriptionFlags

00	Forbidden
01	No in-band, out-of band only
10	In-band only, no out-of band
11	Both in-band and out-of band

`SampleDescription-carriage-flag` – one bit that signals the presence of a list of SampleDescriptions in this textConfig. This bit shall be set to '0', if `sampleDescriptionFlags` is coded with the value '10'.

`contains-associated-video-information` – one bit that signals the presence of the `video-width, video-height, horizontal-offset, and vertical-offset` fields in this TextConfig. If some or all of this information is also provided by external means, then the externally provided information takes priority.

`layer` – unsigned integer specifying the proximity of the text track to the viewer. A higher value means closer to the user. A value of '0' indicates an undefined proximity. If the proximity is also provided by external means, then the externally provided information takes priority.

`text-track-width` – unsigned integer specifying the width of the text track in pixels.

`text-track-height` – unsigned integer specifying the height of the text track in pixels.

`number-of-formats` – unsigned integer specifying the number of immediately following `Compatible-3GPPFormat` fields. The value '0' is forbidden.

`Compatible-3GPPFormat` – one byte that signals a 3GPP formats older than the signaled format by `3GPPBaseFormat`, that the text stream is also compatible with, coded in the same way as the `3GPPBaseFormat` field, specified in table 4.

`number-of-SampleDescriptions` – unsigned integer that specifies the number of immediately following `SampleDescriptions()`.

`SampleDescription()` – one sample description carried in one TTU[5].

`video-width` – unsigned integer specifying the width of the video picture in pixels[1]

`video-height` – unsigned integer specifying the height of the video picture in pixels; see footnote 1.

`horizontal-offset` – signed integer specifying the horizontal distance in pixels (see footnote 1) between the left border of the video picture and the left border of the Text Box; a positive distance value indicates that the left border of the Text Box is right of the left border of the video picture

`vertical-offset` – signed integer specifying the vertical distance in pixels (see footnote 1) between the top border of the video picture and the top border of the Text Box; a positive distance value indicates that the top border of the Text Box is below the top border of the video picture

`remaining-config-data-length` – unsigned integer, specifying the number of immediately following `reserved` bytes in this textConfig.

[1] These are the square pixels used to specify the Text Box dimensions, not coded video pixels.

© ISO/IEC 2004 --- All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1668
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

8. Text stream decoding

A hypothetical decoder has been defined for decoding of text streams. The Hypothetical Text Decoder (HTD) consists of a TTU Decoder, a textSampleBuffer, an in-band-sampleDescriptionBuffer, an out-of-band-sampleDescriptionBuffer and a textDecoder; see figure 5. Input to the HTD is a text stream with a bitrate of R kb/s. The value of R, as well as the textSampleBuffer size, the in-band-sampleDescriptionBuffer size and the out-of-band-sampleDescriptionBuffer size is specified per profile and level; see Clause 9.

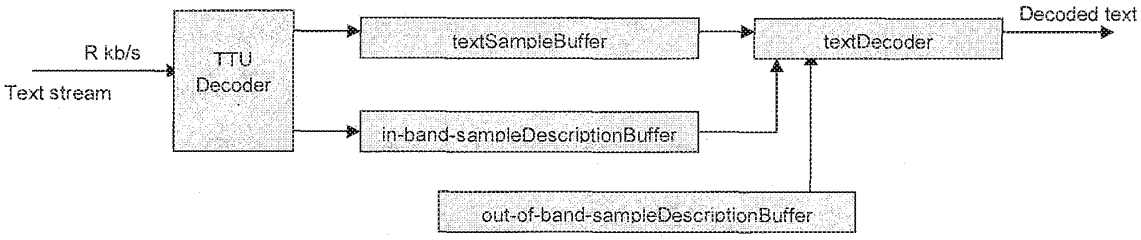


Figure 5 Hypothetical Text Decoder

In the HTD, each out-of-band sample description utilized by the text stream is provided prior to its usage by the text stream to the out-of-band-sampleDescriptionBuffer by means not defined by this Specification. The total size in bytes of all out-of-band sample descriptions used in the text stream shall not exceed the specified size of the out-of-band-sampleDescriptionBuffer.

At the input of the HTD, the TTU decoder processes each received TTU byte instantaneously. Contained data from text samples are transferred to the textSampleBuffer, while in-band provided sample descriptions are transferred to the in-band-sampleDescriptionBuffer. At any point in time, the total size in bytes of all valid in-band sample descriptions shall not exceed the specified size of the in-band-sampleDescriptionBuffer. Text streams shall be constructed so that each in-band sample description is completely present in the in-band-sampleDescriptionBuffer at the instant in time that a text sample that references that sample description is decoded.

If the textSampleBuffer is not full, then the bytes of the text stream enter the TTU decoder at a bitrate R; if the textSampleBuffer is full, no bytes enter the TTU decoder. The byte in the text stream subsequent to the text sample byte that causes the textSampleBuffer to be full does not enter the TTU decoder as long as the textSampleBuffer remains full. At the instant in time that the presentation of a text sample is to start, the text sample is instantaneously removed from the textSampleBuffer, decoded and presented. Underflow of the textSampleBuffer occurs if a text sample has not (yet) fully entered the textSampleBuffer at the instant in time that the text sample is to be presented. Text streams shall be constructed so, that in the hypothetical text decoder no underflow occurs:

- The size of each text sample shall be smaller than or equal to the specified size of the textSampleBuffer;
- Taking the fullness of the textSampleBuffer into account, there shall be sufficient time to deliver the complete text sample to the textSampleBuffer at the specified bitrate prior to its decoding.

Note: In the extreme case that the size of the previous text sample is equal to the size of textSampleBuffer, then the next text sample is to be delivered during the duration of the previous one. To avoid underflow in this extreme case, the encoder has to ensure that the duration between the previous non-empty sample and the next non-empty sample N (note that there may be one or more "empty" samples in-between) is larger than the size of the non-empty sample N divided by the applicable bitrate R.

9. Profiles and Levels for text streams

For text streams this specification defines only a single profile and a single level: the base profile and the base level. Other profile and levels may be defined in future. The profile and level are signaled in textConfig, see Clause 7 and table 6.

In each text stream that conforms to the base profile, each tool specified by 3GPP in 3GPP TS 26.245 may be used. In each text stream that complies with the base level, the HTD parameter values defined in table 8 shall apply. Each decoder that complies with the base profile and the base level shall be capable of decoding each text stream that conforms to the base profile and the base level.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1669
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Table 8 HTD parameter values for the Base Level

HTD Parameter	Value
Bitrate R	10 kb/s
Size of textSampleBuffer	8192 Byte
Size of in-band-sampleDescriptionBuffer	4096 Byte
Size of out-of-band-sampleDescriptionBuffer	4096 Byte

Editorial note: the HTD model and the above numbers need further consideration

© ISO/IEC 2004 --- All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1670
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Annex A

Font Referencing

A.1 Font references in text streams

Timed Text streams may reference fonts in the receiver. These fonts may be residently available in the receiver, or downloaded or streamed to the receiver. To ensure that the desired fonts in the receiver are referenced correctly, the font name identification in textData() has to uniquely reference the name of the desired font in the receiver.

In an MPEG-4 systems context, when a font stream is used to supply fonts for a text stream, that text stream shall indicate that dependency by using the dependsonES_ID field in the ES_Descriptor of the text stream, containing the ES_ID of the font stream. Other systems contexts should indicate that dependency in an appropriate way.

In text streams which have an associated font stream, when a specific font instance in the font stream is to be used for a text access unit, the font name shall use the appropriate naming structure, which for MPEG-4 font streams is defined in 14496-18 section 5.5.

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1671
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

INTERNATIONAL STANDARD

ISO/IEC 14496-18

First edition
2004-07-01

Information technology — Coding of audio-visual objects —

Part 18: Font compression and streaming

Technologies de l'information — Codage des objets audiovisuels —
Partie 18: Compression et transmission de polices de caractères

Reference number
ISO/IEC 14496-18:2004(E)

© ISO/IEC 2004

Licensed to Online Server ANSI c/o X_Default. Downloaded 2020-01-[illegible] Copyright ANSI. Single user license only. Copying, networking and distribution prohibited.

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2004

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

© ISO/IEC 2004 – All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1673
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Contents

Page

Foreword iv
Introduction vi
1 Scope 1
2 Normative references 1
3 Font Data Format 1
4 Font Compression Technology 2
4.1 Overview 2
4.2 Compressed Font Format Specification 2
5 Font Data Stream 12
5.1 Structure of the Font Data Stream 12
5.2 Access Unit Definition 12
5.3 Time Base for Font Data Streams 14
5.4 Font Data Decoder Configuration 14
5.5 Accessing the Font Data 15
6 Text Profiles and Levels 15
6.1 Simple Text Profile and Levels 15
6.2 Advanced Simple Text Profile and Levels 16
6.3 Main Text Profile and Levels 17
Annex A (informative) Patent Statements 18

© ISO/IEC 2004 – All rights reserved

Downloaded by Shaun Doyle (ISO) on Oct 7, June 2005, Downloaded from IHS under no license agreement. For future or additional use, Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1674
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14496-18 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

— *Part 1: Systems*

— *Part 2: Visual*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Delivery Multimedia Integration Framework (DMIF)*

— *Part 7: Optimized reference software for coding of audio-visual objects*

— *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*

— *Part 9: Reference hardware description*

— *Part 10: Advanced Video Coding*

— *Part 11: Scene description and application engine*

— *Part 12: ISO base media file format*

— *Part 13: Intellectual Property Management and Protection (IPMP) extensions*

— *Part 14: MP4 file format*

— *Part 15: Advanced Video Coding (AVC) file format*

— *Part 16: Animation Framework eXtension (AFX)*

© ISO/IEC 2004 – All rights reserved

Licensed to Sean Terry, ANSI Store, 2 Users, Downloaded 8/25/2011 9:22 PM, Single user licensing only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1675
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

— *Part 17: Streaming text format*

— *Part 18: Font compression and streaming*

— *Part 19: Synthesized texture stream*

© ISO/IEC 2004 – All rights reserved

Licensed to Seoul Young ANSI Order #: [illegible], Downloaded: [illegible] Single user licence only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1676
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Introduction

ISO/IEC 14496 specifies a system for the communication of interactive audio-visual scenes. The specification includes the following elements:

1. the coded representation of natural or synthetic, two-dimensional (2D) or three-dimensional (3D) objects that can be manifested audibly and/or visually (audio-visual objects) (specified in part 1,2 and 3 of ISO/IEC 14496);

2. the coded representation of the spatio-temporal positioning of audio-visual objects as well as their behaviour in response to interaction (scene description, specified in part 11 of ISO/IEC 14496);

3. the coded representation of information related to the management of data streams (synchronization, identification, description and association of stream content, specified in part 11 of ISO/IEC 14496);

4. a generic interface to the data stream delivery layer functionality (specified in part 6 of ISO/IEC 14496);

5. an application engine for programmatic control of the player: format, delivery of downloadable Java byte code as well as its execution lifecycle and behaviour through APIs (specified in part 11 of ISO/IEC 14496); and

6. a file format to contain the media information of an ISO/IEC 14496 presentation in a flexible, extensible format to facilitate interchange, management, editing, and presentation of the media.

The information representation, specified in ISO/IEC 14496-1 and in ISO/IEC 14496-11, describes the means to create an interactive audio-visual scene in terms of coded audio-visual information and associated scene description information. The encoded content is presented to a terminal as the collection of elementary streams. Elementary streams contain the coded representation of either audio or visual data or scene description information or user interaction data. Elementary streams may as well themselves convey information to identify streams, to describe logical dependencies between streams, or to describe information related to the content of the streams. Each elementary stream contains only one type of data.

Elementary streams are decoded using their respective stream-specific decoders. The audio-visual objects are composed according to the scene description information and presented by the terminal's presentation device(s). All these processes are synchronized according to the systems decoder model (SDM) using the synchronization information provided at the synchronization layer.

The scene description stream identifies different types of objects, such as audio, visual, 2D and 3D graphics, etc. that define a scene composition of the content. Among these objects, the essential part of almost any multimedia presentation is text objects that are created utilizing specific custom fonts. Font selection determines the appearance of a text in multimedia content and it's the most critical factor that assures text legibility and readability. It also plays critical role in the overall scene composition since the metric properties of a font are used for textual parts of multimedia content layout. Many thousands of fonts are available today for use in content creation and in order to assure correct appearance and layout of a content the font data have to be included (embedded) with the text objects as part of the multimedia presentation.

Font data compression and streaming technology presented in this document provide efficient mechanism to embed font data in MPEG-4 encoded presentations.

© ISO/IEC 2004 -- All rights reserved

Licensed to Brian Berg, ANSI under A_336229. Downloaded 8/23/2011 6:22 PM. Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1677
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Information technology — Coding of audio-visual objects —

Part 18: Font compression and streaming

1 Scope

This part of ISO/IEC 14496 specifies functionalities for the communication of font data as part of the MPEG-4 encoded audio-visual presentation. More specifically, it defines:

1. Font format representation that is utilized for font data encoding (OpenType);
2. Font compression technology for TrueType and OpenType fonts with TrueType outlines; and
3. The coded representation of information in font data streams.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14496-1, *Information technology — Coding of audio-visual objects — Part 1: Systems*

ISO/IEC 14496-11, *Information technology — Coding of audio-visual objects — Part 11: Scene description and application engine*

The OpenType Specification - available on the Microsoft Typography website at <http://www.microsoft.com/typography/otspec/default.htm> or the Adobe Solutions Network website at <http://partners.adobe.com/asn/tech/type/opentype/index.jsp>

3 Font Data Format

In order to guarantee the original appearance of the content, to preserve corporate branding and identity in streaming multimedia presentations and to provide support for all languages, MPEG-4 supports text rendering utilizing rich formatting capabilities and custom fonts.

MPEG-4 adopts OpenType®[1], version 1.4, as its font data format for the purposes of uniform font data transmission and predictable text rendering. OpenType has emerged as the font solution for high-quality text processing, multimedia applications and cross platform Internet document portability. OpenType is a full-featured font format that enables the highest quality of text rendering on low-resolution displays, advanced typographic features and international character support. It is fully compatible with the existing and widely adopted TrueType™[2] fonts.

MPEG-4 requires fonts to contain a Unicode character map ('cmap') table.

---

1) OpenType is a registered trademark of Microsoft Corporation.

2) TrueType is a trademark of Apple Computer Incorporated.

© ISO/IEC 2004 – All rights reserved

Licensed to Bibliotheque ANSI Order # 2363281 Downloaded: 8/27/2011 8:38 PM Single-user license only, Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1678
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4 Font Compression Technology

4.1 Overview

Transmission of the font data through bandwidth-limited channels requires efficient compression techniques be applied that allow reducing font data size without sacrificing quality of text rendering and advanced typographic features supported by the font. MPEG-4 adopts MicroType® Express[3] 2.0 font compression technology, which allows lossless compression of OpenType fonts with TrueType outlines and provides the capability to render the text without decompressing the whole font, which significantly reduces memory requirements. The specification of this compression mechanism is presented below.

4.2 Compressed Font Format Specification

4.2.1 Introduction

This part of the document describes the compressed font format that is designed for compact binary representation of lossless-compressed TrueType and OpenType with TrueType outlines fonts (TTF) and font collections (TTC). This font compression technology for OpenType TrueType fonts is designed to minimize memory usage for font data storage in resource-constrained environments, where both ROM and RAM memory savings are critical, and build relatively small font files without sacrificing the quality, features and capabilities of TrueType and OpenType fonts.

The compressed font format provides random access capabilities into the font data and allows font rasterizer to render glyphs directly from the compressed font tables without decompressing the whole font while maintaining the highest character quality. The selection and number of compressed tables in a font file may differ from one font to another. Any number of tables can be compressed in the same font (from only one to all), which allows application developers to optimize "memory savings – performance of font rendering engine" trade-off.

Compression algorithms, applied to font tables, take advantage of advanced knowledge of OpenType font tables structure and meaning of data (text, TrueType instructions, glyph data, etc.). Separation of these objects into groups with distinctly different statistical distributions allows employing a number of compression techniques, which results in producing most compact lossless-compressed data. These techniques are especially effective when compressing of the glyph data ('glyf' table), which is typically the biggest table in the TTF file. The details of the glyph data compression are described in subclause 4.2.4 of this specification.

4.2.2 Data Types

The compressed font format follows the same SFNT structure (used in both TrueType and OpenType) that provides a "wrapper" for a collection of tables in a general and extensible manner. Rasterizers use a combination of tables contained in the font to render glyph outlines. In order to simplify the process of traversing the tables in a font, the beginning of each table is aligned on the 4-byte boundary and each table data is padded with zeros.

Most tables have version numbers, and the version number for the entire font is contained in the Table Directory. The compressed font format specification uses the same data types that are used in the OpenType specification. For all details on data types and version number format please see the OpenType font file specification.

3) MicroType is a registered trademark of Agfa Monotype Corporation.

This information is given for the convenience of users of this International Standard and does not constitute an endorsement by ISO/IEC of the product named.

© ISO/IEC 2004 – All rights reserved

Licensed to Hilton Kang, ANSI Order # 238128, Downloaded on 2/25/2011 8:05 PM, Single-user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1679
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.2.3 Compressed Font Format

4.2.3.1 The Compressed Font Header

The compressed font file format is the same as an OpenType font file, which always begins with the Offset Table followed by the Table Directory entries. If the font file is TTC file, the offset to each Table Directory is defined in TTC Header. If the font file contains only one font, the Offset Table will begin at byte 0 of the file. For details about an OpenType font Table directory and TTC header structure please see "Organization of an OpenType Font" in the OpenType font file specification.

4.2.3.2 OpenType Tables

The OpenType font file specification defines the set of required tables to be used in an OpenType font with TrueType outlines. When this font compression mechanism is used with OpenType fonts with TrueType outlines, the ACT3 additional table described in the following section is required. The tag describing this table is '**act3**'.

Other tables may be defined but are not required. OpenType fonts may also contain bitmaps of glyphs, in addition to outlines, as well as other optional tables that support vertical layout and advanced typographic features. For a complete list of OpenType font tables, their definitions and data structures see The OpenType font file specification.

4.2.3.3 ACT3 Table Structure

The 'ACT3' table defines the mapping of all tables, present in the font, to the collection of compressed and uncompressed blocks of data in a font file. Since the herein compressed font file format allows a mixture of both compressed and uncompressed data blocks to be present in the file, the mapping of the data blocks relative to the original OpenType TrueType table structure will be as follows:

- all consecutive uncompressed tables of the original font file will be referenced as a contiguous block of uncompressed data by a single entry in the ACT3 table;
- each compressed table of the original font file will be referenced as a block of compressed data by a single entry in the ACT3 table.

NOTE: The 'act3' table is required to be present in the compressed font OpenType Table Directory. If the font file is TrueType collection, then the table will only be part of the first font Table Directory and will contain mapping information for all fonts in TrueType Collection font file.

Table 1 — 'act3' table syntax

Data Type	Syntax	Description and Comments
	`/* ACT3 Table Header*/`	
USHORT	`TableLength`	Length of the 'act3' table padded to 4-byte boundary
ULONG	`CompFontSize`	Size of the compressed font file
ULONG	`UncompFontSize`	Original size of the uncompressed font file after CTF compression was applied to glyph data.
ULONG	`GlyfStart`	Offset to the start of compressed 'glyf' table in uncompressed font file
ULONG	`GlyfEnd`	Offset to the position immediately following the end of compressed 'glyf' table in uncompressed font file
USHORT	`NumEntries`	Number of different compressed/uncompressed data blocks in the font file
	`/* ACT3 Table Entries*/`	
	`for (i=0; i<numEntries; i++)`	
	`{`	

© ISO/IEC 2004 – All rights reserved

Licensed to Brian Berg (ATSI-order #_103826). Downloaded 8/23/2011 1:32 PM. Single user licence only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1680
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ULONG	BlockStart	Offset to the start of uncompressed data block in original font file
ULONG	BlockLength	Length of uncompressed data block in original font file
ULONG	CompBlockStart	Offset to the start of data block in the compressed font file
USHORT	IsCompressed	Flag indicating whether block is compressed (non-zero if block is compressed, **INTERVAL** = isCompressed)
	}	

4.2.3.4 Structure of Compressed Block

Each compressed table in the original OpenType TrueType font or font collection file is represented by the single block of data in the ACT3 table. The data is losslessly compressed using canonical Huffman codes. When the table is compressed, an additional set of data is generated in order to enable the decoder perform random access into compressed data stream and selectively decompress only those chunks of data that are requested by the font rendering engine.

Random access in the compressed dataset is achieved by providing additional information on the corresponding positions of the start of Huffman codeword in the compressed data stream (bitmarks) and the start of a token in the uncompressed data. This relationship is established through predefined intervals between reference points in the original data.

The size of the interval between reference points affects both the compression efficiency of the encoder (a smaller interval would require more data be added to the output compressed data stream) and font rendering performance (a bigger interval would require font engine to decompress larger dataset in order to read the requested data). Therefore, a careful consideration should be given to selection of the interval between random access entry points. The analysis of the TrueType features, their sizes and empirical rendering performance results for Latin fonts showed that the optimal placement of reference points is achieved at the **INTERVAL** = 128 bytes.

The block of compressed data has **RAC3** tag and consists of the following data segments:

- The dictionary of tokens which were defined by the encoder;
- Definition of the canonical Huffman codes;
- Locations of the random access pointers to the compressed bitstream and original data;
- Compressed data bitstream.

Table 2 — RAC3 block format

Data Type	Syntax	Description and Comments
	/* RAC3 Block Header */	
TAG	RACtag	Compressed data ID string: 'R' 'A' 'C' 3 (0x52414303)
ULONG	compDataSize	Size of the compressed data
ULONG	uncompDataSize	Original size of the uncompressed block of data
	/* RAC3 Dictionary */	
ULONG	numTokens	Number of tokens in the dictionary.
BYTE	tokenLength[numTokens]	Array of length for each token in the dictionary (in bytes).
BYTE	offsetLength	Length of the tokenOffset from the beginning of the tokenString array to the start of the token (in bytes)
	if (offsetLength ==0)	Tokens are non-overlapping, each
	{	tokenOffset into the tokenString can be inferred from the array of tokenLength[].

© ISO/IEC 2004 – All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1681
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ULONG	tokenStringSize	Length of the tokenString in bytes.
BYTE	tokenString[tokenStringSize]	Dictionary of the tokens.
	} else	We have overlapping tokens, offsets into the tokenString are explicitly specified in the tokenOffset[].
	{	
BYTE	tokenOffset [(numTokens*offsetLength)]	Array of offsets for each token in the tokenString. Each element of the array is offsetLength bytes long.
ULONG	tokenStringSize	Length of the tokenString in bytes.
BYTE	tokenString[tokenStringSize]	Dictionary of the tokens.
	}	
	/* Canonical Huffman Codes */	
BYTE	numGroups	Number of Huffman code length groups (can be from 1 to 32)
BYTE	codeLength[numGroups]	Array of code length in each Huffman group.
ULONG	firstGroupCode[numGroups]	Array of the first code in each Huffman group.
ULONG	lastGroupCode[numGroups]	Array of the last code in each Huffman group.
	/* Random Access Pointers */	
ULONG	numIntervals	Number of **INTERVAL** byte intervals in the uncompressed source data
ULONG	bitMarks[numIntervals]	Bit offset to the start of the nearest Huffman codeword from the beginning of compressed data block corresponding to n^{th} interval.
BYTE	deltaBytes[numIntervals]	Offset to the start of the nearest Huffman token from the beginning of uncompressed source data = **INTERVAL** * n – deltaBytes[n]
ULONG	BitStringOffset	Bit offset to the start of the compressed bitString[] from the beginning of the compressed data block.
BYTE	BitString[compDataSize-bitStringOffset/8]	Compressed data bitstream.

4.2.4 Glyph Data Compression

4.2.4.1 Overview

The 'glyf' table contains all the information about simple and composite glyphs in OpenType TrueType font. Simple glyphs are recorded as a collection of data segments representing number of contours, bounding box, TTF instructions for glyph scaling and arrays of data point coordinates. If composite characters are present, they contain the array of simple glyph indexes along with the scaling information and TTF instructions for the composite glyph. These datasets have distinctly different statistical distributions of values and, sometimes, significant levels of redundancy. The use of internal dependencies of glyph data segments and knowledge of TrueType rasterizer behaviour allows the application of additional compression algorithms to reduce glyph data set without any affect on the functionality of glyph rendering and the output quality of TrueType rasterizer.

4.2.4.2 Additional Data Types

TrueType (SFNT) file format often uses USHORT and SHORT data types to provide enough storage space for numerical data. However, in the majority of cases the stored number is rather small and can be recorded within one byte. In order to reduce storage size for TTF glyph data while keeping byte alignment of data records we introduce **255USHORT** and **255SHORT** data type that records the numerical **Value** as follow:

© ISO/IEC 2004 – All rights reserved

Licensed to Brian Fung, ANSI order # 0_203920. Downloaded: 6/23/2011 6:22 PM. Single user licence only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1682
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Table 3 — 255USHORT Data Type

Data Type	Syntax	Description and Comments
BYTE	Code if (Code == 255)	Value = Code; /* [0..255] */
BYTE	Value1 else if (Code == 254)	Value = 253 + Value1; /* [253..508] */
BYTE	Value1 else if (Code == 253)	Value = 506 + Value1; /* [506..761] */
USHORT	Value	Value; /* [0..65535] */

Table 4 — 255SHORT Data Type

Data Type	Syntax	Description and Comments
BYTE	Code if (Code < 250) {}	Code; /* [0..255] */ Value = Code; /* [0..249] */
	else if (Code == 250)	Flip sign code.
BYTE	Value1 else if (Code == 253)	Value = (SHORT)(–(Value1 + 1)); Word data code. Codes 251 and 252 are reserved for "Hop Codes", see below and subclause 4.2.4.3 for details.
SHORT	Value else if (Code >= 254)	Value; /* [-32768..32767] */
BYTE	Value1 else if (Code==251 || Code==252) {}	Value = Value1+256*(255-Code)+250; Value = Value1+256*(255-Code)+250; Reserved HopCode values;

4.2.4.3 Glyph Records

4.2.4.3.1 Overview

Glyph records are located in the 'glyf' table of OpenType font. Glyphs are recorded as a set of bounding box values, TrueType instructions, flags and outline data points. The glyph record size can be reduced with no effect on the quality of the glyph rendering. This compressed font format converts simple TTF glyph table records to lossless Compact Table Format (CTF).

4.2.4.3.2 Bounding Box

In most cases, the bounding box information can be computed from the data point coordinates and, therefore, bounding box information may be omitted in glyph record.

4.2.4.3.3 Contour End Points

TrueType glyphs store actual numbers of the end point for each contour. CTF stores the number of points in each contour instead, which is typically less then 255. Translation to the endpoint number is simply the cumulative sum of point count in all preceding contours.

4.2.4.3.4 Instructions

TrueType glyphs contain instructions for glyph scaling. Glyph programs typically start with a burst of various PUSH instructions to initialize the stack for a TrueType instructions belonging to that glyph. TrueType provides four different types of PUSH instructions that occupy 18 different opcodes. In order to optimize data storage

© ISO/IEC 2004 – All rights reserved

Licensed to Brian Bung, ATSI order #_276020. Downloaded 8/23/2011 3:22 PM. Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1683
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

size, the CTF format stores only data to be pushed onto the stack. The translation process back to TTF generates a new sequence of optimized TrueType PUSH instructions that results in the identical functionality of glyph program.

The storage format of the initial push data utilizes additional **HopCodes** that make use of the specific data patterns observed in TrueType data. It capitalizes on a specific distribution of the data that a traditional compression technologies do not take advantage of. A typical TrueType data sequence contains value stream that has the same value repeating periodically among other values, e.g. A, x1, A, x2, A, x3, etc. Each value is of type SHORT and occupies 2 bytes of storage. An additional set of HopCodes allows the reduction of the dataset size by applying the following transformation:

- **A, x1, A, x2, A** – translated to: **A, x1, Hop3Code, x2**. (Hop3Code = 0xFB);
- **A, x1, A, x2, A, x3, A** – translated to **A, x1, Hop4Code, x2, x3**, (Hop4Code = 0xFC).

4.2.4.3.5 Glyph Program Table

The following table describes the glyph program in CTF format:

Table 5 — Glyph Program Table

Data Type	Syntax	Description and Comments
255USHORT	pushCount	Number of data items in the push data stream.
255SHORT	pushData[pushCount]	See sample C function for HopCodes processing.
255USHORT	codeSize	Glyph program remaining code size.
BYTE	instructions[codeSize]	Remaining TrueType instructions.

4.2.4.3.6 Translation of HopCodes

The following C code can be used to translate sequence of HopCodes:

```
CHAR *input;
int i;
   BYTE code;
   SHORT A;
SHORT pushCount = Read255UShort(input);
SHORT *pushData  = (SHORT *) malloc(sizeof(SHORT) * pushCount);
for ( i = 0; i < pushCount; ) {
    code = *input;
    if (code == Hop3Code ) {
        A = pushData[i-2];
        input++;                    /* Hop3Code */
        pushData[i++] = A;              /* A */
        pushData[i++] = Read255Short(input); /* X2 */
        pushData[i++] = A;              /* A */
    } else if (code == Hop4Code ) {
        A = pushData[i-2];
        input++;                    /* Hop4Code */
        pushData[i++] = A;              /* A */
        pushData[i++] = Read255Short(input); /* X2 */
        pushData[i++] = A;              /* A */
        pushData[i++] = Read255Short(input); /* X3 */
        pushData[i++] = A;              /* A */
    } else {
        pushData[i++] = Read255Short(input); /* Data */
    }
}
```

© ISO/IEC 2004 – All rights reserved

Licensed to Brian Jung, ATSI-serial:5_324981, Downloaded 8/27/2011 4:03:19 PM. Single user licence only, Copying and networking prohibited.

4.2.4.4 CTF Glyph Data

The following table contains information that describes the glyph data in CTF format.

Table 6 — CTF Glyph Data Table

Data Type	Syntax	Description and Comments
SHORT	numContours	Number of contours in the glyph, also used as a flag.
	if (numContours < 0)	If numContours == -1, the glyph is composite glyph.
	{	
SHORT	xMin	Minimum X value for coordinate data
SHORT	yMin	Minimum Y value for coordinate data
SHORT	xMax	Maximum X value for coordinate data
SHORT	yMax	Maximum Y value for coordinate data
	/* Composite CTF Glyph Table */	
	} else {	
	if (numContours == 0x7FFF)	This is a flag, read new value and bounding box information below.
	{	
SHORT	numContours	Actual number of contours in the glyph.
SHORT	xMin	Minimum X value for coordinate data
SHORT	yMin	Minimum Y value for coordinate data
SHORT	xMax	Maximum X value for coordinate data
SHORT	yMax	Maximum Y value for coordinate data
	} else	Bounding box information omitted and should be computed from the glyph data coordinates.
	{}	
	/* Simple CTF Glyph Table */	
	}	

4.2.4.5 Simple CTF Glyph Data

A simple glyph defines all the contours and points that are used to create the glyph outline. Each point is presented in a (flag, xCoordinate, yCoordinate) triplet that is stored with the variable length encoding consuming 2 to 5 bytes per triplet. The flag value is always stored in the first byte of triplet and defines the format and properties of the point coordinates.

The most significant bit of the flag indicates whether the point is on- or off-curve point. The remaining 7 bits specify 128 possible combinations which define data format and value for X and Y coordinates of a point and specify the combination of the following properties:

- Total number of bytes used for this triplet (byte count);
- Number of bits used for describing X coordinate value (X bits);
- Number of bits for describing Y coordinate value (Y bits);
- An additional amount to be added to X coordinate value (delta X);
- An additional amount to be added to Y coordinate value (delta Y);
- The sign of X coordinate value (X sign);
- The sign of Y coordinate value (Y sign).

Each of the 128 combinations identify a unique set of these seven properties which have been assigned after careful consideration has been given to the typical statistical distribution of data found in TrueType files. As a result, the dataset size in CTF format is significantly reduced compared to the native TrueType format. However, the data is left byte aligned so that an Huffman entropy coder can be easily applied on the second stage.

© ISO/IEC 2004 – All rights reserved

Licensed to brian berg, ihhs1 order #_[illegible] Downloaded 6/4/2011 1:08 PM. Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1685
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Unlike the native TrueType formats that allows X and Y coordinates values be defined either in relative or absolute values, the CTF format always defines X and Y coordinates as relative values to the previous point. The first point is relative to (0,0). The information on data formats of xCoordinates[] and yCoordinates[] of glyph data can be found in the subclause 4.2.4.7 of this specification. This table describes simple glyph data in CTF format.

Table 7 — Simple CTF Glyph Table

Data Type	Syntax	Description and Comments
	/* Data */	
255USHORT	contourPoints[numContours]	Number of points in each contour, cumulative sum should be calculated to define total number of points.
BYTE	flags[numPoints]	Array of flags for each coordinate in glyph outline.
Variable length bit field	xCoordinates[numPoints]	Array of X coordinates for glyph outline points.
Variable length bit field	yCoordinates[numPoints]	Array of Y coordinates for glyph outline points.
	/* Glyph Program Table */	

4.2.4.6 Composite CTF Glyph Data

This table describes composite glyph data in CTF format.

Table 8 — Composite CTF Glyph Table

Data Type	Syntax	Description and Comments
	/* Data */	
	do {	/* for all glyph components */
USHORT	flags	Flags for each component in composite glyph.
	. . . . .	See "glyf" table specification for more information.
	} while (flags & MORE_COMPONENTS);	
	if (flags & WE_HAVE_INSTR) {	
	/* Glyph Program Table */	
	}	

The information on data formats of flags[], xCoordinates[] and yCoordinates[] of glyph data section can be found in the 'glyf' table of OpenType font format specification.

4.2.4.7 Triplet Encoding of X and Y Coordinates

The following table presents information on Triplet Encoding (flag[], xCoordinates[], yCoordinates[]) of simple glyph outline points.

Please note – "Byte Count" field reflects total size of the triplet, including 'flags'.

© ISO/IEC 2004 – All rights reserved

Licensed to Brian Story, ANSI Order # 2795249, Downloaded on 6/8/2007 1:22:22 PM. Single-user licensed only, copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1686
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Table 9 — Coordinate Encoding Table

Index	Byte Count	X bits	Y bits	Delta X	Delta Y	X sign	Y sign
0	2	8	0	0	N/A	+	N/A
1				0		-	
2				256		+	
3				256		-	
4				512		+	
5				512		-	
6				768		+	
7				768		-	
8				1024		+	
9				1024		-	
10	2	0	8	N/A	0	N/A	+
11					0		-
12					256		+
13					256		-
14					512		+
15					512		-
16					768		+
17					768		-
18					1024		+
19					1024		-
20	2	4	4	1	1	+	+
21					1	+	-
22					1	-	+
23					1	-	-
24					17	+	+
25					17	+	-
26					17	-	+
27					17	-	-
28					33	+	+
29					33	+	-
30					33	-	+
31					33	-	-
32					49	+	+
33					49	+	-
34					49	-	+
35					49	-	-
36	2	4	4	17	1	+	+
37					1	+	-
38					1	-	+
39					1	-	-
40					17	+	+
41					17	+	-
42					17	-	+
43					17	-	-
44					33	+	+
45					33	+	-
46					33	-	+
47					33	-	-
48					49	+	+
49					49	+	-
50					49	-	+
51					49	-	-
52	2	4	4	33	1	+	+
53					1	+	-
54					1	-	+
55					1	-	-
56					17	+	+

© ISO/IEC 2004 – All rights reserved

Downloaded by Orbit Study (MPEG-4 Part X_Study@) (Downloaded 6/2/2021 | Site Edit. Single user licence only. Copying and networking prohibited.

Index	Byte Count	X bits	Y bits	Delta X	Delta Y	X sign	Y sign
57					17	+	-
58					-17	-	+
59					-17	-	-
60					33	+	+
61					33	+	-
62					-33	-	+
63					-33	-	-
64					49	+	+
65					49	+	-
66					-49	-	+
67					-49	-	-
68	2	4	4	49	1	+	+
69					1	+	-
70					-1	-	+
71					-1	-	-
72					17	+	+
73					17	+	-
74					-17	-	+
75					-17	-	-
76					33	+	+
77					33	+	-
78					-33	-	+
79					-33	-	-
80					49	+	+
81					49	+	-
82					-49	-	+
83					-49	-	-
84	3	8	8	1	1	+	+
85					1	+	-
86					-1	-	+
87					-1	-	-
88					257	+	+
89					257	+	-
90					-257	-	+
91					-257	-	-
92					513	+	+
93					513	+	-
94					-513	-	+
95					-513	-	-
96	3	8	8	257	1	+	+
97					1	+	-
98					-1	-	+
99					-1	-	-
100					257	+	+
101					257	+	-
102					-257	-	+
103					-257	-	-
104					513	+	+
105					513	+	-
106					-513	-	+
107					-513	-	-
108	3	8	8	513	1	+	+
109					1	+	-
110					-1	-	+
111					-1	-	-
112					257	+	+
113					257	+	-
114					-257	-	+

© ISO/IEC 2004 – All rights reserved

Licensed to Brian Berg, ANSI order #_238020 Downloaded/Printed on 2011-9-22 PM. Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1688
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Index	Byte Count	X bits	Y bits	Delta X	Delta Y	X sign	Y sign
115					257	-	-
116					513	+	+
117					513	+	-
118					513	-	+
119					513	-	-
120	4	12	12	0	0	+	+
121						+	-
122						-	+
123						-	-
124	5	16	16	0	0	+	+
125						+	-
126						-	+
127						-	-

The coordinate values of simple glyph outline points are calculated as follows:

```
Xcoord[i] = (SHORT)((Xsign)*(xCoordinate[i] + DeltaX[index]);
Ycoord[i] = (SHORT)((Ysign)*(yCoordinate[i] + DeltaY[index]);
```

5 Font Data Stream

5.1 Structure of the Font Data Stream

Font data streams are streams of streamType FontDataStream 0x0C (see ISO/IEC 14496-1, Table 7), identified by a specific objectTypeIndication 0x06 (see ISO/IEC 14496-1, Table 6). A font data stream consists of one or more access units with font data.

5.2 Access Unit Definition

5.2.1 Overview

A font data access unit conveys data for one single font, or a subset thereof. Two access unit formats are specified, discriminated by the DecoderSpecificInfo for the font data stream (see subclause 5.4). Only one of these access unit formats can be selected for a given font data stream.

The basic access unit format permits to convey font data without any header information. The DecoderSpecificInfo provides the identification of the font and, possibly, the subset thereof that is conveyed. The content of subsequent basic access units in such a font data stream shall replace the font or font subset data of previous basic access units.

The enhanced access unit format provides a self-contained format where an access unit header contains all necessary information about the font data conveyed in this access unit.

Access units in font data streams shall be labelled and time-stamped by suitable means. This shall be done via the related flags and the composition time stamps, respectively, in the SL packet header (see ISO/IEC 14496-1, 10.2.4). The composition time indicates the point in time at which a font data access unit becomes valid, i.e., when the embedded font information becomes available for font rendering. Decoding and composition time for a font data access unit shall always have the same value.

For all basic access units and those enhanced access units that have `fontSubsetExtensionFlag` set to '0' the `randomAccessPointFlag` in the SL packet header shall be set to '1' for this access unit. Otherwise, the `randomAccessPointFlag` shall be set to '0'.

© ISO/IEC 2004 – All rights reserved

Licensed to Brian Berg, ATSI member, 2009-03. Downloaded 6/21/2011 6:02 PM. Single user licence only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1689
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

5.2.2 Syntax

```
class BasicFontAccessUnit() {
   bit(8)  fontData[sizeOfInstance];
}

class EnhancedFontAccessUnit() {
   bit(7)  fontFormat;
   bit(1)  storeFont;
   bit(8)  fontNameLength;
   bit(8)  fontName[fontNameLength];
   bit(7)  fontSubsetID;
   bit(1)  fontSubsetExtensionFlag;
   bit(8)  fontSpecInfoLength;
   bit(8)  fontSpecInfo[fontSpecInfoLength];
   bit(8)  fontData[sizeOfInstance - fontNameLength - fontSpecInfoLength - 5]
}
```

5.2.3 Semantics

`fontFormat` -- indicates the format of the access unit as follows.

Table 10 — `fontFormat` values for EnhancedAccessUnit

`fontFormat`	Access unit content
0x00	Forbidden
0x01	OpenType with TrueType outlines (uncompressed)
0x02	OpenType with TrueType outlines compressed using the compression mechanism described in subclause 4.2
0x03	OpenType CFF
0x04 – 0x3F	ISO reserved
0x40 – 0x7F	User private

`storeFont` -- if set to '1', the terminal shall store the font information contained within the access unit at least until the end of the current session. Otherwise, the font information becomes unavailable as soon as the font data stream becomes unavailable.

NOTE — Details of persistent storage are beyond the scope of this specification.

`fontNameLength` -- specifies the length in characters of the `fontName` field.

`fontName` -- a Unicode (ISO/IEC 10646-1) encoded string that indicates the name of the font.

`fontSubsetID` -- specifies a numerical ID for a subset of the font conveyed in this access unit. If `fontSubsetID` equals 0x00, the access unit contains the complete font data. The value of 0x7F shall not be used.

`fontSubsetExtensionFlag` -- if set to one indicates that the font data in this access unit is an extension of the previously sent font subset with the same `fontName` and `fontSubsetID`. Otherwise the font subset replaces any previous font subset with the same `fontName` and `fontSubsetID`.

`fontSpecInfoLength` - defines the length in bytes of the `fontSpecInfo` field.

`fontSpecInfo` -- is an opaque container with information for a specific font handler.

`fontData` -- contains the font data in the specified `fontFormat`.

© ISO/IEC 2004 -- All rights reserved

Licensed to [illegible] Downloaded 6/2/2021 | 5:22 PM. Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1690
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

5.3 Time Base for Font Data Streams

The time base associated with a font data stream shall be indicated by suitable means. This shall be done by the use of object clock reference time stamps in the SL packet headers (see ISO/IEC 14496-1, 10.2.4) for this stream or by indicating the elementary stream from which this font data stream inherits the time base (see ISO/IEC 14496-1, 10.2.3). All time stamps in the SL-packetized font data stream refer to this time base.

5.4 Font Data Decoder Configuration

5.4.1 Syntax

```
class FontDataDecoderConfiguration extends DecoderSpecificInfo  : bit(8)
tag=DecSpecificInfoTag {
   bit(7)  fontFormat;
   if (fontFormat != 0x00) {
      bit(1)  storeFont;
      bit(8)  fontNameLength;
      bit(8)  fontName[fontNameLength];
      bit(7)  fontSubsetID;
      bit(1)  reserved = 1;
      bit(8)  fontSpecInfo[sizeOfInstance - fontNameLength - 4];
   }
}
```

5.4.2 Semantics

`fontFormat` – indicates the format of the access unit as follows.

Table 11 — `fontFormat` values for FontDataDecoderConfiguration

`fontFormat`	Access unit content
0x00	enhanced access unit format
0x01	OpenType with TrueType outlines (uncompressed)
0x02	OpenType with TrueType outlines compressed using the compression mechanism described in subclause 4.2
0x03	OpenType CFF
0x04 – 0x3F	ISO reserved
0x40 – 0x7F	User private

A `fontFormat` of 0x00 indicates the use of the enhanced access unit format for this stream. In this format, all information about the font data is conveyed within each access unit.

`storeFont` – if set to '1', the terminal shall store the font information contained within the access unit at least until the end of the current session. Otherwise, the font information becomes unavailable as soon as the font data stream becomes unavailable.

NOTE — Details of persistent storage are beyond the scope of this specification.

`fontNameLength` – specifies the length in characters of the `fontName` field.

`fontName` – a Unicode (ISO/IEC 10646-1) encoded string that indicates the name of the font.

`fontSubsetID` – specifies a numerical ID for a subset of the font conveyed in this access unit. If `fontSubsetID` equals 0x00, the access unit contains the complete font data. The value of 0x7F shall not be used.

`fontSpecInfoLength` - defines the length in bytes of the `fontSpecInfo` field.

`fontSpecInfo` – is an opaque container with information for a specific font handler.

© ISO/IEC 2004 – All rights reserved

Licensed to Chun Sung, ANSI Order #_____ Downloaded 6/20/2011 8:22 PM. Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1691
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

5.5 Accessing the Font Data

A font presented in either `BasicFontAccessUnit` or `EnhancedFontAccessUnit` of a `FontDataStream` can be identified by a number of different attributes. One common identifier for fonts is the font name. However, a font can be presented either as a whole font, or as an arbitrary subset of an original font. In both cases the font name will be the same while the actual font data included in access unit may vary. In order to explicitly identify a font (or a subset thereof), the following syntax shall be used as the font name:

"`<fontName>[;OD:<odid>][;FSID:<fsid>]`", where :

- `<fontName>` is a Unicode encoded string that indicates the name of the font, as conveyed by `fontName` field in either `BasicFontAccessUnit` or `EnhancedFontAccessUnit` of associated font data stream;
- `<odid>` is the numeric value of the `objectDescriptorID` of the associated font data stream;
- `<fsid>` is the numeric value of the requested font (or a font subset) as conveyed by `fontSubsetID` within the `BasicFontAccessUnit` or `EnhancedFontAccessUnit` of associated font data stream.

For example, a fontName `"Times New Roman; OD:16; FSID:11"` would indicate that a font data presented in the font data stream associated with the `objectDescriptorID` = 16 and is conveyed by the `FontAccessUnit` with `fontSubsetID = 11 shall be used, or, if font data stream is not present, a` font `"Times New Roman"` shall be used for text rendering, if available in the MPEG-4 terminal.

6 Text Profiles and Levels

6.1 Simple Text Profile and Levels

6.1.1 Application area

Proposed structure of the Simple profile is designed to satisfy the needs of the majority of languages that do not require support for complex script processing and advanced typographic features. It's also aimed to minimize the implementation complexity of the terminals, limiting font rendering capabilities to fonts with TrueType outlines only. These limitations allow using all existing TrueType fonts and OpenType fonts with TrueType outlines containing the set of required tables only and embedded bitmaps. The functionality of the Simple profile can be implemented utilizing previously defined **Text** and **FontStyle** BIFS nodes and does not require the use of new **XFontStyle** node (see ISO/IEC 14496-11 for details).

6.1.2 List of tools/functionalities

The Simple Text Profile represents the collection of tools that are required to guarantee the predictable and faithful reproduction of text objects utilizing TrueType fonts and simple language scripts (e.g. Western European and some Asian languages). The profile also provides tools for font compression and streaming.

6.1.3 Comparison with existing profiles

The Simple Text Profile is the lowest level profile defined for text support. It requires support for rendering of fonts with TrueType outlines, which is available on the majority of software platforms and operations systems.

6.1.4 Simple Text Profile @ Level 1 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines (only required tables defined in the OpenType specification and embedded bitmaps are supported).

© ISO/IEC 2004 – All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1692
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.1.5 Simple Text Profile @ Level 2 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines (only required tables defined in the OpenType specification and embedded bitmaps are supported);
- Streaming of uncompressed fonts only.

6.1.6 Simple Text Profile @ Level 3 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines (only required tables defined in the OpenType specification and embedded bitmaps are supported);
- Streaming of compressed and uncompressed fonts.

6.2 Advanced Simple Text Profile and Levels

6.2.1 Application area

Proposed structure of the Advanced Simple Text Profile is designed to satisfy the needs of those languages that either require or would benefit from the support for advanced typographic features (e.g. stylistic alternatives and historical forms in Latin languages, support for both lowercase and small cap characters, support for ligatures and glyph substitution and positioning, etc.). It's also aimed to reduce the implementation complexity of the terminals, limiting font rendering capabilities to support TrueType outlines only. This limitation allows using all existing TrueType fonts and OpenType fonts with TrueType outlines.

6.2.2 List of tools/functionalities

The Advanced Simple Text Profile represents the collection of tools that are required to guarantee the predictable and faithful reproduction of text objects utilizing OpenType fonts with TrueType outlines and complex language scripts (e.g. Arabic and Indic scripts, as well as many other language scripts). The profile also provides tools for font compression and streaming.

6.2.3 Comparison with existing profiles

The Advanced Simple Profile extends the Simple Text Profile by adding the capability to support advanced text layout and advanced typographic features (e.g. bi-directional text rendering, ligatures, glyph substitution and positioning, etc.)

6.2.4 Advanced Simple Text Profile @ Level 1 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines;
- Support for OpenType Layout and advanced features.

6.2.5 Advanced Simple Text Profile @ Level 2 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines;
- Support for OpenType Layout and advanced features;
- Streaming of uncompressed fonts only.

6.2.6 Advanced Simple Text Profile @ Level 3 Definition

- Support for resident fonts;

© ISO/IEC 2004 – All rights reserved

Licensed to Sean Barry, ISO Store, 1_Seat2F. Network Licence, 2004/12/07. Single user licence only, Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1693
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

- Rendering of OpenType fonts with TrueType outlines;
- Support for OpenType Layout and advanced features;
- Streaming of compressed and uncompressed fonts.

6.3 Main Text Profile and Levels

6.3.1 Application area

Proposed structure of the Main Text Profile is designed to satisfy the needs of those languages that either require or would benefit from the support for advanced typographic features. It also allows using all existing TrueType fonts and OpenType fonts with both TrueType and CFF outlines.

6.3.2 List of tools/functionalities

The Main Text Profile represents the collection of tools that are required to guarantee the predictable and faithful reproduction of text objects utilizing OpenType fonts with either TrueType or CFF outlines and complex language scripts (e.g. Arabic and Indic scripts, as well as many other language scripts). The profile also provides tools for font compression and streaming.

6.3.3 Comparison with existing profiles

The Main Text Profile extends the Advanced Simple Text Profile by adding the capability to render OpenType fonts with CFF/Type 1 outlines with full support for advanced text layout and advanced typographic features (e.g. bi-directional text rendering, ligatures, glyph substitution and positioning, etc.)

6.3.4 Main Text Profile @ Level 1 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType and CFF outlines;
- Support for OpenType Layout and advanced features.

6.3.5 Main Text Profile @ Level 2 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType and CFF outlines;
- Support for OpenType Layout and advanced features;
- Streaming of uncompressed fonts only.

6.3.6 Main Text Profile @ Level 3 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType and CFF outlines;
- Support for OpenType Layout and advanced features;
- Streaming of compressed and uncompressed fonts.

© ISO/IEC 2004 – All rights reserved

Licensed to Brian Berg, ANSI online 2_2/9/20. Downloaded 8/23/2011 8:22 PM. Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1694
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Annex A
(informative)

Patent Statements

The International Organization for Standardization and the International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this part of ISO/IEC 14496 may involve the use of patents.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patents right are registered with ISO and IEC. Information may be obtained from the companies listed below.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 14496 may be the subject of patent rights other than those identified in this annex. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Company
Agfa Monotype Corporation
Apple Computer, Inc.
IBM
Microsoft Corporation

© ISO/IEC 2004 – All rights reserved

Downloaded by Guest Guest ATRI Member X_328328 Downloaded 8/2/2020 1:02:52 PM. Single user licence only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1695
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Licensed to Global Sony ANSI Order # 0 Downloaded 6/22/2011 1:22 PM. Single user license only. Copying and networking prohibited.

ICS 35.040
Price based on 18 pages

© ISO/IEC 2004 – All rights reserved

Licensed to Brian Berg, AVSI online X_636326. Downloaded 6/2/2011 6:02 PM. Single user license only. Copying and networking prohibited.

I·O DATA

Japanese Quality Control

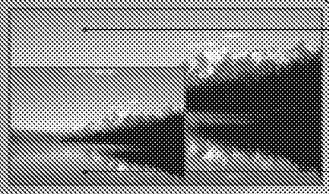
Rhapsody

Innovation technology arrived

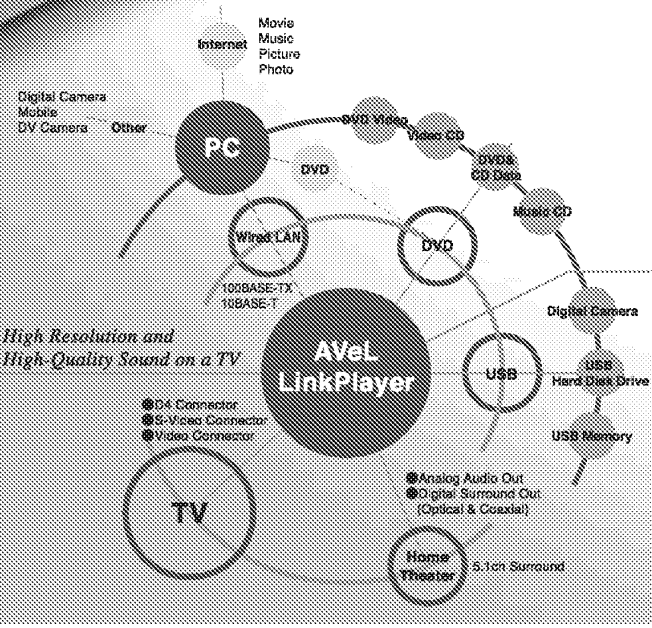
AVeL Link player

- DivX® HD Certified, MPEG2-TS & AVI2.0 Supported
- Microsoft WMV9(Windows Media® Video 9) & PlaysForSure Certified
- Apple iPod or USB 2.0/1.1 storage devices compatible to playback

The quality of HD mode is more advanced than DVD image



USB 2.0/1.1 supported

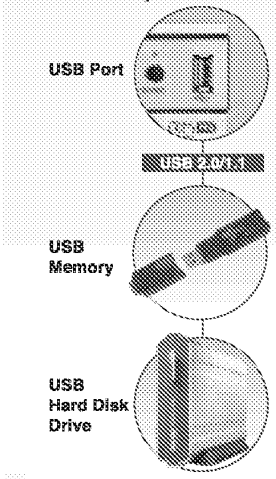


High Resolution and High-Quality Sound on a TV

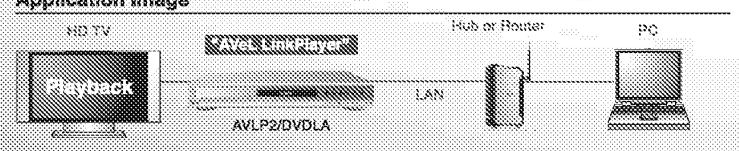


Now you can enjoy movies on your TV from your PC with the Intel® Pentium® 4 Processor with HT Technology and the "AVeL LinkPlayer" digital media adapter. Tap into your PC's digital music and memories, then enjoy them on your TV from any room in your home.

Direct Playback to connect USB memory & USB Hard Disk

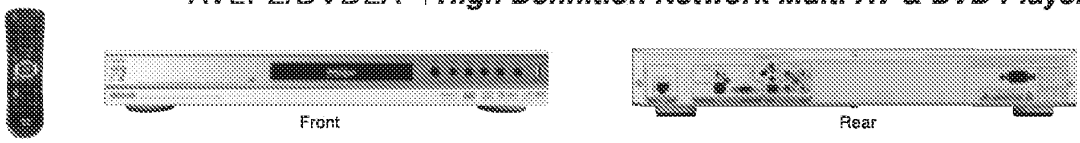


Application Image



DVD Drive Built-in Network Media Player

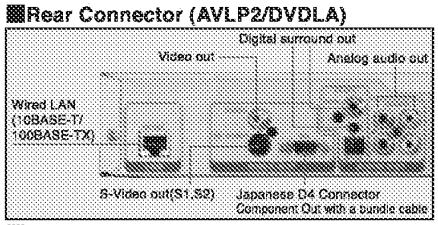
AVLP2/DVDLA | ***High Definition Network Multi AV & DVD Player***



Interface

Video out	Composite video out×1 S-video out(S1, S2)×1	
	D4 connector×1	1080i, 720p, 480p, 480i
Audio out	Analog audio out×2, Optical digital Out ×1, Digital Surround Out(Coaxial)×1	
Network (Wired)	Conforming standards	100BASE-TX / 10BASE-T(RJ-45)×1
	IP address	Auto/Manual selectable
USB port (USB 2.0/1.1 supported)	Spec	USB Specification Ver.2.0(High Speed, Full Speed)
	Connector	1 connector, USB A Type
	Supported devices	Mass storage class (USB hard disk, USB memory, USB memory card reader/writer)
	Supported file system	FAT(16), FAT32

Rear Connector (AVLP2/DVDLA)



Supported Formats and Maximum Bit Rate

	Format	Wired LAN	DVD	CD	USB 2.0	USB 1.1
Video formats※1	MPEG-1	12Mbps	10Mbps	4Mbps	10Mbps	3Mbps
	MPEG-2					
	MPEG2-TS※2					
	DivX® Video	1.5Mbps	1.5Mbps	1.5Mbps	1.5Mbps	1.5Mbps
	XviD					
	WMV9	8Mbps	8Mbps	4Mbps	8Mbps	3Mbps
Limitations on Video file size	AVI(DivX® Video, XviD)	2GB				
	AVI 2.0(DivX® HD, XviD)※2	4GB if the disk format is FAT32 or 2 TB if the disk format is NTFS※3				
	MPEG					
	WMV9					

Audio formats	Format	Bit rate	Sampling rate
	MP3	64K~192Kbps	32K / 44.1K / 48KHz
	WMA※2		44.1K / 48KHz
	PCM(WAVE)	Up to the sampling rate	32K / 44.1K / 48KHz
	Ogg Vorbis	64K~192Kbps	—
	AAC		32K / 44.1K / 48KHz

Image formats	Format	Maximum resolution	Display method	Note
	JPEG	2048×1532	Network, CD, DVD, USB, UHDL	Maximum 1600×1200※4 on CD / DVD / USB / UHDL
	BMP	2048×1532	Network※5	Non support on CD / DVD / USB / UHDL
	GIF			
	PNG			

※1 AVI files can be played only when they are interleaved. If a non-interleaved file is played, either video or audio will not be played properly. Make sure the file you want to play is interleaved. For interleaving of a file during the file creation process, refer to the Help screens or manual of the encoding application you are using.
※2 MPEG2-TS, AVI2.0, DivX® HD DVD, WMA Pro will be supported by firmware update. The release date of the supported firmware will be announced on our web.
※3 Varies in accordance with the format specification of the disk in which the file is saved.
※4 Maximum resolution for progressive JPEG is 1280×720. ※5 Network refers to wired LAN connection.
※6 Some media may not be playback due to a quality of media, copyright-protection and so on.
※7 Basically the "AVeL LinkPlayer" can play DVD-ROMs and DVD-Videos created on DVD-R, DVD-RW, DVD+R and DVD+RW media. Note that some media may not be played properly. Only UDF/ISO Bridge and ISO 9660 are supported. UDF and PacketLight are not supported.
※8 Apple PowerMac series are supported, which OS are compatible.

DVD Drive

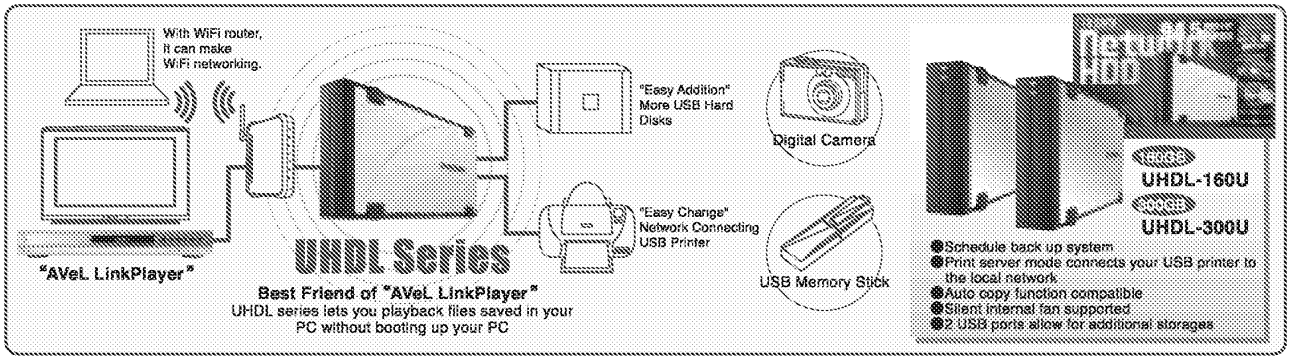
Supported Media※6	DVD+R / DVD+RW / DVD-R / DVD-RW / CD-R / CD-RW ※Supported disk size 8cm / 12cm
Supported Media formats※7	DVD-Video / CD-ROM / DVD-ROM / CD-DA / Video-CD / DivX® HD DVD※2
Playback function	Play / Pause / Stop / Forward / Reverse / Next chapter / Previous chapter Select title chapter, Time search, Repeat, Slow, Multi angle Language, Subtitle
Audio playback function	Stereo PCM / Dolby Digital(AC-3)
Region code	1

Specifications

Supported models	Windows machines Apple PowerMac series※8
Supported OS	Windows XP, Windows Me, Windows 2000 Professional, Windows 98 Second Edition, MacOS X 10.3
Recommended system	CPU : Intel Pentium Ⅲ 500MHz or faster Memory : 128MB or over
Manual operation button	Power / Eject / Play / Pause / Stop / Forward / Reverse / Next / Previous
Dimensions	430(W)×291(D)×55(H)mm(excluding projections)
Weight	Approx. 3.5kg(main unit only)
Power Supply	AC100V(50 / 60Hz) up to 240V
Bundle items	Remote controller×1, AAA battery×2, AV cable(composite & stereo), LAN cable×1, Support soft(CD-ROM), Component conversion cable
Warranty	1 year

●Apple, Macintosh, PowerMacintosh and Mac are registered trademarks of Apple Computer, Inc. in the United States.
●DivX and the DivX logo are registered trademarks of DivXNetworks, Inc. in the United States and other countries.
●Microsoft, Windows, Windows Media Player, and PlaysForSure are registered trademarks of Microsoft Corporation in the United States.
●Intel and Pentium are registered trademarks of Intel Corporation in the United States.
●Other company names and product names mentioned in this catalog may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

Network Hard Disk UHDL series (Option)



For more information

Web Site **http://www.iodata.com/usa/**

E-Mail **info@iodata.com**

I-O DATA DEVICE USA, INC.

●I-O DATA and I-O DATA logo are registered trademarks of I-O DATA DEVICE, INC.
●Actual colors of products may differ relative to filming or when printed.
●This product is not intended for use as medial equipment, nuclear facility or equipment, aerospace equipment, transportation facility or equipment, weapon system or other facility or equipment that effects human life; undersea relay, space satellite or other facility or equipment that requires high levels of reliability; or any component thereof.
●Information on this catalog is as of November 2004. Product specifications, design and appearance may change without prior notice.

I-O DATA

Japanese Quality Control

I-O DATA DEVICE, INC. is a top manufacturer and provider of high-quality computer peripherals and interface products, representing the finest technology Japan offers. Publicly listed in Japan, we're found globally with 11 sales offices, over 5000 retail stores, and 5 international group companies. We are also ISO9001/14001 certified. We invite you to experience our ability to think beyond convention.

http://www.kiss-technology.com/?p=dp500

KiSS DP-500

Summary

The DP-500 is based on the ground breaking DP-450 with its DivX®, XviD, Mpeg 4, MP3, Ogg Vorbis reading capabilities. An Ethernet 10/100 port has been added to the player along with a Sigma Designs EM8550 progressive DVD decoder chip which enable any user with a Broadband Internet Connection to stream DVD quality movies directly from a Computer. To enhance the user's experience and to take the best out of this connection, the DP-500 ships with the KiSS PC-Link software which makes possible to read any of the supported file format straight from your PC to your DVD player and your TV via a straight forward and user-friendly interface. To change your DP-500 experience from exceptional to unforgetable, KiSS introduces WebRadio. Accessing the WebRadio menu on your Player will provide you with a high number of radio stations, more than a thousand, that can be streamed from the internet to your DVD player and your TV. To make this session comfortable and according to your own taste, once registered, WebRadio will allow you to choose, customize, add as many radio station as you wish. The KiSS DP-500 also comes complete with the standard CD/MP3, Ogg Vorbis, DivX®, XviD, CD-RW and DVD+-RW playback. Regular firmware updates for this product are available on www.kiss-technology.com.

watch out for

ETHERNET CONNECTION
The big new feature with KiSS DP-500 is the ethernet connection found on its back. The KiSS Player can be connected through different ways to allow all its features and services to run. It can be either connected directly to the internet via a broadband connection or through a network (LAN) with a PC holding contents that can be read using KiSS PC-Link.

WEBRADIO
Thanks to its ethernet connection, the DP-500 is the only player on the market to provide users with a direct connection to the Internet. This internet connection enables the KiSS DP-500 to play internet radio. Just select this feature on the Player, and you will have access to a few thousand radio stations that can be streamed and played on the DP-500.

KISS PC-LINK
KiSS PC-Link is a software that enables the KiSS DP-500 user to play music, see pictures and view movies from a PC to the DivX compatible DVD player thanks to the ethernet connection found on the back of the player. KiSS PC-Link is a user-friendly software to install on a PC running at least Windows 98, it easy to select any media files on a PC's hard disk.

BEST EVER DivX® COMPATIBILITY
KiSS DVD players play all DivX format. DivX® 3.11, DivX® 4 and DivX® 5. What does this mean for you? It means that now, for the first time, you can download full-screen, full-motion videos from the Internet that actually look and sound like what you get from your television or DVD player. With DivX® compression, you can download and watch high-quality video on your TV.

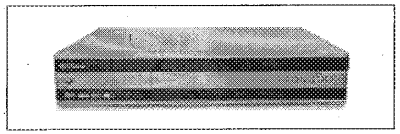
5 STEP QUICK SETUP
To tackle long and too technical setup problem, KiSS introduces a Five Step Quick Setup to it DP-500 model. As soon as the player is plugged-in for the first time, a clear and user friendly graphic interface asks the user to select its language, video connection, screen size, audio connection and enter its IP address. Once the selection has been made, the player is ready to be used.

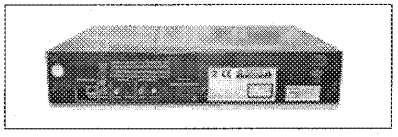
FIRMWARE UPGRADABLE
KiSS is keen in keeping a special relation with its customers, so, even after a player has been sold, its software and technology keeps moving. The firmware of the player keeps updated with the latest research and development and KiSS is able to release a firmware update on a regular basis. This latest firmware is available from our web site in the support section.



KiSS DP-500

front and back





specifications

- Full DVD/MPEG-4/DivX®/XviD/CD/MP3/Ogg Vorbis/CD-RW/DVD+/-RW compatibility
- JPEG Picture CD viewing
- Progressive Scan Video Out
- Time Search
- Repeat Playback
- Title/chapter select
- Parental lock, Multi Language, Multi Subtitle, Multi Angle
- Remote Control
- TV Type (PAL and NTSC)
- Configuration Setup Menu
- Aspect Ratio (4:3 Pan and Scan, 4:3 Letter Box and 16:9 Wide)
- S-Video Output
- Composite Video Output
- Component Output
- Dolby Digital, MPEG Audio Decoding
- Dolby Digital/MPEG 5.1 Digital Out, S/PDIF Outputs
- Analogue stereo
- Ethernet 10/100
- WebRadio
- KiSS PC-Link
- KiSS PC-Link Software for PC
- Track name search on audio CDs
- Firmware Upgradable

	DVD	DivX 3.11, 4 AND 5	XviD	MPEG-1, 2 AND 4	MP3, OGG VORBIS	AUDIO CD	JPEG PICTURES	DIVX CERTIFIED	PROGRESSIVE SCAN	ETHERNET	WIRELESS CONNECTION	DIGITAL AMPLIFIER	FM/AM RADIO	KISS PC-LINK	WEBSOUND	40GB HARD DISK	80GB HARD DISK	HARD DISK RECORDER	TV TUNER	SLIMLINE DESIGN	FIRMWARE UPGRADABLE
KiSS DP-500	●	●	●	●	●	●	●	●	●	●		●		●							●

Official DivX Certified™ product • Plays DivX® 5, DivX® 4, DivX® 3, and DivX® VOD video content in compliance with DivX Certified™ technical requirements

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1700
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

MATROŠKA

Main Menu

- Home
- What is Matroska?
- Downloads
- News
- Guides
- FAQ
- Technical / Info
  - Diagram
  - Specifications
  - Specification Notes
  - Order Guidelines
  - Codec Specs
  - Chapters
  - Subtitles
  - Tags
  - Cover Art
  - **Streaming**
  - Menu
  - Overhead
  - EBML RFC
  - SVN Repository
  - Team
  - License
  - Contributions
- Project Page
- Blog
- Forum

Contact

- Legal
- Logos / Trademarks
- Contact
- Sponsors

Matroska Streaming

There exist multiple ways to stream a content. The term streaming itself is very vague. It means reading a file stored on a server. But the server could be very distant or very close. The transport system and the protocol used for streaming makes the whole difference.

In the case of Matroska, there are mostly 2 different kinds of stream: file access and live streaming.

File Access

File access can simply be reading a file located on your computer, but also accessing it from an HTTP (web) server or CIFS (windows share) server. All these protocol are usually safe from reading errors and seeking in the stream is possible. On other hand when the file is stored far or on a slow server, seeking can be an expensive operation and should be avoided. That's why we set a few guidelines that, when followed, help reduce the number of seeking for regular playback and also have the playback start quickly without a lot of data needed to read first (like the Cues (index). Attachments or Meta Seek of all the Clusters).

Matroska having a small overhead, it is well suited for storing music/videos on file servers without having a big impact on the bandwidth used. It doesn't require to load the index before playing (the index can be loaded only when seeking is requested the first time), so playback can start very quickly too.

Live Streaming

Live streaming is the equivalent of TV broadcasting on the internet. There are 2 families of servers for that. The RTP/RTSP ones and the HTTP servers. Matroska is not meant to be used over RTP. RTP already has timing and channel mechanisms that would wasted if doubled in Matroska. On the other hand live streaming of Matroska over HTTP (or any other plain protocol based on TCP) is very possible.

A live Matroska stream is different than a file, because it may have no known end (only when the client disconnects). For that the

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1701
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Segment *must* use the "unknow" size (all 1s in the size). The other option would be to concatenate Segments with known sizes one after the other. This solution allows a change of codec/resolution between each segment which can be useful in some cases (switch between 4:3 and 16:9 in some TV programs for example).

The Segment(s) being continuous, certain elements like Meta Seek, Cues, Chapters, Attachments *must* not be used in this context.

On the player side, it is possible to detect that a stream is not seekable. If the stream does not have a Meta Seek list or a Cues list at the beginning of the stream, it *should* be considered as non seekable. Even though it's still theoretically possible to seek blindly forward in the stream, if the server supports it.

In the context of a live radio or even web TV it should be possible to "Tag" the content that is currently playing. The Tags level 1 element can be placed between Clusters each time necessary. In that case, the new Tags found *must* reset the previously encountered tags and use the new values instead (be they empty).

Copyright © 2005 - 2010 Matroska (non-profit org). All Rights Reserved.
XHTML & CSS Valid.

BNSDOCID: <XP_____55245997A__1_> 101217114310/http://matroska.org/technical/streaming/index.html

New User? Register Sign In Help Yahoo! Mail

YAHOO! NEWS

Search

TECH

Video Internet Gadgets Cell Phones Apple Social Media Security Linux/Open Source Gaming Apps

Search All News

A NEWS BLOG
TODAY IN TECH

Netflix turns on subtitles for PC, Mac streaming

39 10 Email Print

RELATED QUOTES

^DJUSS	489.02	+12.38
^IXIC	2,479.85	+66.32
^IXK	1,208.56	+32.57

Finally addressing one of the biggest pet peeves about its otherwise stellar "Watch Instantly" service, Netflix announced late Thursday that it's at last rolling out subtitles on its streaming movies and TV shows, with about 100 titles now boasting closed captioning.

In a blog post Thursday night, Netflix Chief Product Officer Neil Hunt admitted that there's only a "limited library" of subtitled streaming videos for now, including "most episodes" of the first season of "Lost", also, subtitles are only available on PC and Mac, with closed captions for game console, Blu-ray, and HDTV clients due in the fall. However, Hunt wrote, "we now have released the technology, and we will be working to fill in the library over time."

You'll find the button for subtitles in the bottom right corner of the streaming Netflix Web client, between the volume and full-screen controls. The subtitles I watched for the "Lost" pilot were big, yellow and bold, and included descriptions for key audio cues ("dog barking," "woman screaming," and so on).

Netflix has been taking flak from deaf and hard-of-hearing advocates for months over the lack of subtitles in its "Watch Instantly" library, most recently after a highly publicized free showing of "The Wizard of Oz" last October to celebrate the movie's 70th anniversary.

Both YouTube and Hulu have been offering subtitled videos for many moons now, although it's worth noting that the videos on those sites are powered by Flash, while Netflix uses Microsoft's Silverlight technology.

Hunt wrote last June that the tools for rendering optional subtitles in Silverlight and for consumer electronics devices such as game consoles and Blu-ray players were "weak or nonexistent, and there is some technology development required." Hunt promised that subtitled Netflix videos for Silverlight would be ready "sometime in 2010," and it looks like he's delivered (if on a limited basis, anyway).

Subtitles aren't the only item on Netflix's "Watch Instantly" to-do list; 5.1 audio tracks for streaming titles are also in the works and should arrive in the fall, Hunt wrote, along with streaming closed captions for Netflix-enabled HDTVs, game consoles, and other devices.

Netflix: Subtitles Now Available for Some Titles for PC/Mac Viewing

--- Ben Patterson is a technology writer for Yahoo! News.

Follow me on Twitter!

Follow Yahoo! News on Twitter!

Follow Yahoo! News on Twitter, become a fan on Facebook

39 10 Email Print

Top Stories

Flood worries rise with w
Damage from Irene appr
feared
Airlines, rail, struggle in s
Poker-faced Bernanke bi
options
Lockerbie bomber "al de
report
Analysis: Economic lead
France eyes financial sa

Advertis


Penny stoc
buying hap
recession.
Awesome P


Texas Dem
mus anti ag
leaked...
Stages Of Re


Shocking d
researcher
relief...
Instaflex

sponsored links

Get All 3 Credit Report
View Your Credit Report
Bureaus in 60 seconds.
FreeScoreOnline.com

Man Cheats Credit Sco
He Added 128 Points To
Using This 1 Easy Tip
www.CreditRepair.Fromi

Man "Cheats" Credit S
News: He Added 183 Po
Score Using This 1 Easy
www.TheCreditSolution.F

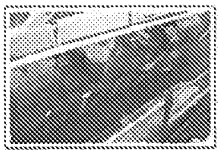
Yahoo! News on Fa

Like 115,026 likes. 7
friends like.

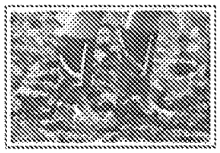
Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1703
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Editor's Picks

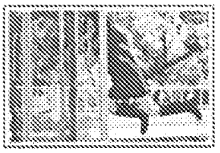
More Slideshows »



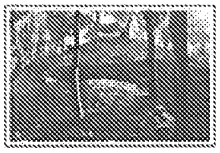
Confederate sub upright for first time since 1864



Crowds brave rain, mud at Glastonbury



Defying gravity: Tokyo photographer 'levitates'



Look inside ancient Mayan ruler's tomb

More on Gadgets & Personal Technology

Will Amazon's Tablet Kill the iPad?

Can Obama Save Android?

Best HP TouchPad Apps

More »

More...

Technology Video: TechBytes (08.26.11)

Technology Video: Emotions Run Wild With Steve Jobs Resignation

Technology Video: Steve Jobs resigns from Apple

4 Comments

Show: Highest Rated

Post a Comment

Vikingscout | Report Abuse 1 0

Whooo! I am happy for PC users, but I'm Wii user, just need to be closed-captioning because I am deaf. Hopefully Netflix will promising will be standing for the Wii, PS3, and HDTV.

Reply

Sean | Report Abuse 0 0

is there a plan to have subtitles on Roku as well?

Reply

Neil Kratt | Report Abuse 0 0

until there is 5.1 I'm not using it

Reply

Jayf | Report Abuse 0 0

When are they going to stream to linux?

Replies (0)

Post a Comment

Sign in to post a comment, or Sign up for a free account.

Featured



Upgrade
Sometimes
last year's
» More opti



'Secret li
What is yo
slips out of
» More vide



American
Five myths
Spangled B
» More fron



Top Stori
Are you on
page for to
» Join our F

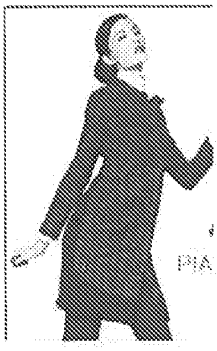
Education



Six quick st
With the rig
could be pr
career in 1-



What's the
now?
See how yo
career in thi

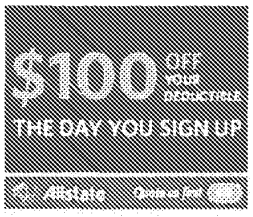


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1704
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Most Viewed - Technology

Oil wavers on debt ceiling, tropical storm

All Most Viewed »



Elsewhere on the Web

Mashable: Twitter Boasts That It's Faster Than Earthquakes [VIDEO]

Mashable: Facebook Kills Off Deals, Its Groupon Competitor

Mashable: New Apple CEO Tim Cook Gets $383 Million Stock Bonus — in 10 Years

XP-002179318

P.D. 00-09-1997
F. 1-42 (42)

OpenDML AVI File Format Extensions

Version 1.02

OpenDML AVI M-JPEG File Format Subcommittee

Last revision: February 28, 1996
Reformatting: September 1997

BNSDOCID: <XP_____2179318A_1_>

Please note

The information provided by this document is provided on an "as is" basis. No responsibility is assumed by Matrox Electronic Systems Ltd. for its use; nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of Matrox Electronic Systems Ltd.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1707
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Table of Contents

1.0 Introduction 1
Scope 1
OpenDML Sub-Committee on File Format: High Level Goals 1
Version History 1
1.0 Overview of Profession Video Requirements 2
Goal 2
Issues 2
AVI: Support Current AVI Format 2
60 Fields Per Second vs. 30 Frames Per Second; or 24 Frames Per Second For Film 2
Improved Tombstone and Header Information 2
2.0 Increased AVI File Size 3
Introduction: RIFF Chunk Format 3
Increased File Size Limits (>> 1 GB) 3
Extension for File Size > 1 GB 3
3.0 Frame and Field Indexing 5
Introduction: 'idx1' Index Chunk 5
Issues 5
Allow Field Indexing 5
Allow Incremental Growth of Files 5
Minimize Disk Seeks 5
Proposed Index List 6
Overview 6
Base Index Form 'indx' 6
AVI Standard Index Chunk 7
AVI Field Index Chunk 8
AVI Super Index Chunk 9
Index Locations in RIFF File 10
4.0 Interleaving Audio with Video 13
Issues 13
Interleaving Rate 13

BNSDOCID: <XP_____2179318A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1708
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Single or Multiple Files for Video and Audio 13
Multiple Tracks of Audio Storage 13
Mono and Stereo Audio 13
Audio / Video Time Stamping (Sync Point List) 13
Notes on Interleaving Audio and Video 13
5.0 Source and Header Information Storage **15**
Goal 15
Required Information 15
Main AVI Header (avih) 15
AVI Stream Header (strh) 15
AVI Stream Format (strf) 16
Extended AVI Header (dmlh) 16
Video Properties Header (vprp) 16
Source and Timecode Information 20
Film Transfer Log Information 22
Microsoft-Defined Tombstone Data 23
6.0 Other File Issues **28**
Issues 28
Continuation Over Multiple Files 28
Minimized Time to Open Files 28
Simultaneous Reading and Writing / Multiple Access 28
7.0 MJPG Motion JPEG DIB Extensions **30**
Issues 30
ISO vs. Non ISO Formats 30
8-bit vs. 10-bit YUV 30
Video vs. Bitmap 30
Proposed Data Chunk Format 30
From the DIB spec: 30
And a new proposal: 32
MJPEG Baseline DCT - YCbCr DIB Map 33
Sector Alignment and Padding 34
8.0 OpenDML File Format Certification Procedures **36**
Concurrently Develop a Test Suite to measure compatibility 36

BNSDOCID: <XP.....2179318A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1709
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Table of Contents

SNSDOCID: <XP.....2179318A_1_>

SNEDOCID: <XP 2179818A 1>

1.0 Introduction

The OpenDML File Format Subcommittee is defining an *AVI-compatible* file format that addresses the particular needs of professional video. As such, this document relates specific proposals to these needs.

Further work by OpenDML will elaborate on extensions to the Codec and Vidcap specifications of Video for Windows to meet the needs of professional video.

Scope

This document describes the proposed format of the OpenDML-compliant AVI extensions, specifically for the Motion-JPEG DIB AVI file. This format is an extension to the AVI M-JPEG DIB as defined in the *JPEG DIB FORMAT* technical note from Microsoft. The format is based on the ISO 10918 defined JPEG format.

It is assumed that the reader is familiar with JPEG as defined in the ISO 10918 document. For additional information on JPEG, see the *ISO 10918 Information technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines.*

For additional information about the JPEG DIB format, see Microsoft Multimedia Technical Note *JPEG DIB FORMAT*.

For additional information about RIFF files see the Microsoft Windows Software Development Kit *Multimedia Programmer's Guide* and *Multimedia Programmer's Reference.*

For additional information on installable compressors and decompressors, see the *Video Compression/Decompression Drivers* technical note from Microsoft.

Portions of the above Microsoft documents have been reprinted with permission of Microsoft Corp.

OpenDML Sub-Committee on File Format: High Level Goals

- Interoperability and AVI Compatibility
- Large File Support
- Frame and Field Indexing
- Interleaving Audio with Video
- Source and Header Information Storage
- OpenDML File Format Certification Code
- Microsoft Certification and Communication

Version History

Version 1.0	November 13, 1995	First Release
Version 1.1	December 18, 1995	Includes small changes from Microsoft

BNSDOCID: <XP_____2179318A__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1712
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

1.0 Overview of Profession Video Requirements

Goal

Provide interoperability between different hardware and software vendors' Motion JPEG codecs. Support Video for Windows AVI file format for MJPG video, while improving the performance of these systems.

Issues

AVI: Support Current AVI Format

The goal is to support interoperability by standardizing on the AVI file format for Motion-JPEG video. This implies a minimum compatibility to support the standard AVI file access interface, as well as the vidcap and codecs that are part of Video for Windows. This means support of the MJPG DIB specification, supporting the RIFF file format as well as the standard AVI stream headers, allowing for optional audio interleaving, and supporting one storage of one frame of video per ##dc data chunk in the LIST 'movi' chunk of the AVI file.

60 Fields Per Second vs. 30 Frames Per Second; or 24 Frames Per Second For Film

Professional Video Applications require the ability to sequence individual fields of the video data for certain playback rates and video effects. For example, in order to do a slow motion effect, it is not sufficient to just repeat the frames; the fields must be individually repeated. For example, as sequence of fields in a file numbered 123456... should be played at half speed as 11223344... rather than 12123434.... As such, the individual fields should be accessible. AVI requires that frames are stored per data chunk. Storing individual fields in each chunk would break AVI applications.

The extended AVI format will allow for access to individual fields. In addition, the format will allow for storage of 24 frame based files, such that file readers can properly convert between 24 frame-based data storage and 25 or 30 frame PAL and NTSC playback.

Improved Tombstone and Header Information

Professional video applications need extended information such as starting timecode and reel ID to be contained inside the AVI file. More advanced information such as the location of timecode discontinuities, etc. are also desirable.

BNSDOCID: <XP.....2179318A...>

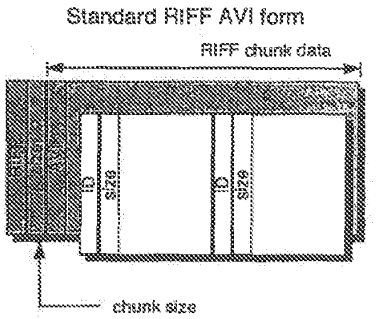
Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1713
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

2.0 Increased AVI File Size

Introduction: RIFF Chunk Format

The AVI file begins as a standard RIFF chunk (note that this is a non-standard representation that shows the 4-byte size):

```
RIFF (size) 'AVI ' ...
```



There is only one RIFF chunk per file. RIFF sub-chunks may be either LIST chunks or regular sub-chunks. The LIST chunk obeys the same structure and may have regular or LIST sub-chunks; all other RIFF sub-chunks have just an ID identifier and a size. These regular sub-chunks may not have sub-chunks.

Increased File Size Limits (>> 1 GB)

The current RIFF file format implies a maximum chunk size of 4 GB because the size is stored as a 32-bit value. However, limitations to the RIFF parser code and MCIAVI limits the file to only 1 GB. At data rates of 10 MB/s, a 1 GB file will last less than 2 minutes.

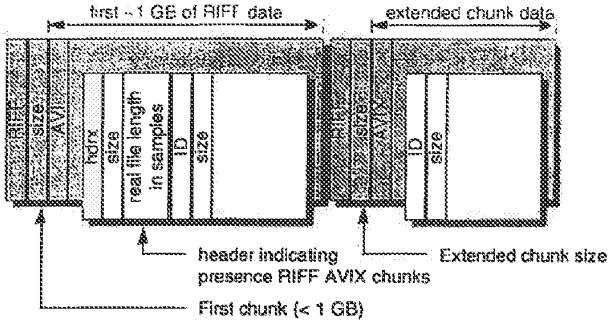
Extension for File Size > 1 GB

An AVI file can be extended beyond 1 GB by placing more than one RIFF chunk in the same file (which is legal). Each RIFF chunk can have a maximum size of 1 GB. Standard AVI applications will see the first RIFF chunk (RIFF 'AVI") as a standard AVI file, and this chunk should be complete. Information that the file is extended over 1 GB should be placed in the Extended AVI header (see below). Subsequent RIFF chunks will be identified by the AVIX (for AVI extended) chunk id. These RIFF 'AVIX' chunks do not need to contain further data. It is expected that the AVIX chunks only contain LIST 'movi' data. All other AVI information should be stored in the first (RIFF 'AVI') chunk so that it is accessible by all applications.

SNSDOCID: <XP.....21793188A...1.>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1714
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

RIFF AVI form for files > 1 GB



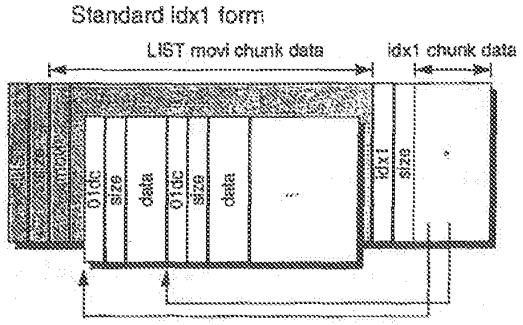
OpenDML AVI File Format Extensions

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1715
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.0 Frame and Field Indexing

Introduction: 'idx1' Index Chunk

The AVI file format specifies an optional 'idx1' chunk, which contains a list of the offsets and chunk types of every chunk inside the LIST 'movi' chunk. This list is used to make AVI playback and seeks more efficient because the location of each frame of video can be found without scanning through each sub-chunk of the LIST 'movi' data.



* data of idx1 chunk is an index to each movi sub-chunk, includes offset and length of each chunk

Issues

Allow Field Indexing

In order to support field accurate indexing while still retaining AVI compatibility, the video data must be stored one frame per ##dc 'movi' chunk, while an index must be present that gives the locations of both fields (if they exist) within the frame. Applications could then use field-based codecs to perform field-based effects (e.g., slow motion, etc.).

Allow Incremental Growth of Files

The 'idx1' chunk follows the LIST 'movi' chunk in the file. As such, it must be moved in order to insert new data into the file. An interleaved frame index would be more efficient for the purposes of file growth.

Minimize Disk Seeks

In playback, there are several causes of inefficiency. One is that the index is at the end of the file, requiring a large seek when opening the file. The other is that the index itself contains all the chunks inside the 'movi' data, not just those for one AVI stream. As such, the index must be preprocessed so that the location of each frame in the index can be found.

BNSDOCID: <XP_____21758318A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1716
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Proposed Index List

Overview

Instead of using an index chunk like 'idx1', a new index structure is proposed. The index has the following main characteristics:

- The can either be a single index chunk, or a two tiered index, with a super index pointing to interleaved index segments in the 'movi' data.
- The index can include locations of the fields within a frame.
- The index has no restrictions to 4 GB file size.
- There is an index per data stream (if necessary).

The advantages of the index are as follows:

- The index can contain the locations of the individual fields in each chunk as well as the chunk information as a whole. The 'idx1' chunk only contains indices of the frames.
- The index segments are interleaved within the 'movi' data, meaning that the file can be grown. The 'idx1' chunk is present at the end of the file. It must be moved for the file to grow.
- The index segment is smaller than an 'idx1' chunk, so the amount of data needed to be read in order to access a particular frame is less. Also, if the index is interleaved before its data (i.e., less efficient in write), then playback will be more efficient than the 'idx1' chunk.
- The index lists are present per stream number (##). As such, the list can be accesses like an array, unlike the 'idx1' chunk, which contains entries for all 'movi' chunks from all streams.

The index is implemented as an index of indexes (optional) and an index of chunks. Both are based on a base index form.

Base Index Form 'indx'

Thus the actual implementation is based on a base index form 'indx':

```
struct _aviindex_chunk {
   FOURCC fcc;
   DWORD  cb;
   WORD   wLongsPerEntry;    // size of each entry in aIndex array
   BYTE   bIndexSubType;     // future use.  must be 0
   BYTE   bIndexType;        // one of AVI_INDEX_* codes
   DWORD  nEntriesInUse;     // index of first unused member in aIndex array
   DWORD  dwChunkId;         // fcc of what is indexed
   DWORD  dwReserved[3];     // meaning differs for each index
                             // type/subtype.   0 if unused
   struct _aviindex_entry {
      DWORD adw[wLongsPerEntry];
   } aIndex[ ];
};
```


BNSDOCID: <XP_____2179318A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1717
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The actual size of the entries of the aIndex array is entered into wLongsPerEntry. Between the cb field and wLongsPerEntry the actual size of the array is known. The field nEntriesInUse allows a chunk to be allocated longer than the actual number of used elements in the array.

The types defined for the bIndexType and bIndexSubtype field are the following:

```
// bIndexType codes
//
#define AVI_INDEX_OF_INDEXES 0x00    // when each entry in aIndex
                                     // array points to an index chunk
#define AVI_INDEX_OF_CHUNKS  0x01    // when each entry in aIndex
                                     // array   points to a chunk in the
file
#define AVI_INDEX_IS_DATA       0x80       // when each entry is aIndex is
                                     // really the data
// bIndexSubtype codes for INDEX_OF_CHUNKS
//
#define AVI_INDEX_2FIELD     0x01    // when fields within frames
                                     // are also indexed
```

AVI Standard Index Chunk

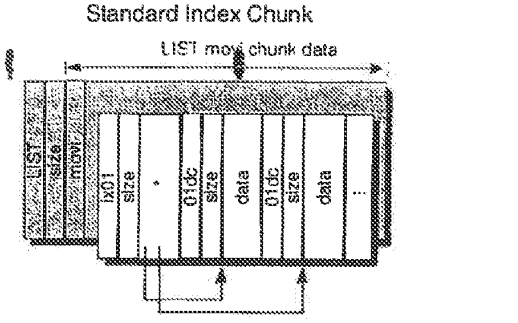
The AVI Standard Index chunk contains information that indexes AVI frames.

```
typedef struct _avistdindex_chunk {
   FOURCC   fcc;              // 'ix##'
   DWORD    cb;
   WORD     wLongsPerEntry;  // must be sizeof(aIndex[0])/sizeof(DWORD)
   BYTE     bIndexSubType;   // must be 0
   BYTE     bIndexType;      // must be AVI_INDEX_OF_CHUNKS
   DWORD    nEntriesInUse;   //
   DWORD    dwChunkId;       // '##dc' or '##db' or '##wb' etc..
   QUADWORD qwBaseOffset;    // all dwOffsets in aIndex array are
                             // relative to this
   DWORD    dwReserved3;     // must be 0
   struct _avistdindex_entry {
      DWORD    dwOffset;     // qwBaseOffset + this is absolute file offset
      DWORD    dwSize;       // bit 31 is set if this is NOT a keyframe
   } aIndex[ ];
} AVISTDINDEX, * PAVISTDINDEX;
```

A single standard index chunk can only index data within a 4 GB region. The dwOffset field points to the start of the data itself, and not to the start of the RIFF chunk for that field.

BNSDOCID: <XP_____2178318A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1718
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)



The index chunk above is shown in the 'movi' data, but may be found in the stream header (see below).

AVI Field Index Chunk

The AVI Field Index Chunk is the same as the Standard Index Chunk except that it contains the locations of each field in the frame.

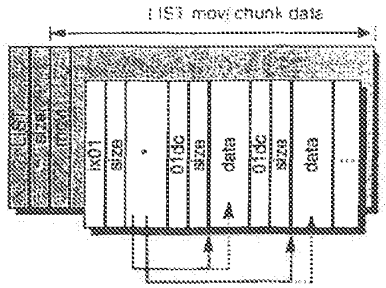
```
typedef struct _avifieldindex_chunk {
   FOURCC   fcc;               // 'ix##'
   DWORD    cb;
   WORD     wLongsPerEntry;    // must be 3 (size of each entry in
                               // aIndex array)
   BYTE     bIndexSubType;     // AVI_INDEX_2FIELD
   BYTE     bIndexType;        // AVI_INDEX_OF_CHUNKS
   DWORD    nEntriesInUse;     //
   DWORD    dwChunkId;         // '##dc' or '##db'
   QUADWORD qwBaseOffset;      // offsets in aIndex array are relative to this
   DWORD    dwReserved3;       // must be 0
   struct _avifieldindex_entry {
      DWORD    dwOffset;
      DWORD    dwSize;        // size of all fields
                                 // (bit 31 set for NON-keyframes)
      DWORD    dwOffsetField2; // offset to second field
   } aIndex[ ];
} AVIFIELDINDEX, * PAVIFIELDINDEX;
```

The field `wLongsPerEntry` is set to 3 because of the addition of the `dwOffsetField2` field in the array. This is indicated by setting the sub type to `AVI_INDEX_2FIELD`.

8NSOOOD: <XP_____2179318A_.I_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1719
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Field Index Chunk



* data of Field Index Chunk is an index to each movi sub-chunk data, includes offset and length of chunk, as well as offset to the second field.

The index chunk above is shown in the movi data, but may be found in the stream header (see below).

AVI Super Index Chunk

The Super Index Chunk is an index of indexes and is always found in the 'indx' chunk of an AVI file. It is defined as follows:

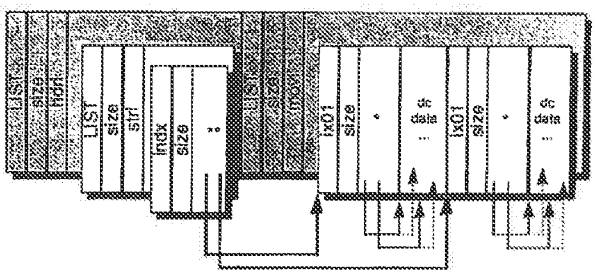
```
typedef struct _avisuperindex_chunk {
   FOURCC  fcc;              // 'ix##'
   DWORD   cb;               // size of this structure
   WORD    wLongsPerEntry;   // must be 4 (size of each entry in aIndex array)
   BYTE    bIndexSubType;    // must be 0 or AVI_INDEX_2FIELD
   BYTE    bIndexType;       // must be AVI_INDEX_OF_INDEXES
   DWORD   nEntriesInUse;    // number of entries in aIndex array that
                             // are used
   DWORD   dwChunkId;        // '##dc' or '##db' or '##wb', etc
   DWORD   dwReserved[3];    // must be 0
   struct _avisuperindex_entry {
      QUADWORD qwOffset;     // absolute file offset, offset 0 is
                             // unused entry??
      DWORD    dwSize;       // size of index chunk at this offset
      DWORD    dwDuration;   // time span in stream ticks
   } aIndex[ ];
} AVISUPERINDEX, * PAVISUPERINDEX;
```

The `bIndexSubType` is set to the type of index that the Super Index points to. If the index chunks are Standard Index Chunks, then the value should be 0. If the index chunks are AVI Field Index Chunks, then the value should be `AVI_INDEX_2FIELD`. This implies that a stream cannot mix Field and Standard Index Chunks.

BNSDOCID: <XP_____21783318A__>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1720
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Super Index Chunk



* data of ##ix chunk is an index to each movi
sub-chunk data, includes offset and length of chunk,
and possibly the offset to the second field.
** data of Super Index indx chunk is an index to each ##ix
movi sub-chunk.

If we use the ##pc chunks, then they must be indexed by their own Index List: Palette changes should not appear in the same Index List as ##dc chunks, since they eliminate the ability to access an element (frame) directly without having to scan the list.

Index Locations in RIFF File

Unlike the 'idx1' chunk, a single index is stored per *stream* in the AVI file. An 'indx' chunk follows the 'strf' chunk in the LIST 'strl' chunk of an AVI header. This 'indx' chunk may either be an index of indexes (super index), or may be an index to the chunks directly. In the case of video, this means that the chunk is either a `AVISUPERINDEX` or an `AVIFIELDINDEX/AVISTDINDEX`.

If the 'indx' chunk is a standard or field index chunk (i.e., not an index of indexes) then the stream has only one index chunk and there is none in the 'movi' data.

If the 'indx' chunk is a Super Index, then the corresponding index chunks are marked with 'ix##' in the 'movi' data. The ## is the stream number, the same as for the ##dc or ##wb. The index chunks can be either standard or field index chunks.

A file can be easily grown if it has a standard index in the 'indx' chunk position. The chunk can be moved to a new 'ix##' chunk, and a new super index can be inserted into the stream header ('indx' position). New 'ix##' chunks can be added to grow the file.

Note the reversal of the '##ix' to 'ix##'. This is for AVI backward compatibility. Note that INDEX_IS_DATA streams remain as '##ix'.

Here is a sample RIFF file in the format shown by the RIFFWALK utility:

```
RIFF (7F038718) 'AVI '
     LIST (0001084C) 'hdrl'
          avih (00000038)
```


BNSDOCID: <XP_____2179018A__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1721
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
        LIST (000044E0) 'strl'
            strh (00000038)
            strf (00000428)
            indx (00003FF8)    <- video superindex
            vprp (00000064)    <- video property header
        LIST (00004058) 'strl'
            strh (00000038)
            strf (00000012)
            indx (00003FF8)    <- audio superindex
        LIST (0000404C) 'strl'
            strh (00000038)    <- timecode stream...
            strf (00000000)
            indx (00003FF8)    <- timecode superindex
        LIST (00000060) 'odml'
            dmlh (00000054)    <- size still temp.
    LIST (7F000000) 'movi'
        01wb (00002dF0)
        00db (00030004)
        01wb (00002dF0)
        00db (00030004)
        ... time passes
        02ix (00003FF8)      <- data chunk for timecode stream
        ... time passes
        ix00 (00003FF8)      <- partial video index
        ix01 (00003FF8)      <- partial audio index
        .. etc ..
    idx1 (00030000)          <- original index (optional)
RIFF (7F038718) 'AVIX'
```

Proposed Index List

BNSDOCID: <XP.....2178918A_1_>

SNEDOCID: <XP_____21763193A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1723
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.0 Interleaving Audio with Video

Issues

Interleaving Rate
Files for Editing
Files for Playback

Single or Multiple Files for Video and Audio

Multiple Tracks of Audio Storage

Mono and Stereo Audio

Support of multiple streams of data exists in AVI. A proposal to recommend that audio be stored as two mono streams should be examined. Not all applications would be able to play this back in stereo, however.

Audio / Video Time Stamping (Sync Point List)

It is necessary to add time stamping information to account (and correct) for drift between audio and video when using hardware that is not able to maintain a perfect synchronization between audio and video (via a hardware connection) and when there exists a drift in the crystal frequency on audio hardware.

This can be accomplished by adding a sync point list into the file. A sync point list identifies two samples in different streams that are to be played at the same time. Thus, drift can be corrected by identifying sync points and if necessary, correcting for the drift during playback.

Notes on Interleaving Audio and Video

With the exception of AVI files playing off CD-ROM drives, it is not necessary to interleave the audio and video into REC chunks. In fact, with higher video and audio data rates, it is probably more efficient to store the audio in large consecutive blocks. As such, the audio can be interleaved in large blocks (not necessarily done on a frame basis).

Audio data chunks may or may not be in integral frames, or multiple of frames. The other option would be a constant number of bytes. In addition, audio may be interleaved before or after the corresponding video. This is just a tradeoff between record and playback performance. Typically, saving audio chunks after video in record will give better record performance, which is important because hard drives tend to be less performant during write operations. However, during playback, multiple streams and cuts may be played, reducing the drive performance.

BNSDOCID: <XP_____2179818A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1724
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

BNSDOCID: <XP_____2179318A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1725
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

5.0 Source and Header Information Storage

Goal

The goal is to implement a standard way to store certain header and tombstone information in an AVI file so that other applications can interpret and use this data. In many cases, the header data is not uniformly set between different AVI files.

Required Information

Main AVI Header (avih)

Total Frames

The `dwTotalFrames` field indicates the size (number of frames) within the first RIFF 'AVI' chunk.

AVI Stream Header (strh)

Quality Information

AVI defines the quality of a clip by a DWORD value from 0 to 10000. This value seems arbitrary. For MJPG files, a standard quality should be defined. The quality can be defined by either of these three modes:

- constant average data rate
- constant Q-table
- lossless

Lossless is not part of the Baseline process. As such it requires the definition of a new Process. Under baseline, lossless can be achieved by setting the Q-tables to unity[1].

Under constant data rate, the desired data rate for the file should be specified (in kb/s). This, plus the number of fields per frame, should be able to determine the Q-tables required to compress the image.

---

[1] While a lossless (or pseudo lossless) process can be defined by setting the Q-tables to unity, the ISO specification of lossless JPEG compression is a different process. Lossless encoding as defined by ISO does not use DCT transformed data. For a sample to be encoded, a predictor is formed from the reconstructed values of up to three neighborhood samples. This gives a prediction of the of the sample to be encoded. The prediction is subtracted from the actual value of the sample and the difference is losslessly entropy coded using either Huffman or arithmetic coding. As far as forming a type of pseudo lossless compression from a baseline JPEG process with unity Q tables, there is the potential of some slight loss of data during the DCT process due to arithmetic precision. Practically speaking, there is no real image quality loss, but the word *lossless* implies a guarantee of reconstructing the source image on a bit by bit basis.

BNSDOCID: <XP.....2179318A...1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1726
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Under constant Q-table, there are two possibilities. Either the absolute Q-tables are given (8x8 matrices) per component (in our case, 2), or a factor which is used to generate any Q-table of a standard one. The constant Q-table file would of course have the same tables repeated per field, as per the MJPG DIB spec. The advantage of a Q-factor is the reduction of header information. However, tweaked Q-tables would no longer be possible.

Scale / Rate

The scale and rate parameters will define the correct ratio for different video standards. Known ratio values are (rate/scale):

NTSC	30000/1001
PAL	25/1

rcFrame

The on-disk RECT coordinates are 16-bit values. The RECT structure in NT is a 32-bit (LONG) for each coordinate, but the on-disk ones remain 16-bit.

AVI Stream Format (strf)

Height

The MJPG DIB spec lists that heights less than 288 are for single frames (one chunk) and greater than 288 are for interleaved fields. However, there is no real connection between frame height and interleaving. The **biHeight** parameter refers to the raw height of the frame (interleaved or not). OpenDML codecs that operate in field by field modes will be passed this value divided by 2 to get the field height.

Extended AVI Header (dmlh)

```
typedef struct {
   DWORD dwTotalFrames;
} ODMLExtendedAVIHeader;
```

Total Frames

The `dwTotalFrames` field indicates the **real** size of the AVI file. Since the same field in the Main AVI Header 'avih' indicates the size within the first RIFF 'AVI' chunk.

Video Properties Header (vprp)

The video properties header identifies video signal properties associated with a digital video stream in an AVI file. This header attempts to address two main video properties:

- The type of video signal (PAL, NTSC, etc., as well as the resolution of the video signal).
- The framing of the compression within a video signal.

The parameters can be used to uniquely describe a video signal.

```
typedef struct {
   DWORD          VideoFormatToken;
   DWORD          VideoStandard;
   DWORD          dwVerticalRefreshRate;
   DWORD          dwHTotalInT;
   DWORD          dwVTotalInLines;
```


BNSDOCID: <XP.....21793184A...(..>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1727
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
    DWORD               dwFrameAspectRatio;
    DWORD               dwFrameWidthInPixels;
    DWORD               dwFrameHeightInLines;
    DWORD               nbFieldPerFrame;
    VIDEO_FIELD_DESC    FieldInfo[nbFieldPerFrame];
} VideoPropHeader;
```

Video Format Token

```
enum {FORMAT_UNKNOWN, FORMAT_PAL_SQUARE, FORMAT_PAL_CCIR_601,
    FORMAT_NTSC_SQUARE, FORMAT_NTSC_CCIR_601,...} VIDEO_FORMAT;
```

The format token indicates that a known standard is defined for the following data fields. Those fields must be filled, but their value can be expected to be the defined standard. If the format is defined as FORMAT_UNKNOWN then the fields may contain special values. Known tokens are defined in the table below:

Token	*Standard*	*Refresh Rate*	*H-Total*	*V-Total*	*Frame Aspect Ratio*	*Frame Width*	*Frame Height*	*Pixel Aspect Ratio (derived)*
NTSC CCIR 601	NTSC	60	858	525	0x0004 0003 (4:3)	720	485	2160:1940
NTSC SQUARE	NTSC	60	780	525	0x0004 0003 (4:3)	640	480	1:1
PAL CCIR 601	PAL	50	864	625	0x0004 0003 (4:3)	720	576	2160:2304
PAL SQUARE	PAL	50	944	625	0x0004 0003 (4:3)	768	576	1:1

Video Standard

```
enum {STANDARD_UNKNOWN, STANDARD_PAL, STANDARD_NTSC, STANDARD_SECAM}
    VIDEO_STANDARD;
```

Defines standards such as NTSC, PAL etc. Implicitly defines vertical refresh rate.

Vertical Refresh Rate

Used when an unknown standard is specified. Normally, 60 for NTSC, and 50 for PAL.

BNSDOCID: <XP_____2178318A__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1728
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

H-Total in T

Defines the horizontal total, in T (one luminance sample: pixel)

V-Total in Lines

Defines the vertical total, in lines.

Active Frame Aspect Ratio

The aspect ratio is stored as a DWORD value with a word each storing the x:y ratio. For example, 1 to 1 is 0x00010001. Standard values for television is 4:3 or 16:9. This value can be used with the frame width and height to calculate the pixel aspect ratio.

Active Frame Width in Pixels

Defines the active frame width in pixels. The bitmap might digitize a region that is smaller or bigger than the active video width.

Active Frame Height in Lines

Defines the frame height in lines. The bitmap might digitize a region that is smaller or bigger than the active video height.

Number of Fields Per Frame

One or two, depending on whether the video is interlaced or progressive.

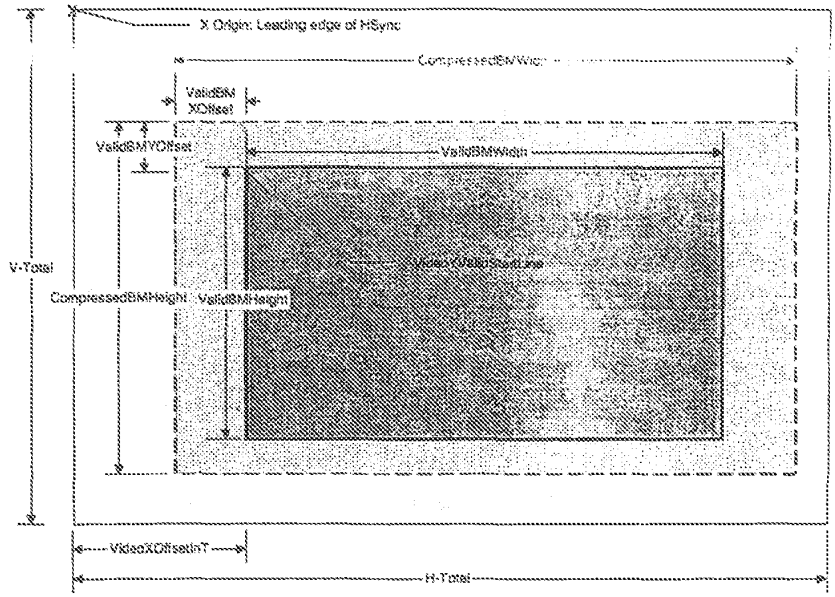
Field Framing Information

The field framing information defines where the compressed image is with respect to the video signal. The data is present for each field (and may be different).

```
Typedef struct {
   DWORD   CompressedBMHeight;
   DWORD   CompressedBMWidth;
   DWORD   ValidBMHeight;
   DWORD   ValidBMWidth;
   DWORD   ValidBMXOffset;
   DWORD   ValidBMYOffset;
   DWORD   VideoXOffsetInT;
   DWORD   VideoYValidStartLine;
} VIDEO_FIELD_DESC;
```


8NSDOCID: <XP_2179318A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1729
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)



Compressed Bitmap Height and Width

The compressed bitmap height and width represent the size of the compressed image. For JPEG, these values are multiples of 8.

Valid Bitmap Height and Width, X and Y Offset

The valid bitmap height, width and x and y offsets represent the size of the valid data within the compressed bitmap. Because padding may be required when compressing, it is not guaranteed that all the data within the compressed image is valid. Note that compressing blanking is still valid. In the case where all the compressed bitmap comes from the video signal, then the valid height and width are equal to the compressed height and width, and the offsets are 0.

Valid X-Offset In T

The `VideoXOffsetInT` is used to locate the x position of the start of the valid bitmap with reference to the video signal. The value is a measurement in units of T, which is one luminance-sampling clock, from the leading edge of the horizontal sync pulse (CCIR 624-3).

Valid Y Start Line

The `VideoYValidStartLine` field is used to locate the line that the valid bitmap starts on. This value will be different for each field. (CCIR 624-3).

A typical value for the Framing Information for NTSC CCIR 601 would be:

Vide	*Co*	*Co*	*Vali*	*Vali*	*Vali*	*Va*	*Vide*	*Video*

Required Information

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1730
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

o Typ e	*mpr esse d BM Hei ght*	*mpr esse d BM Wid th*	*d BM Hei ght*	*d BM Wid th*	*d BM X Off set*	*lid B M Y Off set*	*o X Offs et In T*	*Y Valid Start Line*
Fiel d 1	248	720	248	720	0	0	122	20
Fiel d 1	248	720	248	720	0	0	122	283

Source and Timecode Information

The following information is used to describe timecode and source information inside an AVI file.

Base Timecode Structure

All SMPTE timecode information is stored in the following format:

```
typedef union _timecode {
   struct {
      WORD      wFrameRate;  // 0 is 30 drop.  do we need other drop
                             // frame rates?
      WORD      wFramePract; // fractional frame. full scale is always 0x10000
      LONG      lFrame;
   };
   __int64  qwAll;
} TIMECODE, *PTIMECODE;
```

This format allows for a frame, and a frame fractional value to be specified, with the frame rate stored in the low word of the value. As the values are stored, two timecodes can be compared and subtracted. For drop frame code, no `lFrame` values are skipped, so a drop frame timecode of 1:00:00;00 would be a `lFrame` value of 107892.

Timecode Discontinuity Table (Stream) 'tcdl'

The following structure defines a timecode discontinuity table. This would be stored as a stream itself. The table would either be a table alone in the 'indx' chunk of that stream, or if the table is large, an index to the table could be stored with the table itself stored as a ##ix in the 'movi' data.

```
#define FILM_SEQUENCE_NONE       0
#define FILM_SEQUENCE_AABBBCCCDD 1
#define FILM_SEQUENCE_AAABBCCCDD 2
// structure of a timecode discontinuity table. note that this fits
// within the definition of the structure of an index chunk
//
typedef struct _timecode_dl_chunk {
```


BNSDOCID: <XP_____2179318A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1731
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
   FOURCC   fcc;              // '##ix'
   DWORD    cb;               // sizeof this structure less 8
   WORD     wLongsPerEntry;   // must be 4 or more
   BYTE     bIndexSubType;    // must be 0
   BYTE     bIndexType;       // AVI_INDEX_IS_DATA
   DWORD    nEntriesInUse;    // index of first unused entry in aIndex array
   DWORD    dwChunkId;        // 'tcdl' (timecode discontinuity list)
   DWORD    dwReserved[3];    // future, must be 0
   struct _timecode_dl_entry {
      QUADWORD  qwTick;       // time in terms of this streams's tick rate
      TIMECODE  timecode;     // timecode
      DWORD     dwUser;       // timecode user data
      struct {
         DWORD  ColorFrame : 4;       // Which frame in color sequence
         DWORD  ColorSequence : 4;    // Duration in frames of complete sequence
         DWORD  FilmFrame : 5;        // Offset into pull-down sequence
         DWORD  FilmSequenceType : 3; // One of FILM_SEQUENCE_XXX defines
         DWORD  Reserved : 16;        // Future use - set to 0
      } Flags;
      TCHAR     szReelId[32]; // source id
   } aIndex[ ];
};
```

The Flags structure can optionally specify color framing, and film pull-down information. If all fields are 0 then no such optional information is present. Notice that there is no explicit flag for drop frame NTSC since that information in already stored in the TIMECODE structure.

If `ColorSequence` is non-zero value, then `ColorFrame` contains a frame number from 0 to `ColorSequence-1`, which specifies the color framing of the frame specified in the timecode structure.

If `FilmSequenceType` is not `FILM_SEQUENCE_NONE`, then `FilmFrame` contains the offset into the pull-down sequence corresponding to the frame specified in the timecode structure. For example, if `FilmFrame` is 1 and `FilmSequenceType` is `FILM_SEQUENCE_AABBBCCDDD` then the referenced frame is a BB frame. Again, if `FilmFrame` is 1 and `FilmSequenceType` is `FILM_SEQUENCE_AAABBCCCDD` then the referenced frame is an AB frame.

Timecode Stream 'time'

This structure defines a timecode value for every tick of the timecode stream

```
// structure of a timecode stream data chunk. for storing every
// event in a timecode stream rather than just the discontinuities
//
typedef struct _timecode_stream_chunk {
   FOURCC   fcc;              // '##ix'
   DWORD    cb;               // sizeof this structure less 8
   WORD     wLongsPerEntry;   // must be 3
```


BNEDOCID: <XP_____21780318A__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1732
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
   BYTE     bIndexSubType;   // must be 0
   BYTE     bIndexType;      // AVI_INDEX_IS_DATA
   DWORD    nEntriesInUse;   // index of first unused entry in aIndex   array
   DWORD    dwChunkId;       // 'time' (timecode stream)
   DWORD    dwReserved[3];   // future, must be 0
   struct _timecode_stream_entry {
      TIMECODE  timecode;    // timecode
      DWORD     dwUser;      // timecode user data
      struct {
         DWORD  ColorFrame : 4;       // Which frame in color sequence
         DWORD  ColorSequence : 4;    // Duration in frames of complete sequence
         DWORD  FilmFrame : 5;        // Offset into sequence
         DWORD  FilmSequenceType : 3; // One of FILM_SEQUENCE_XXX defines
         DWORD  Reserved : 16;        // Future use - set to 0
      } Flags;
   } aIndex[ ];
};
```

Film Transfer Log Information

```
#define FILM_LOG_EVERY1          1
#define FILM_LOG_FLEX            2
// structure of a film transfer log.  Note that this fits
// within the definition of the structure of an index chunk
//
typedef struct _film_transfer_chunk {
   FOURCC   fcc;               // '##ix'
   DWORD    cb;                // sizeof this structure less 8
   WORD     wLongsPerEntry;    // must be 4 or more
   BYTE     bIndexSubType;     // must be 0
   BYTE     bIndexType;        // AVI_INDEX_IS_DATA
   DWORD    nEntriesInUse;     // index of first unused entry in aIndex array
   DWORD    dwChunkId;         // 'film' (film transfer log)
   DWORD    dwReserved[3];     // future, must be 0
   struct _film_transfer_header {
      DWORD     dwLogType;     // Which type of FILM_LOG_XXX file format
      DWORD     dwHeaderSize; // Size of this _film_transfer_header structure
      DWORD     dwReserved[4];// Future use - set to 0
      // DWORD  dwUserDefined[];  // Optional user data
   } FilmHeader;
   // BYTE  bTransferLog[];     // The actual transfer log begins here
};
```

The size of the `FilmHeader` is given by the `dwHeaderSize`. This allows for user defined additions to the header, which appear at the end of the `FilmHeader`. The ASCII film

BNSDOCID: <XP 2179316A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1733
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

transfer log begins immediately after the `FilmHeader` structure. The size of the ASCII data can be determined from the `dwHeaderSize` of the header and the `cb` size of the entire structure. The ASCII data should be padded with NULLs to the nearest DWORD structure size.

The contents of the `bTransferLog` array is byte for byte the contents of the ASCII transfer log file of the type given in the `dwLogType` field. For now, two types are defined:

FILM_LOG_EVERTZ: the detailed .FTL file format can be obtained from

Evertz Microsystems, Ltd.

3465 Mainway

Burlington, Ontario L7M 1A9, Canada

905-335-3700

FILM_LOG_FLEX: the detailed FLEX file format can be obtained from

Time Logic, Inc.

11992 Challenger Ct.

Moorpark, CA 93021

805-529-1155

Microsoft-Defined Tombstone Data

The following tombstone data has already been defined by Microsoft:

Additional AVI information following the 'strl' chunk in the AVI header LIST 'hdrl' chunk. (from *AVI RIFF form, Video for Windows SDK*):

Additional Header Data

After the 'strl' chunk, some AVI files might have additional header data. Additional information can include timecode chunks that apply to the whole file. The following timecode four-character codes are useful in AVI files:[2]

Four-character code	Description
ISMP	Indicates the chunk contains the SMPTE timecode of the digitization starting point. The time is expressed as a zero-terminated text string of the form HH:MM:SS.FF. If performing MCI capture in AVICAP, this chunk is automatically set from the MCI start time.
IDIT	Indicates the chunk specifies the time and date digitizing commenced. This time is contained in an ASCII string of

[2] Note that these chunks are Ixxx chunks, and as such, should be found in an INFO chunk (see comment below)?

BNSDOCID: <XP.....2178318A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1734
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

exactly 26 characters and has the format "Wed Jan 02 02:03:55 1990\n\0". The ctime and asctime functions can be used to create strings in this format.

The 'ISMP' timecode chunk usually corresponds to the starting timecode copied from the first sample of a captured sequence. Use this chunk to index the captured sample to the original sample. For example, if you need to recapture a video sequence, you can use the timecode to reposition the original sequence and restart capture from that point.

The 'IDIT' timecode chunk usually corresponds to the time the sample was captured. This timecode provides a reference to the age of a sample and creates a history if a series of samples is captured.

The weakness with the above section is that the ISMP information does not include any reel ID, so the effective timecode is meaningless.

Additional information, global to RIFF files (from *Multimedia File Formats, Windows 3.1 SDK*):

The INFO List Chunk

The "INFO" list is a registered global form type that can store information that helps identify the contents of the chunk. This information is useful but does not affect the way a program interprets the file; examples are copyright information and comments. An "INFO" list is a "LIST" chunk with list type "INFO." The following example shows a sample "INFO" list chunk:

```
LIST('INFO'  INAM("Two Trees"Z)
             ICMT("A picture for the opening screen"Z) )
```

An "INFO" list should contain only the following chunks. New chunks may be defined, but an application should ignore any chunk it doesn't understand. The chunks listed below may only appear in an "INFO" list. Each chunk contains a ZSTR, or null-terminated text string.

Chunk ID	Description
IARL	*Archival Location*. Indicates where the subject of the file is archived.
IART	*Artist*. Lists the artist of the original subject of the file; for example, "Michaelangelo."
ICMS	*Commissioned*. Lists the name of the person or organization that commissioned the subject of the file; for example, "Pope Julian II."
ICMT	*Comments*. Provides general comments about the file or the subject of the file. If the comment is several sentences long, end each sentence with

BNSDOCID: <XP.....2179318A...3.>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1735
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

a period. Do not include new-line characters.

ICOP	*Copyright*. Records the copyright information for the file; for example, "Copyright Encyclopedia International 1991." If there are multiple copyrights, separate them by a semicolon followed by a space.
ICRD	*Creation date*. Specifies the date the subject of the file was created. List dates in year-month-day format, padding one-digit months and days with a zero on the left; for example, "1553-05-03" for May 3, 1553.
ICRP	*Cropped*. Describes whether an image has been cropped and, if so, how it was cropped; for example, "lower-right corner."
IDIM	*Dimensions*. Specifies the size of the original subject of the file; for example, "8.5 in h, 11 in w."
IDPI	*Dots Per Inch*. Stores dots per inch setting of the digitizer used to produce the file, such as "300."
IENG	*Engineer*. Stores the name of the engineer who worked on the file. If there are multiple engineers, separate the names by a semicolon and a blank; for example, "Smith, John; Adams, Joe."
IGNR	*Genre*. Describes the original work, such as "landscape," "portrait," "still life," etc.
IKEY	*Keywords*. Provides a list of keywords that refer to the file or subject of the file. Separate multiple keywords with a semicolon and a blank; for example, "Seattle; aerial view; scenery."
ILGT	*Lightness*. Describes the changes in lightness settings on the digitizer required to produce the file. Note that the format of this information depends on hardware used.
IMED	*Medium*. Describes the original subject of the file, such as "computer image," "drawing," "lithograph," and so on.
INAM	*Name*. Stores the title of the subject of the file, such as "Seattle From Above."
IPLT	*Palette Setting*. Specifies the number of colors requested when digitizing an image, such as "256."
IPRD	*Product*. Specifies the name of the title the file was originally intended for, such as "Encyclopedia of Pacific Northwest Geography."
ISBJ	*Subject*. Describes the contents of the file, such as "Aerial view of Seattle."
ISFT	*Software*. Identifies the name of the software package used to create the file, such as "Microsoft WaveEdit."

BINSDOCID: <XP.....2179318A...1.>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1736
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ISHP	*Sharpness*. Identifies the changes in sharpness for the digitizer required to produce the file (the format depends on the hardware used).
ISRC	*Source*. Identifies the name of the person or organization who supplied the original subject of the file; for example, "Trey Research."
ISRF	*Source Form*. Identifies the original form of the material that was digitized, such as "slide," "paper," "map," and so on. This is not necessarily the same as IMED.
ITCH	*Technician*. Identifies the technician who digitized the subject file; for example, "Smith, John."

BNSDOCID: <XP_____2179016A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1737
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Microsoft-Defined Tombstone Data

6.0 Other File Issues

Issues

Continuation Over Multiple Files

With restrictions to the maximum file format and limitations on the maximum size of disk partitions, it may be desirable to implement a mechanism that specifies a linkage across multiple files. It will certainly be desired that the data of one stream of video can be captured across multiple files. For the moment, this will be external to the AVI file.

Minimized Time to Open Files

tbd

Simultaneous Reading and Writing / Multiple Access

tbd

BNSDOCID: <XP.....2179316A...I.>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1739
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

issues

IBNSDOCID: <XP_____3179818A_1_>

7.0 MJPG Motion JPEG DIB Extensions

Issues

ISO vs. Non ISO Formats

For interoperability between various hardware and software codecs, it is not sufficient that the file format be standardized; the contents of the video data chunks must be standardized. In order to comply with industry standards, the AVI MJPG specification and OMF 2.0 file format specifications, the standard ISO Motion JPEG specification is proposed, with certain additions to support efficient identification of the interleaved fields in the frame.

Non-ISO formats could be used, provided that they provide the same outward appearance as the ISO data chunk. This would require different codecs to decompress and compress the data.

8-bit vs. 10-bit YUV

This document currently addresses 8-bit 4:2:2 YUV as per the Microsoft specification.

Video vs. Bitmap

Issues between the fact that a bitmap is a pure rectangle, while video starts with half lines; Motion JPEG compresses on multiple of 8 lines, so what is the frame of video that is compressed vs. the real frame.

See the section of the Video Properties Header in Section 5.0.

Proposed Data Chunk Format

The data chunk format is defined as per the Microsoft *JPEG DIB FORMAT* Technical Note, Type 2: Motion JPEG DIB.

From the DIB spec:

Type 2: Motion JPEG

Motion JPEG DIBs shall accommodate interchange formats which satisfy the "General sequential and progressive syntax" (ISO 10918 Part 1, Annex B, Para. B.2). A set of images of this type with compatible parameters can be placed in an AVI file to describe a motion sequence. Frame headers for these DIBs shall be limited to those specified in Para B.2.2 of the cited Annex B. These types are SOF0, SOF1, SOF2, SOF3, SOF9, SOF10 and SOF11. Of the types accommodated, this specification provides implementation only for the Baseline Sequential DCT (SOF0).

This DIB type contains incomplete JPEG data (Abbreviated Format per ISO 10918) and is not intended for stand-alone single image frame disk files. It may be used within RIFF files and other contexts where it is appropriate to:

a. Decode an image without supplying the associated JPEG Huffman tables. This presumes the codec has been properly pre-initialized prior to image decode.

BNSDOCID: <XP.....2179318A...3_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1741
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

b. Request encoder output of compressed image data absent embedded Huffman Tables.

All motion JPEG data will use YCbCr encoding.

In an AVI sequence all JPEG frames will be key frames as this ensures that within the AVI and Video for Windows architecture all frames will be directly and independently addressable.

For optimal size and speed during playback of an AVI file the Huffman data used by motion JPEG will be fixed and defined by this document. This will make the individual frames of every motion sequence smaller and more efficient to play back. Also as all sequences of motion images use the same Huffman data and color space it is much more likely that motion data can be directly exchanged without re-compression. A definition of the Huffman data will be provided in MMREG.H (which is listed at the end of this document) as a byte string which can be concatenated onto the start of a motion JPEG image to form a valid still JPEG image-

```
MJPGDHTSeg = ( 'X'FF', DHT, length, JPEG Huffman table
parameters )
```

Q-table data is present and may vary in every frame of a motion sequence to permit control over the bandwidth of sequences that contain bursts of frames of varying levels of complexity. The restart interval used during the compression process may also vary for every frame.

Only the interleaved form of YCrCb images is supported for motion JPEG data. This implies that only one SOS segment will be present in any particular motion JPEG image.

The following applies (again from MS DIB spec):

As in the JPEG DIB format the JPEG stream syntax is used for the image data with the following constraints. The following JPEG marker codes SOI, DRI, DQT, SOF0, SOS and EOI are expected (mandatory) in the image data chunk, and the constrained values shown in the example below are mandatory for the image data within the AVI stream.

Any parameters in the SOF0 (frame) and SOS (start of scan) headers that are duplicated in the `BITMAPINFOHEADER` for JPEG must be the same. This would include Sample Precision, subsampling, number of components (as implied by `JPEGColorSpaceID`), etc. The number of lines and samples per lines in the SOF0 segment and the width and height defined in the format chunk must match the main AVI header width and height values. All of these values are expected to remain the same for every image data chunk in the AVI sequence.

Within the image data chunk two JPEG segments beginning with the SOI marker and ending with the EOI marker are allowed to accommodate field-interleaved streams. There is an APP0 marker immediately following the SOI marker that contains information about the video image. Specifically, this allows the identification of the two fields that comprise an interleaved frame. This APP0 marker is expected to have the first 4 bytes following the length bytes set to the characters 'A', 'V', 'I', '1'. The next byte indicates which field the JPEG data was compressed from and has an expected value of 0x01 for the FIRST JPEG data segment and 0x02 for the SECOND segment, indicating the FIRST and SECOND fields respectively. If the stream is not field interleaved then this value will be 0x00 and

BNSDOC00: <XP.....2178015A...1.>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1742
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

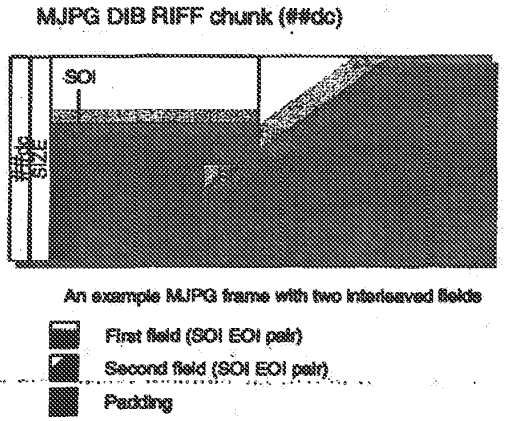
there will only be one JPEG segment. The remaining seven bytes are expected to be set to 0 and will be ignored by the codec.

If a codec cannot handle the interleaved fields, the codec will use only the FIRST field and will replicate the lines as necessary to provide an image that conforms to the image size defined in the main AVI header. Conversely if a capture system only accesses a single field of each source frame only a single field image may be present in a JPEG stream. This implies that the single field data should be used as the source of both fields by a decompressor that wishes to process full-interlaced data.

And a new proposal:

The APP0 marker will have an added use beside the field polarity and will also specify the size of the current field (entirely, from SOI to EOI), as well as any padding required. This is to allow applications and codecs to avoid being forced to scan for the second SOI-EOI pair. The size information in the APP0 marker does not include any information about a second field in the ##dc chunk. This way, fields can be copied as integral units without having to have the contained data modified for the presence and size of neighbouring fields.

The byte following the polarity will be reserved for future expansion. The remaining 8 bytes of this marker will be used to store two sizes values. The first value specifies the length in bytes of the field (entirely, from SOI to EOI) including any padding after the EOI (which should be small) and the second value specifies the size of the same field excluding any padding after the EOI. Both size values will be stored in the usual ISO JPEG fashion where bytes are stored in decreasing order with the most significant byte being first. The length of this marker was previously 14 and is now set to 16.



BNSDOCID: <XP_____2179318A_(_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1743
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

MJPEG Baseline DCT - YCbCr DIB Map

A Baseline DCT - YCbCr will now therefore have the following look :

```
SOI   (xFFD8)
APP0  (xFFE0),
      Length (16),
      "AVI1", Polarity, 0, FieldSize,
FieldSizeLessPadding.
DRI   (xFFDD),
      Length (4),
      Restart interval.
DQT   (xFFDB),
      Length (132),
      Precision (0), Table ID (0),
      DQT data [64] for table 0,
      Precision (0), Table ID (1),
      DQT data [64] for table 1.
SOF0  (xFFC0),
      Length (17),
      Sample Precision (8),
      Number of lines (biHeight : multiple of 8),
      Sample per line (biWidth : multiple of 16),
      Number of components (3)
      ID of 1st component (1),
      Sampling ratio of 1st component (H,V),
      Q-table ID of 1st component (0),
      ID of 2nd component (2),
      Sampling ratio of 2nd component (H,V),
      Q table ID of 2nd component (1),
      ID of 3rd component (3),
      Sampling ratio of 3rd component (H,V),
      Q table ID of 3rd component (1).
SOS   (xFFDA),
      Length (12),
      Number of components (3)
      ID of 1st component (1),
      DC and AC Huffman ID for 1st component (0,0),
      ID of 2nd component (2),
```


BNSDOCID: <XP_____2179318A_(_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1744
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
        DC and AC Huffman ID for 2nd component (1,1),
        ID of 3rd component (3),
        DC and AC Huffman ID for 3rd component (1,1),
        Start of spectral (0),
        End of spectral (63),
        Successive approximation bit position, high and
low (0,0).
*** IMAGE DATA ***
There could be some Restart markers.
Values are from xFFD0 to xFFD7.
EOI (xFFD9).
```

Be sure to have a single DQT as the previous ones will be overridden.

The order of the markers may affect some codecs.

IDs for the 3 component as recommended by the Microsoft DIB specification are 1,2 and 3.

Sector Alignment and Padding

The ##dc chunks may be sector aligned if necessary. As such, a padding chunk would be inserted following the ##dc to preserve the RIFF format. Padding between fields should be discouraged; however the ISO specification allows padding if the data contains 0xFF for each padded byte.

BNSDOCID: <XP_____2179318A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1745
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Proposed Data Chunk Format

8.0 OpenDML File Format Certification Procedures

Concurrently Develop a Test Suite to measure compatibility

Test older Codecs with new AVI files.

Test old AVI files with new Codecs.

Circulate all files through industry (i.e., everybody tests everybody else's files).

File checker routine to verify AVI format. Program may grow to include test decompressions using software Codec.

Test using MCIAVI with standard codecs.

Go to MS porting labs for specific video file format / Codec test week.

SNSDOCID: <XP.....2179318A_1_>

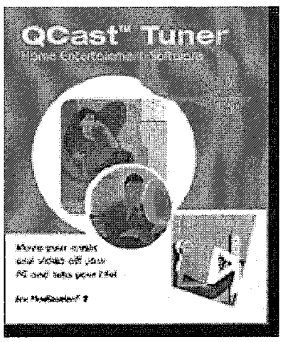
Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1747
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Internet Explorer cannot display the webpage

Search for Movies
FIND EVERY DIVX MOVIE ON THE NET!
GO

Advanced Search

GAMECUBE? XBox?. PS2?
Game Developers, use DivX® video in your games.
LEARN MORE



Questions?

Have a question about one of these products? Visit our Help Desk.

QCast™ Tuner for PS2

Category: Player Apps
By: BroadQ, LLC
Price: 49.95 USD

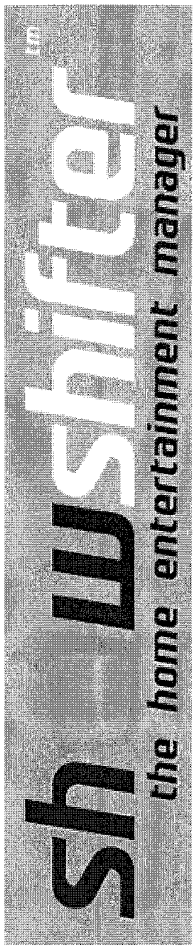
QCast Tuner home entertainment software runs on PlayStation 2 consoles (US Version) equipped with a Sony Network adapter and provides access to media files stored on a computer for viewing on a home entertainment system or television.

Features:

- State of the Art Digital Sound and Vision Technologies - MP3 music, MPEG-1, MPEG-2, MPEG-4 AND DivX (5.xx and earlier) video at resolutions of 720x320 and below.
- Dynamic, Continuous Updates - QCast Tuner software automatically updates itself, so consumers always have the latest version of the software.
- Simple to Use Push Button Navigation on your Television - Select and play PC-based digital media from the couch using the Sony PlayStation 2 game pad or DVD remote.
- Network Capability - QCast Tuner is optimized for media delivery over wired and wireless 802.11a or 802.11b (WiFi) environments using the Sony Ethernet network adapter.
- Powerful Playlist Organization and Management Functions - Review, manage, organize and enjoy integrated lists of music and video.

Product Home Page Buy Now

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1748
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)



Home | DivX | Movies | Software | Hardware | Skins | Forums | Support
About DivX | Contact Us | Advertising | Jobs | Corporate Information

Copyright © 2000-2003 All rights reserved.
DivX® is a registered trademark of DivXNetworks, Inc.

Privacy Policy | Terms of Use



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1749
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Smooth Streaming Client

The IIS Smooth Streaming Client allows you to build rich client experiences that take full advantage of IIS Smooth Streaming capabilities. IIS Smooth Streaming Client 1.1 allows you to build Silverlight® players for both live and on-demand Smooth Streaming. Examples of available client-side features include simple controls, such as Play, Pause, and Stop, as well as more complex functionality, such as full DVR, Instant Replay, Live Ad Insertion, Multiple Camera Angles, and Fast Forward / Rewind.

For developers and designers, the IIS Smooth Streaming Client APIs provide an easy interface to create rich Smooth Streaming experiences using the Silverlight platform. These .NET Framework-based APIs provide support for simple operations such as Play, Pause, and Stop and also more complex operations such as scheduling advertisements, selecting and tracking bit rates for Smooth Streaming playback, and many more.

The Smooth Streaming Client is a core component of the broader open source Silverlight Media Framework. For most scenarios, the Silverlight Media Framework is the simplest, most robust way to build a new Silverlight player that supports the full set of Smooth Streaming functionality. The Smooth Streaming Client may be ideal for some scenarios, however, such as adding rich Smooth Streaming features to a Silverlight media player based on a different framework, or where unique scenarios cannot otherwise be addressed.

To get started quickly, a sample Smooth Streaming Player is provided on the Silverlight Media Framework Downloads page. Quick-start documentation on building your own player for multiple endpoints, including Windows Phone 7, is on the Silverlight Media Framework Documentation page. For additional Windows Phone 7 development information, please see the Windows Phone Developer portal.

The Smooth Streaming Client supports these high-level features:

- Basic playback controls:
    - APIs such as Play, Pause, Stop
    - Events for playback and diagnostics
    - Properties to track position, etc.
- Advanced playback support:
    - DVR support for Live Smooth Streaming
    - Support for tracking live events while in DVR
    - Trickplay – slow motion, fast-forward, rewind *
    - Multiple Audio Language support

    - Text stream support for captions and subtitles *
- Support for offline playback scenarios
- H.264/AAC support
- Monetization
    - Ad playback integration – scheduling capabilities, tracking ad progress *
    - Live ad insertion with IIS Live Smooth Streaming *
    - Rich analytics with IIS Advanced Logging
    - Analytics APIs
    - Support for specifying markers and ad ingestion points outside the main manifest *
- Content Protection – PlayReady integration for VC-1 & H.264 content
- Composite manifest support for scenarios such as Rough Cut Editing (RCE) *
- Selecting tracks for playback (for example, restrict the available bit rates, support multiple camera angles in a single stream, etc.)
- Support for progressive download ads and content *

Note: Items denoted above by an asterisk (*) work on Silverlight 4 players only, and are not available on Silverlight for Windows Phone 7

Download

Smooth Streaming Client 1.5

Install | Using the Microsoft® Web Platform Installer

Or, Download

More IIS Media Information

- Integrated Media Platform Overview
- More IIS Media Extensions

More IIS Extensions
Administration Pack
Advanced Logging
Application Request Routing
Application Warm-Up
Bit Rate Throttling
Database Manager
Dynamic IP Restrictions
FastCGI for IIS
FTP Publishing Service
IIS Manager for Remote Administration
Live Smooth Streaming
Search Engine Optimization Toolkit
Smooth Streaming
Smooth Streaming Client
Smooth Streaming Format SDK
Transform Manager
URL Rewrite
UrlScan 3.1
Web Deploy 2.0
Web Farm Framework
Web Platform Installer
Web Playlists
WebDAV Extension
Windows Cache for PHP
Windows Media Services 2008
Windows PowerShell Snap-In

Learn More

Articles

- Smooth Streaming Client Readme
- IIS Smooth Streaming Client API Documentation

Forum

- Media Forum

Blog Posts

- IIS Smooth Streaming Player Development Kit 1.0 – Beta 1 Released
- Building your first Smooth Streaming Player using SSME

Download Smooth Streaming Client 1.5

WebPI / Download

MATROŠKA

Main Menu

- Home
- What is Matroska?
- Downloads
- News
- Guides
- FAQ
- Technical / Info
  - Diagram
  - **Specifications**
  - Specification Notes
  - Order Guidelines
  - Codec Specs
  - Chapters
  - Subtitles
  - Tags
  - Menu
  - Streaming
  - Overhead
  - EBML RFC
  - SVN Repository
  - Team
  - License
  - Contributions
- Project Page
- Blog
- Forum

Contact

- Legal
- Logos / Trademarks
- Contact
- Sponsors

Specifications

Status of this document

This document is not the real format specification. It's a simple draft to work. (For a simplified diagram of the layout of a Matroska file, see the Diagram page.) But since it's quite complete it is used as a reference for the development of libmatroska. An alternate version of the specification can be found here (PDF doc maintained by Alexander Noé).

EBML principle

EBML is short for Extensible Binary Meta Language. EBML specifies a binary and octet (byte) aligned format inspired by the principle of XML. EBML itself is a generalized description of the technique of binary markup. Like XML, it is completely agnostic to any data that it might contain. Therein, the Matroska project is a specific implementation using the rules of EBML: It seeks to define a subset of the EBML language in the context of audio and video data (though it obviously isn't limited to this purpose). The format is made of 2 parts: the semantic and the syntax. The semantic specifies a number of IDs and their basic type and is not included in the data file/stream. There is a specific project dealing with EBML in more details and more recent updates.

Just like XML, the specific "tags" (IDs in EBML parlance) used in an EBML implementation are arbitrary. However, the semantic of EBML outlines general data types and ID's.

The known basic types are:

- Signed Integer - Big-endian, any size from 1 to 8 octets
- Unsigned Integer - Big-endian, any size from 1 to 8 octets
- Float - Big-endian, defined for 4 and 8 octets (32, 64 bits)
- String - Printable ASCII (0x20 to 0x7E), zero-padded when needed
- UTF-8 - Unicode string, zero padded when needed (RFC 2279)
- Date - signed 8 octets integer in nanoseconds with 0 indicating the precise beginning of the millennium (at

BNSDOCID: <XP_____55246006A__I_> 1007060413 03/http://www.matroska.org/technical/specs/index.html

2001-01-01T00:00:00,000000000 UTC)

Master-Element - contains other EBML sub-elements of the next lower level

Binary - not interpreted by the parser

As well as defining standard data types, EBML uses a system of Elements to make up an EBML "document." Elements incorporate an Element ID, a descriptor for the size of the element, and the binary data itself. Futher, Elements can be nested, or contain, Elements of a lower "level."

Element IDs (also called EBML IDs) are outlined as follows, beginning with the ID itself, followed by the Data Size, and then the non-interpreted Binary itself:

Element ID coded with an UTF-8 like system :

```
bits, big-endian
1xxx xxxx                                  - Class A IDs (2^7 -1 possible values) (base 0x8X)
01xx xxxx  xxxx xxxx                       - Class B IDs (2^14-1 possible values) (base 0x4X 0xXX)
001x xxxx  xxxx xxxx  xxxx xxxx            - Class C IDs (2^21-1 possible values) (base 0x2X 0xXX 0xXX)
0001 xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx - Class D IDs (2^28-1 possible values) (base 0x1X 0xXX 0xXX 0xXX)
```

Some Notes:

The leading bits of the EBML IDs are used to identify the length of the ID. The number of leading 0's + 1 is the length of the ID in octets. We will refer to the leading bits as the Length Descriptor.

Any ID where all x's are composed entirely of 1's is a Reserved ID, thus the -1 in the definitions above.

The Reserved IDs (all x set to 1) are the only IDs that may change the Length Descriptor.

Data size, in octets, is also coded with an UTF-8 like system :

```
bits, big-endian
1xxx xxxx                                                                       - value 0 to  2^7-2
01xx xxxx  xxxx xxxx                                                            - value 0 to 2^14-2
001x xxxx  xxxx xxxx  xxxx xxxx                                                 - value 0 to 2^21-2
0001 xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx                                      - value 0 to 2^28-2
0000 1xxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx                           - value 0 to 2^35-2
0000 01xx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx                - value 0 to 2^42-2
0000 001x  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx     - value 0 to 2^49-2
0000 0001  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx - value 0 to 2^56-2
```

Since modern computers do not easily deal with data coded in sizes greater than 64 bits, any larger Element Sizes are left undefined at the moment. Currently, the Element Size coding allows for an Element to grow to 72000 To, i.e. 7x10^16 octets or 72000 terabytes, which will be sufficient for the time being.

There is only one reserved word for Element Size encoding, which is an Element Size encoded to all 1's. Such a coding indicates that the size of the Element is unknown, which is a special case that we believe will be useful for live streaming purposes. However, avoid using this reserved word unnecessarily, because it makes parsing slower and more difficult to implement.

Data

Integers are stored in their standard big-endian form (no UTF-like encoding), only the size may differ from their usual form (24 or 40 bits for example).

The Signed Integer is just the big-endian representation trimmed from some 0x00 and 0xFF where they are not meaningful (sign). For example -2 can be coded as 0xFFFFFFFFFFFFFFFE or 0xFFFE or 0xFE and 5 can be coded 0x000000000005 or 0x0005 or 0x05.

Elements semantic

A more detailed description of the column headers can be found in the Specification Notes.

Element Name - The full name of the described element.

L - Level - The level within an EBML tree that the element may occur at. + is for a recursive level (can be its own child).

EBML ID - The Element ID displayed as octets.

Ma - Mandatory - This element is mandatory in the file.

Mu - Multiple - The element may appear multiple times within its parent element.

Rng - Range - Valid range of values to store in the element.

Default - The default value of the element.

T - Element Type - The form of data the element contains. m: Master, u: unsigned int, i: signed integer, s: string, 8: UTF-8 string, b: binary, f: float, d: date

1 - The element is contained in Matroska version 1.

2 - The element is contained in Matroska version 2.

W - All elements available for use in WebM.

Description - A short description of the element's purpose.

Element Name	L	EBML ID	Ma	Mu	Rng	Default	T	1	2	W	Description
EBML Header											
EBML	0	[1A][45][DF][A3]	*	*	-	-	m				Set the EBML characteristics of the data to follow. Each EBML document has to start with this.
EBMLVersion	1	[42][86]	*	-	-	1	u				The version of EBML parser used to create the file.
EBMLReadVersion	1	[42][F7]	*	-	-	1	u				The minimum EBML version a parser has to support to read this file.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1755
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

EBMLMaxIDLength	1	[42][F2]	*	-	-	4	u				The maximum length of the IDs you'll find in this file (4 or less in Matroska).
EBMLMaxSizeLength	1	[42][F3]	*	-	-	8	u				The maximum length of the sizes you'll find in this file (8 or less in Matroska). This does not override the element size indicated at the beginning of an element. Elements that have an indicated size which is larger than what is allowed by EBMLMaxSizeLength shall be considered invalid.
DocType	1	[42][82]	*	-	-	matroska	s				A string that describes the type of document that follows this EBML header. 'matroska' in our case or 'webm' for webm files.
DocTypeVersion	1	[42][87]	*	-	-	1	u				The version of DocType interpreter used to create the file.
DocTypeReadVersion	1	[42][85]	*	-	-	1	u				The minimum DocType version an interpreter has to support to read this file.
Global elements (used everywhere in the format)											
CRC-32	1+	[BF]	-	-	-	-	b	*	*		The CRC is computed on all the data of the Master element it's in, regardless of its position. It's recommended to put the CRC value at the beggining of the Master element for easier reading. All level 1 elements should include a CRC-32.
Void	1+	[EC]	-	-	-	-	b	*	*	*	Used to void damaged data, to avoid unexpected behaviors when using damaged data. The content is discarded. Also used to reserve space in a sub-element for later use.
signature											
SignatureSlot	1+	[1B][53][86][67]	-	*	-	-	m				Contain signature of some (coming) elements in the stream.
SignatureAlgo	2+	[7E][8A]	-	-	-	-	u				Signature algorithm used (1=RSA, 2=elliptic).
SignatureHash	2+	[7E][9A]	-	-	-	-	u				Hash algorithm used (1=SHA1-160, 2=MD5).
SignaturePublicKey	2+	[7E][A5]	-	-	-	-	b				The public key to use with the algorithm (in the case of a PKI-based signature).
Signature	2+	[7E][B5]	-	-	-	-	b				The signature of the data (until a new.
SignatureElements	2+	[7E][5B]	-	-	-	-	m				Contains elements that will be used to compute the signature.
SignatureElementList	3+	[7E][7B]	-	*	-	-	m				A list consists of a number of consecutive elements that represent one case where data is used in signature. Ex: *Cluster|Block|BlockAdditional* means that the BlockAdditional of all Blocks in all Clusters is used for encryption.
SignedElement	4+	[65][32]	-	*	-	-	b				An element ID whose data will be used to compute the signature.
end of signature											
Element Name	**L**	**EBML ID**	**Ma**	**Mu**	**Rng**	**Default**	**T**	**1**	**2**	**W**	**Description**
Segment											
Segment	0	[18][53][80][67]	*	*	-	-	m	*	*	*	This element contains all other top-level (level 1) elements. Typically a Matroska file is composed of 1 segment.
Meta Seek Information											
SeekHead	1	[11][4D][9B][74]	-	*	-	-	m	*	*	*	Contains the position of other level 1 elements.
Seek	2	[4D][BB]	*	*	-	-	m	*	*	*	Contains a single seek entry to an EBML element.
SeekID	3	[53][AB]	*	-	-	-	b	*	*	*	The binary ID corresponding to the element name.
SeekPosition	3	[53][AC]	*	-	-	-	u	*	*	*	The position of the element in the segment in octets (0 = first level 1 element).

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1756
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Segment Information												
Info	1	[15][49][A9][66]	*	*	-	-	m	*	*	*	Contains miscellaneous general information and statistics on the file.
SegmentUID	2	[73][A4]	-	-	not 0	-	b	*	*		A randomly generated unique ID to identify the current segment between many others (128 bits).
SegmentFilename	2	[73][84]	-	-	-	-	8	*	*		A filename corresponding to this segment.
PrevUID	2	[3C][B9][23]	-	-	-	-	b	*	*		A unique ID to identify the previous chained segment (128 bits).
PrevFilename	2	[3C][83][AB]	-	-	-	-	8	*	*		An escaped filename corresponding to the previous segment.
NextUID	2	[3E][B9][23]	-	-	-	-	b	*	*		A unique ID to identify the next chained segment (128 bits).
NextFilename	2	[3E][83][BB]	-	-	-	-	8	*	*		An escaped filename corresponding to the next segment.
SegmentFamily	2	[44][44]	-	*	-	-	b	*	*		A randomly generated unique ID that all segments related to each other must use (128 bits).
ChapterTranslate	2	[69][24]	-	*	-	-	m	*	*		A tuple of corresponding ID used by chapter codecs to represent this segment.
ChapterTranslateEditionUID	3	[69][FC]	-	*	-	-	u	*	*		Specify an edition UID on which this correspondance applies. When not specified, it means for all editions found in the segment.
ChapterTranslateCodec	3	[69][BF]	*	-	-	-	u	*	*		The chapter codec using this ID (0: Matroska Script, 1: DVD-menu).
ChapterTranslateID	3	[69][A5]	*	-	-	-	b	*	*		The binary value used to represent this segment in the chapter codec data. The format depends on the ChapProcessCodecID used.
TimecodeScale	2	[2A][D7][B1]	*	-	-	1.000.000	u	*	*	*	Timecode scale in nanoseconds (1.000.000 means all timecodes in the segment are expressed in milliseconds).
Duration	2	[44][89]	-	-	> 0	-	f	*	*	*	Duration of the segment (based on TimecodeScale).
DateUTC	2	[44][61]	-	-	-	-	d	*	*	*	Date of the origin of timecode (value 0), i.e. production date.
Title	2	[7B][A9]	-	-	-	-	8	*	*		General name of the segment.
MuxingApp	2	[4D][80]	*	-	-	-	8	*	*	*	Muxing application or library ("libmatroska-0.4.3").
WritingApp	2	[57][41]	*	-	-	-	8	*	*	*	Writing application ("mkvmerge-0.3.3").
Cluster											
Cluster	1	[1F][43][B6][75]	-	*	-	-	m	*	*	*	The lower level element containing the (monolithic) Block structure.
Timecode	2	[E7]	*	-	-	-	u	*	*	*	Absolute timecode of the cluster (based on TimecodeScale).
SilentTracks	2	[58][54]	-	-	-	-	m	*	*		The list of tracks that are not used in that part of the stream. It is useful when using overlay tracks on seeking. Then you should decide what track to use.
SilentTrackNumber	3	[58][D7]	-	*	-	-	u	*	*		One of the track number that are not used from now on in the stream. It could change later if not specified as silent in a further Cluster.
Position	2	[A7]	-	-	-	-	u	*	*		Position of the Cluster in the segment (0 in live broadcast streams). It might help to resynchronise offset on damaged streams.
PrevSize	2	[AB]	-	-	-	-	u	*	*	*	Size of the previous Cluster, in octets. Can be useful for backward playing.
SimpleBlock	2	[A3]	-	*	-	-	b		*	*	Similar to Block but without all the extra information, mostly used to reduced overhead when no extra feature is needed. (see SimpleBlock Structure)

IBNSDOCID: <XP 552460064 1 > 10070604120 2 [http://www.matroska.org/technical/specs/index.html]

BlockGroup	2	[A0]	-	*	-	-	m	*	*	*	Basic container of information containing a single Block or BlockVirtual, and information specific to that Block/VirtualBlock.
Block	3	[A1]	*	-	-	-	b	*	*	*	Block containing the actual data to be rendered and a timecode relative to the Cluster Timecode. (see Block Structure)
BlockVirtual	3	[A2]	-	*	-	-	b				A Block with no data. It must be stored in the stream at the place the real Block should be in display order. (see Block Virtual)
BlockAdditions	3	[75][A1]	-	-	-	-	m	*	*		Contain additional blocks to complete the main one. An EBML parser that has no knowledge of the Block structure could still see and use/skip these data.
BlockMore	4	[A6]	*	*	-	-	m	*	*		Contain the BlockAdditional and some parameters.
BlockAddID	5	[EE]	*	-	not 0	1	u	*	*		An ID to identify the BlockAdditional level.
BlockAdditional	5	[A5]	*	-	-	-	b	*	*		Interpreted by the codec as it wishes (using the BlockAddID).
BlockDuration	3	[9B]	-	-	-	TrackDuration	u	*	*	*	The duration of the Block (based on TimecodeScale). This element is mandatory when DefaultDuration is set for the track. When not written and with no DefaultDuration, the value is assumed to be the difference between the timecode of this Block and the timecode of the next Block in "display" order (not coding order). This element can be useful at the end of a Track (as there is not other Block available), or when there is a break in a track like for subtitle tracks.
ReferencePriority	3	[FA]	*	-	-	0	u	*	*		This frame is referenced and has the specified cache priority. In cache only a frame of the same or higher priority can replace this frame. A value of 0 means the frame is not referenced.
ReferenceBlock	3	[FB]	-	*	-	-	i	*	*	*	Timecode of another frame used as a reference (ie: B or P frame). The timecode is relative to the block it's attached to.
ReferenceVirtual	3	[FD]	-	-	-	-	i				Relative position of the data that should be in position of the virtual block.
CodecState	3	[A4]	-	-	-	-	b		*		The new codec state to use. Data interpretation is private to the codec. This information should always be referenced by a seek entry.
Slices	3	[8E]	-	*	-	-	m	*	*		Contains slices description.
TimeSlice	4	[E8]	-	*	-	-	m	*	*		Contains extra time information about the data contained in the Block. While there are a few files in the wild with this element, it is no longer in use and has been deprecated. Being able to interpret this element is not required for playback.
LaceNumber	5	[CC]	-	-	-	0	u	*	*	*	The reverse number of the frame in the lace (0 is the last frame, 1 is the next to last, etc). While there are a few files in the wild with this element, it is no longer in use and has been deprecated. Being able to interpret this element is not required for playback.
FrameNumber	5	[CD]	-	-	-	0	u				The number of the frame to generate from this lace with this delay (allow you to generate many frames from the same Block/Frame).
BlockAdditionID	5	[CB]	-	-	-	0	u				The ID of the BlockAdditional element (0 is the main Block).
Delay	5	[CE]	-	-	-	0	u				The (scaled) delay to apply to the element.
Duration	5	[CF]	-	-	-	0	u				The (scaled) duration to apply to the element.

[BNSDOCID: <XP_____55246006A__I_> 100706041303/http://www.matroska.org/technical/specs/index.html

EncryptedBlock	2	[AF]	-	*	-	-	b				Similar to SimpleBlock but the data inside the Block are Transformed (encrypt and/or signed). (see EncryptedBlock Structure)
Track											
Tracks	1	[16][54][AE][6B]	-	*	-	-	m	*	*	*	A top-level block of information with many tracks described.
TrackEntry	2	[AE]	*	*	-	-	m	*	*	*	Describes a track with all elements.
TrackNumber	3	[D7]	*	-	not 0	-	u	*	*	*	The track number as used in the Block Header (using more than 127 tracks is not encouraged, though the design allows an unlimited number).
TrackUID	3	[73][C5]	*	-	not 0	-	u	*	*	*	A unique ID to identify the Track. This should be kept the same when making a direct stream copy of the Track to another file.
TrackType	3	[83]	*	-	1-254	-	u	*	*	*	A set of track types coded on 8 bits (1: video, 2: audio, 3: complex, 0x10: logo, 0x11: subtitle, 0x12: buttons, 0x20: control).
FlagEnabled	3	[B9]	*	-	0-1	1	u		*	*	Set if the track is used. (1 bit)
FlagDefault	3	[88]	*	-	0-1	1	u	*	*	*	Set if that track (audio, video or subs) SHOULD be used if no language found matches the user preference. (1 bit)
FlagForced	3	[55][AA]	*	*	0-1	0	u	*	*	*	Set if that track MUST be used during playback. There can be many forced track for a kind (audio, video or subs), the player should select the one which language matches the user preference or the default + forced track. Overlay MAY happen between a forced and non-forced track of the same kind. (1 bit)
FlagLacing	3	[9C]	*	-	0-1	1	u	*	*	*	Set if the track may contain blocks using lacing. (1 bit)
MinCache	3	[6D][E7]	*	-	-	0	u	*	*		The minimum number of frames a player should be able to cache during playback. If set to 0, the reference pseudo-cache system is not used.
MaxCache	3	[6D][F8]	-	-	-	-	u	*	*		The maximum cache size required to store referenced frames in and the current frame. 0 means no cache is needed.
DefaultDuration	3	[23][E3][83]	-	-	not 0	-	u	*	*	*	Number of nanoseconds (i.e. not scaled) per frame.
TrackTimecodeScale	3	[23][31][4F]	*	-	> 0	1.0	f	*	*		The scale to apply on this track to work at normal speed in relation with other tracks (mostly used to adjust video speed when the audio length differs).
TrackOffset	3	[53][7F]	-	-	-	0	i				A value to add to the Block's Timecode. This can be used to adjust the playback offset of a track.
MaxBlockAdditionID	3	[55][EE]	*	-	-	0	u	*	*		The maximum value of BlockAddID. A value 0 means there is no BlockAdditions for this track.
Name	3	[53][6E]	-	-	-	-	8	*	*	*	A human-readable track name.
Language	3	[22][B5][9C]	-	-	-	eng	s	*	*	*	Specifies the language of the track in the Matroska languages form.
CodecID	3	[86]	*	-	-	-	s	*	*	*	An ID corresponding to the codec, see the codec page for more info.
CodecPrivate	3	[63][A2]	-	-	-	-	b	*	*	*	Private data only known to the codec.
CodecName	3	[25][86][38]	-	-	-	-	8	*	*	*	A human-readable string specifying the codec.
AttachmentLink	3	[74][46]	-	-	not 0	-	u	*	*		The UID of an attachment that is used by this codec.
CodecSettings	3	[3A][96][97]	-	-	-	-	8				A string describing the encoding setting used.
CodecInfoURL	3	[3B][40][40]	-	*	-	-	s				A URL to find information about the codec used.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1759
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

CodecDownloadURL	3	[26][B2][40]	-	*	-	-	s				A URL to download about the codec used.
CodecDecodeAll	3	[AA]	*	-	0-1	1	u			*	The codec can decode potentially damaged data (1 bit).
TrackOverlay	3	[6F][AB]	-	*	-	-	u	*	*		Specify that this track is an overlay track for the Track specified (in the u-integer). That means when this track has a gap (see SilentTracks) the overlay track should be used instead. The order of multiple TrackOverlay matters, the first one is the one that should be used. If not found it should be the second, etc.
TrackTranslate	3	[66][24]	-	*	-	-	m	*	*		The track identification for the given Chapter Codec.
TrackTranslateEditionUID	4	[66][FC]	-	*	-	-	u	*	*		Specify an edition UID on which this translation applies. When not specified, it means for all editions found in the segment.
TrackTranslateCodec	4	[66][BF]	*	-	-	-	u	*	*		The chapter codec using this ID (0: Matroska Script, 1: DVD-menu).
TrackTranslateTrackID	4	[66][A5]	*	-	-	-	b	*	*		The binary value used to represent this track in the chapter codec data. The format depends on the ChapProcessCodecID used.
video											
Video	3	[E0]	-	-	-	-	m	*	*	*	Video settings.
FlagInterlaced	4	[9A]	*	-	0-1	0	u		*	*	Set if the video is interlaced. (1 bit)
StereoMode	4	[53][B8]	-	-	0-3	0	u		*		Stereo-3D video mode on 2 bits (0: mono, 1: right eye, 2: left eye, 3: both eyes).
PixelWidth	4	[B0]	*	-	not 0	-	u	*	*	*	Width of the encoded video frames in pixels.
PixelHeight	4	[BA]	*	-	not 0	-	u	*	*	*	Height of the encoded video frames in pixels.
PixelCropBottom	4	[54][AA]	-	-	-	0	u	*	*	*	The number of video pixels to remove at the bottom of the image (for HDTV content).
PixelCropTop	4	[54][BB]	-	-	-	0	u	*	*	*	The number of video pixels to remove at the top of the image.
PixelCropLeft	4	[54][CC]	-	-	-	0	u	*	*	*	The number of video pixels to remove on the left of the image.
PixelCropRight	4	[54][DD]	-	-	-	0	u	*	*	*	The number of video pixels to remove on the right of the image.
DisplayWidth	4	[54][B0]	-	-	not 0	PixelWidth	u	*	*	*	Width of the video frames to display.
DisplayHeight	4	[54][BA]	-	-	not 0	PixelHeight	u	*	*	*	Height of the video frames to display.
DisplayUnit	4	[54][B2]	-	-	-	0	u			*	Type of the unit for DisplayWidth/Height (0: pixels, 1: centimeters, 2: inches).
AspectRatioType	4	[54][B3]	-	-	-	0	u			*	Specify the possible modifications to the aspect ratio (0: free resizing, 1: keep aspect ratio, 2: fixed).
ColourSpace	4	[2E][B5][24]	-	-	-	-	b	*	*		Same value as in AVI (32 bits).
GammaValue	4	[2F][B5][23]	-	-	> 0	-	f				Gamma Value.
FrameRate	4	[23][83][E3]	-	-	> 0	-	f	*	*	*	Number of frames per second. **Informational** only.
end video											
audio											
Audio	3	[E1]	-	-	-	-	m	*	*	*	Audio settings.
SamplingFrequency	4	[B5]	*	-	> 0	8000.0	f	*	*	*	Sampling frequency in Hz.
OutputSamplingFrequency	4	[78][B5]	-	-	> 0	Sampling Frequency	f	*	*	*	Real output sampling frequency in Hz (used for SBR techniques).
Channels	4	[9F]	*	-	not 0	1	u	*	*	*	Numbers of channels in the track.
ChannelPositions	4	[7D][7B]	-	-	-	-	b				Table of horizontal angles for each successive channel, see appendix.
BitDepth	4	[62][64]	-	-	not 0	-	u	*	*	*	Bits per sample, mostly used for PCM.
end audio											
content encoding											
ContentEncodings	3	[6d][80]	-	-	-	-	m	*	*		Settings for several content encoding mechanisms like compression or encryption.
ContentEncoding	4	[62][40]	*	*	-	-	m	*	*		Settings for one content encoding like compression or encryption.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1760
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ContentEncodingOrder	5	[50][31]	*	-	-	0	u	*	*	Tells when this modification was used during encoding/muxing starting with 0 and counting upwards. The decoder/demuxer has to start with the highest order number it finds and work its way down. This value has to be unique over all ContentEncodingOrder elements in the segment.
ContentEncodingScope	5	[50][32]	*	-	not 0	1	u	*	*	A bit field that describes which elements have been modified in this way. Values (big endian) can be OR'ed. Possible values: 1 - all frame contents, 2 - the track's private data, 4 - the next ContentEncoding (next ContentEncodingOrder. Either the data inside ContentCompression and/or ContentEncryption)
ContentEncodingType	5	[50][33]	*	-	-	0	u	*	*	A value describing what kind of transformation has been done. Possible values: 0 - compression, 1 - encryption
ContentCompression	5	[50][34]	-	-	-	-	m	*	*	Settings describing the compression used. Must be present if the value of ContentEncodingType is 0 and absent otherwise. Each block must be decompressable even if no previous block is available in order not to prevent seeking.
ContentCompAlgo	6	[42][54]	*	-	-	0	u	*	*	The compression algorithm used. Algorithms that have been specified so far are: 0 - zlib, 1 - bzlib, 2 - lzo1x 3 - Header Stripping,
ContentCompSettings	6	[42][55]	-	-	-	-	b	*	*	Settings that might be needed by the decompressor. For Header Stripping (ContentCompAlgo=3), the bytes that were removed from the beggining of each frames of the track.
ContentEncryption	5	[50][35]	-	-	-	-	m	*	*	Settings describing the encryption used. Must be present if the value of ContentEncodingType is 1 and absent otherwise.
ContentEncAlgo	6	[47][e1]	-	-	-	0	u	*	*	The encryption algorithm used. The value '0' means that the contents have not been encrypted but only signed. Predefined values: 1 - DES, 2 - 3DES, 3 - Twofish, 4 - Blowfish, 5 - AES
ContentEncKeyID	6	[47][e2]	-	-	-	-	b	*	*	For public key algorithms this is the ID of the public key the the data was encrypted with.
ContentSignature	6	[47][e3]	-	-	-	-	b	*	*	A cryptographic signature of the contents.
ContentSigKeyID	6	[47][e4]	-	-	-	-	b	*	*	This is the ID of the private key the data was signed with.
ContentSigAlgo	6	[47][e5]	-	-	-	0	u	*	*	The algorithm used for the signature. A value of '0' means that the contents have not been signed but only encrypted. Predefined values: 1 - RSA
ContentSigHashAlgo	6	[47][e6]	-	-	-	0	u	*	*	The hash algorithm used for the signature. A value of '0' means that the contents have not been signed but only encrypted. Predefined values: 1 - SHA1-160 2 - MD5

end content encoding

Cueing Data

IBNSDOCID: <XP 55246006A [>1007060413 03/http://www.matroska.org/technical/specs/index.html

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1761
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Cues	1	[1C][53][BB][6B]	-	-	-	-	m	*	*	*	A top-level element to speed seeking access. All entries are local to the segment. Should be mandatory for non "live" streams.
CuePoint	2	[BB]	*	*	-	-	m	*	*	*	Contains all information relative to a seek point in the segment.
CueTime	3	[B3]	*	-	-	-	u	*	*	*	Absolute timecode according to the segment time base.
CueTrackPositions	3	[B7]	*	*	-	-	m	*	*	*	Contain positions for different tracks corresponding to the timecode.
CueTrack	4	[F7]	*	-	not 0	-	u	*	*	*	The track for which a position is given.
CueClusterPosition	4	[F1]	*	-	-	-	u	*	*	*	The position of the Cluster containing the required Block.
CueBlockNumber	4	[53][78]	-	-	not 0	1	u	*	*	*	Number of the Block in the specified Cluster.
CueCodecState	4	[EA]	-	-	-	0	u			*	The position of the Codec State corresponding to this Cue element. 0 means that the data is taken from the initial Track Entry.
CueReference	4	[DB]	-	*	-	-	m			*	The Clusters containing the required referenced Blocks.
CueRefTime	5	[96]	*	-	-	-	u			*	Timecode of the referenced Block.
CueRefCluster	5	[97]	*	-	-	-	u				Position of the Cluster containing the referenced Block.
CueRefNumber	5	[53][5F]	-	-	not 0	1	u				Number of the referenced Block of Track X in the specified Cluster.
CueRefCodecState	5	[EB]	-	-	-	0	u				The position of the Codec State corresponding to this referenced element. 0 means that the data is taken from the initial Track Entry.
Attachment											
Attachments	1	[19][41][A4][69]	-	-	-	-	m	*	*		Contain attached files.
AttachedFile	2	[61][A7]	*	*	-	-	m	*	*		An attached file.
FileDescription	3	[46][7E]	-	-	-	-	s	*	*		A human-friendly name for the attached file.
FileName	3	[46][6E]	*	-	-	-	s	*	*		Filename of the attached file.
FileMimeType	3	[46][60]	*	-	-	-	s	*	*		MIME type of the file.
FileData	3	[46][5C]	*	-	-	-	b	*	*		The data of the file.
FileUID	3	[46][AE]	*	-	not 0	-	u	*	*		Unique ID representing the file, as random as possible.
FileReferral	3	[46][75]	-	-	-	-	b				A binary value that a track/codec can refer to when the attachment is needed.
Chapters											
Chapters	1	[10][43][A7][70]	-	-	-	-	m	*	*		A system to define basic menus and partition data. For more detailed information, look at the Chapters Explanation.
EditionEntry	2	[45][B9]	*	*	-	-	m	*	*		Contains all information about a segment edition.
EditionUID	3	[45][BC]	-	-	not 0	-	u	*	*		A unique ID to identify the edition. It's useful for tagging an edition.
EditionFlagHidden	3	[45][BD]	*	-	0-1	0	u	*	*		If an edition is hidden (1), it should not be available to the user interface (but still to Control Tracks). (1 bit)
EditionFlagDefault	3	[45][DB]	*	-	0-1	0	u	*	*		If a flag is set (1) the edition should be used as the default one. (1 bit)
EditionFlagOrdered	3	[45][DD]	-	-	0-1	0	u	*	*		Specify if the chapters can be defined multiple times and the order to play them is enforced. (1 bit)
ChapterAtom	3+	[B6]	*	*	-	-	m	*	*		Contains the atom information to use as the chapter atom (apply to all tracks).
ChapterUID	4+	[73][C4]	*	-	not 0	-	u	*	*		A unique ID to identify the Chapter.
ChapterTimeStart	4+	[91]	*	-	-	-	u	*	*		Timecode of the start of Chapter (not scaled).
ChapterTimeEnd	4+	[92]	-	-	-	-	u	*	*		Timecode of the end of Chapter (timecode excluded, not scaled).

BNSDOCID: <XP_____55246006A__I_> 100706041303/http://www.matroska.org/technical/specs/index.html

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1762
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ChapterFlagHidden	4+	[98]	*	-	0-1	0	u	*	*	If a chapter is hidden (1), it should not be available to the user interface (but still to Control Tracks). (1 bit)
ChapterFlagEnabled	4+	[45][98]	*	-	0-1	1	u	*	*	Specify wether the chapter is enabled. It can be enabled/disabled by a Control Track. When disabled, the movie should skip all the content between the TimeStart and TimeEnd of this chapter. (1 bit)
ChapterSegmentUID	4+	[6E][67]	-	-	>0	-	b	*	*	A segment to play in place of this chapter. Edition ChapterSegmentEditionUID should be used for this segment, otherwise no edition is used.
ChapterSegmentEditionUID	4+	[6E][BC]	-	-	not 0	-	b	*	*	The edition to play from the segment linked in ChapterSegmentUID.
ChapterPhysicalEquiv	4+	[63][C3]	-	-	-	-	u	*	*	Specify the physical equivalent of this ChapterAtom like "DVD" (60) or "SIDE" (50), see complete list of values.
ChapterTrack	4+	[8F]	-	-	-	-	m	*	*	List of tracks on which the chapter applies. If this element is not present, all tracks apply
ChapterTrackNumber	5+	[89]	*	*	not 0	-	u	*	*	UID of the Track to apply this chapter too. In the absense of a control track, choosing this chapter will select the listed Tracks and deselect unlisted tracks. Absense of this element indicates that the Chapter should be applied to any currently used Tracks.
ChapterDisplay	4+	[80]	-	*	-	-	m	*	*	Contains all possible strings to use for the chapter display.
ChapString	5+	[85]	*	-	-	-	8	*	*	Contains the string to use as the chapter atom.
ChapLanguage	5+	[43][7C]	*	*	-	eng	s	*	*	The languages corresponding to the string, in the bibliographic ISO-639-2 form.
ChapCountry	5+	[43][7E]	-	*	-	-	s	*	*	The countries corresponding to the string, same 2 octets as in Internet domains.
ChapProcess	4+	[69][44]	-	*	-	-	m	*	*	Contains all the commands associated to the Atom.
ChapProcessCodecID	5+	[69][55]	*	-	-	0	u	*	*	Contains the type of the codec used for the processing. A value of 0 means native Matroska processing (to be defined), a value of 1 means the DVD command set is used. More codec IDs can be added later.
ChapProcessPrivate	5+	[45][0D]	-	-	-	-	b	*	*	Some optional data attached to the ChapProcessCodecID information. For ChapProcessCodecID = 1, it is the "DVD level" equivalent.
ChapProcessCommand	5+	[69][11]	-	*	-	-	m	*	*	Contains all the commands associated to the Atom.
ChapProcessTime	6+	[69][22]	*	-	-	-	u	*	*	Defines when the process command should be handled (0: during the whole chapter, 1: before starting playback, 2: after playback of the chapter).
ChapProcessData	6+	[69][33]	*	-	-	-	b	*	*	Contains the command information. The data should be interpreted depending on the ChapProcessCodecID value. For ChapProcessCodecID = 1, the data correspond to the binary DVD cell pre/post commands.
Tagging										
Tags	1	[12][54][C3][67]	-	*	-	-	m	*	*	Element containing elements specific to Tracks/Chapters. A list of valid tags can be found here.
Tag	2	[73][73]	*	*	-	-	m	*	*	Element containing elements specific to Tracks/Chapters.
Targets	3	[63][C0]	*	-	-	-	m	*	*	Contain all UIDs where the specified meta data apply. It is empty to describe everything in the segment.

[BNSDOCID: <XP_____55246006A__I_> 100706041303/http://www.matroska.org/technical/specs/index.html

TargetTypeValue	4	[68][CA]	-	-	-	50	u	*	*	A number to indicate the logical level of the target (see TargetType).
TargetType	4	[63][CA]	-	-	-	-	s	*	*	An **informational** string that can be used to display the logical level of the target like "ALBUM", "TRACK", "MOVIE", "CHAPTER", etc (see TargetType).
TrackUID	4	[63][C5]	-	*	-	0	u	*	*	A unique ID to identify the Track(s) the tags belong to. If the value is 0 at this level, the tags apply to all tracks in the Segment.
EditionUID	4	[63][C9]	-	*	-	0	u	*	*	A unique ID to identify the EditionEntry(s) the tags belong to. If the value is 0 at this level, the tags apply to all editions in the Segment.
ChapterUID	4	[63][C4]	-	*	-	0	u	*	*	A unique ID to identify the Chapter(s) the tags belong to. If the value is 0 at this level, the tags apply to all chapters in the Segment.
AttachmentUID	4	[63][C6]	-	*	-	0	u	*	*	A unique ID to identify the Attachment(s) the tags belong to. If the value is 0 at this level, the tags apply to all the attachments in the Segment.
SimpleTag	3+	[67][C8]	*	*	-	-	m	*	*	Contains general information about the target.
TagName	4+	[45][A3]	*	-	-	-	8	*	*	The name of the Tag that is going to be stored.
TagLanguage	4+	[44][7A]	*	-	-	und	s	*	*	Specifies the language of the tag specified, in the Matroska languages form.
TagDefault	4+	[44][84]	*	-	0-1	1	u	*	*	Indication to know if this is the default/original language to use for the given tag. (1 bit)
TagString	4+	[44][87]	-	-	-	-	8	*	*	The value of the Tag.
TagBinary	4+	[44][85]	-	-	-	-	b	*	*	The values of the Tag if it is binary. Note that this cannot be used in the same SimpleTag as TagString.

Element Name	L	EBML ID	Ma	Mu	Rng	Default	T	1	2	W	Description

All top-levels elements (Segment and direct sub-elements) are coded on 4 octets, i.e. class D elements.

Appendix

Language Codes

Language codes can be either the 3 letters bibliographic ISO-639-2 form (like "fre" for french), or a language code mixed with a country code for specialities in languages (like "fre-ca" for canadian french).

Physical Types

Each level can have different meanings for audio and video. The ORIGINAL_MEDIUM tag can be used to specify a string for ChapterPhysicalEquiv = 60. Here is the list of possible levels for both audio and video :

ChapterPhysicalEquiv	Audio	Video	Comment
70	SET / PACKAGE	SET / PACKAGE	the collection of different media
60	CD / 12" / 10" / 7" / TAPE / MINIDISC / DAT	DVD / VHS / LASERDISC	the physical medium like a CD or a DVD
50	SIDE	SIDE	when the original medium (LP/DVD) has different sides
40	-	LAYER	another physical level on DVDs
30	SESSION	SESSION	as found on CDs and DVDs
20	TRACK	-	as found on audio CDs
10	INDEX	-	the first logical level of the side/medium

Block Structure

Size = 1 + (1-8) + 4 + (4 + (4)) octets. So from 6 to 21 octets.

Bit 0 is the most significant bit.

Frames using references should be stored in "coding order". That means the references first and then the frames referencing them. A consequence is that timecodes may not be consecutive. But a frame with a past timecode must reference a frame already known, otherwise it's considered bad/void.

There can be many Blocks in a BlockGroup provided they all have the same timecode. It is used with different parts of a frame with different priorities.

BNSDOCID: <XP_____55246006A__I_> 10070604130 3/http://www.matroska.org/technical/specs/index.html

Block Header

Offset	Player	Description
0x00+	must	Track Number (Track Entry). It is coded in EBML like form (1 octet if the value is < 0x80, 2 if < 0x4000, etc) (most significant bits set to increase the range).
0x01+	must	Timecode (relative to Cluster timecode, signed int16)
0x03+	-	**Flags** Bit 0-3 - Reserved, set to 0 Bit 4 - Invisible, the codec should decode this frame but not display it Bit 5-6 must - Lacing 00 : no lacing 01 : Xiph lacing 11 : EBML lacing 10 : fixed-size lacing Bit 7 - not used

Lace (when lacing bit is set)

0x00	must	Number of frames in the lace-1 (uint8)
0x01 / 0xXX	must*	Lace-coded size of each frame of the lace, except for the last one (multiple uint8). *This is not used with Fixed-size lacing as it is calculated automatically from (total size of lace) / (number of frames in lace).

(possibly) Laced Data

0x00	must	Consecutive laced frames

Lacing

Lacing is a mechanism to save space when storing data. It is typically used for small blocks of data (refered to as frames in matroska). There are 3 types of lacing : the Xiph one inspired by what is found in the Ogg container, the EBML one which is the same with sizes coded differently and the fixed-size one where the size is not coded. As an example is better than words...

Let's say you want to store 3 frames of the same track. The first frame is 800 octets long, the second is 500 octets long and the third is 1000 octets long. As these data are small, you can store them in a lace to save space. They will then be solved in the same block as follows:

Xiph lacing

Block head (with lacing bits set to 01)

Lacing head: Number of frames in the lace -1, i.e. 2 (the 800 and 400 octets one)

Lacing sizes: only the 2 first ones will be coded, 800 gives 255;255;255;35, 500 gives 255;245. The size of the last frame is deduced from the total size of the Block.

Data in frame 1

Data in frame 2

Data in frame 3

A frame with a size multiple of 255 is coded with a 0 at the end of the size, for example 765 is coded 255;255;255;0.

EBML lacing

In this case the size is not coded as blocks of 255 bytes, but as a difference with the previous size and this size is coded as in EBML. The first size in the lace is unsigned as in EBML. The others use a range shifting to get a sign on each value :

```
1xxx xxxx                                                                   - value -(2^6-1) to  2^6-1
                                                                             (ie 0 to 2^7-2 minus 2^6-1, half of the range)
01xx xxxx  xxxx xxxx                                                        - value -(2^13-1) to 2^13-1
001x xxxx  xxxx xxxx  xxxx xxxx                                             - value -(2^20-1) to 2^20-1
0001 xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx                                  - value -(2^27-1) to 2^27-1
0000 1xxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx                       - value -(2^34-1) to 2^34-1
0000 01xx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx            - value -(2^41-1) to 2^41-1
0000 001x  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx  xxxx xxxx - value -(2^48-1) to 2^48-1
```

Block head (with lacing bits set to 11)

Lacing head: Number of frames in the lace -1, i.e. 2 (the 800 and 400 octets one)

Lacing sizes: only the 2 first ones will be coded, 800 gives 0x320 0x4000 = 0x4320, 500 is coded as -300 : - 0x12C + 0x1FFF + 0x4000 = 0x5ED3. The size of the last frame is deduced from the total size of the Block.

Data in frame 1

Data in frame 2

Data in frame 3

Fixed-size lacing

In this case only the number of frames in the lace is saved, the size of each frame is deduced from the total size of the Block. For

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1765
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

example, for 3 frames of 800 octets each :

Block head (with lacing bits set to 10)
Lacing head: Number of frames in the lace -1, i.e. 2
Data in frame 1
Data in frame 2
Data in frame 3

SimpleBlock Structure

The SimpleBlock is very inspired by the Block structure. The main differences are the added Keyframe flag and Discardable flag. Otherwise everything is the same.

Size = 1 + (1-8) + 4 + (4 + (4)) octets. So from 6 to 21 octets.

Bit 0 is the most significant bit.

Frames using references should be stored in "coding order". That means the references first and then the frames referencing them. A consequence is that timecodes may not be consecutive. But a frame with a past timecode must reference a frame already known, otherwise it's considered bad/void.

There can be many Blocks in a BlockGroup provided they all have the same timecode. It is used with different parts of a frame with different priorities.

SimpleBlock Header

Offset	Player	Description
0x00+	must	Track Number (Track Entry). It is coded in EBML like form (1 octet if the value is < 0x80, 2 if < 0x4000, etc) (most significant bits set to increase the range).
0x01+	must	Timecode (relative to Cluster timecode, signed int16)
0x03+	-	**Flags** Bit / Player / Description 0 - Keyframe, set when the Block contains only keyframes 1-3 - Reserved, set to 0 4 - Invisible, the codec should decode this frame but not display it 5-6 must Lacing 00 : no lacing 01 : Xiph lacing 11 : EBML lacing 10 : fixed-size lacing 7 - Discardable, the frames of the Block can be discarded during playing if needed

Lace (when lacing bit is set)

0x00	must	Number of frames in the lace-1 (uint8)
0x01 / 0xXX	must*	Lace-coded size of each frame of the lace, except for the last one (multiple uint8). *This is not used with Fixed-size lacing as it is calculated automatically from (total size of lace) / (number of frames in lace).

(possibly) Laced Data

0x00	must	Consecutive laced frames

EncryptedBlock Structure

The EncryptedBlock is very inspired by the SimpleBlock structure. The main differences is that the raw data are Transformed. That means the data after the lacing definition (if present) have been processed before put into the Block. The laced sizes apply on the decoded (Inverse Transform) data. This size of the Transformed data may not match the size of the initial chunk of data.

The other difference is that the number of frames in the lace are not saved if "no lacing" is specified (bits 5 and 6 set to 0).

The Transformation is specified by a TransformID in the Block (must be the same for all frames within the EncryptedBlock).

Size = 1 + (1-8) + 4 + (4 + (4)) octets. So from 6 to 21 octets.

Bit 0 is the most significant bit.

Frames using references should be stored in "coding order". That means the references first and then the frames referencing them. A consequence is that timecodes may not be consecutive. But a frame with a past timecode must reference a frame already known, otherwise it's considered bad/void.

There can be many Blocks in a BlockGroup provided they all have the same timecode. It is used with different parts of a frame with different priorities.

EncryptedBlock Header

Offset	Player	Description
0x00+	must	Track Number (Track Entry). It is coded in EBML like form (1 octet if the value is < 0x80, 2 if < 0x4000, etc) (most significant bits set to increase the range).

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1766
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

0x01+	must	Timecode (relative to Cluster timecode, signed int16)
0x03+	-	**Flags** **Bit Player Description** 0 - Keyframe, set when the Block contains only keyframes 1-3 - Reserved, set to 0 4 - Invisible, the codec should decode this frame but not display it 5-6 must Lacing 00 : no lacing 01 : Xiph lacing 11 : EBML lacing 10 : fixed-size lacing 7 - Discardable, the frames of the Block can be discarded during playing if needed

Lace (when lacing bit is set)

0x00	must*	Number of frames in the lace-1 (uint8) *Only available if bit 5 or bit 6 of the EncryptedBlock flag is set to one.
0x01 / 0xXX	must*	Lace-coded size of each frame of the lace, except for the last one (multiple uint8). *This is not used with Fixed-size lacing as it is calculated automatically from (total size of lace) / (number of frames in lace).

(possibly) Laced Data

0x00	must	TransformID (EBML coded integer value). Value 0 = Null Transform
0x01+	must	Consecutive laced frames

Virtual Block

The data in matroska is stored in coding order. But that means if you seek to a particular point and a frame has been referenced far away, you won't know while playing and you might miss this frame (true for independent frames and overlapping of dependent frames). So the idea is to have a placeholder for the original frame in the timecode (display) order.

The structure is a scaled down version of the normal Block.

Virtual Block Header

Offset	Player	Description
0x00+	must	Track Number (Track Entry). It is coded in EBML like form (1 octet if the value is < 0x80, 2 if < 0x4000, etc) (most significant bits set to increase the range).
0x01+	must	Timecode (relative to Cluster timecode, signed int16)
0x03+	-	**Flags** **Bit Player Description** 7-0 - Reserved, set to 0

Copyright © 2005 - 2010 CoreCodec, Inc. All Rights Reserved.
Matroska, MKV, and the Matroska logos are either trademarks or registered trademarks of CoreCodec, Inc.
XHTML & CSS Valid.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1767
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Europäisches Patentamt
European Patent Office
Office européen des brevets

SUPPLEMENTARY EUROPEAN SEARCH REPORT

Application Number
EP 10 83 4935

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	EP 0 936 812 A1 (CANAL PLUS SA [FR]) 18 August 1999 (1999-08-18) * figure 3 * * paragraph [0047] * * paragraph [0073] - paragraph [0074] * * paragraph [0125] - paragraph [0126] * * paragraph [0132] - paragraph [0136] *	1-15	INV. H04N7/167 H04N21/235 H04N21/236 H04N21/434 H04N21/4405 H04N21/6334
X	WO 01/65762 A2 (TIVO INC [US]; PLATT DAVID C [US]; GOODMAN ANDREW [US]; ZENCHELSKY DAN) 7 September 2001 (2001-09-07) * page 3, lines 15-25 * * page 3, lines 35-38 * * page 5, lines 18-23 * * page 6, lines 17-19 * * page 7, lines 17-19 *	1-9, 13-15	
X	US 2007/192810 A1 (PRITCHETT THADDEUS C [US] ET AL) 16 August 2007 (2007-08-16) * figures 2-4 * * paragraphs [0018] - [0021] * * paragraphs [0023], [0026], [0031], [0032], [0040], [0041] *	1-15	TECHNICAL FIELDS SEARCHED (IPC) H04N
A	US 2004/139335 A1 (DIAMAND JOSEPH [US] ET AL) 15 July 2004 (2004-07-15) * paragraphs [0014] - [0017], [0026] - [0032] *	1-15	

The supplementary search report has been based on the last set of claims valid and available at the start of the search.

Place of search	Date of completion of the search	Examiner
The Hague	27 May 2014	Lefol, Damien

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C04)

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1768
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 10 83 4935

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

27-05-2014

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0936812 A1	18-08-1999	AT 225999 T	15-10-2002
		AU 754015 B2	31-10-2002
		AU 2295199 A	30-08-1999
		BR 9907878 A	31-10-2000
		CA 2318992 A1	19-08-1999
		CN 1296695 A	23-05-2001
		DE 69903408 D1	14-11-2002
		DE 69903408 T2	18-06-2003
		EP 0936812 A1	18-08-1999
		ES 2185311 T3	16-04-2003
		HK 1031070 A1	11-07-2003
		HR P20000487 A2	28-02-2001
		HU 0100651 A2	28-06-2001
		ID 25466 A	05-10-2000
		IL 137639 A	10-12-2006
		JP 2002514834 A	21-05-2002
		NO 20004063 A	13-10-2000
		PL 342260 A1	04-06-2001
		RU 2225681 C2	10-03-2004
		TR 200002348 T2	21-12-2000
		US 7239704 B1	03-07-2007
		WO 9941907 A1	19-08-1999
WO 0165762 A2	07-09-2001	AU 4197201 A	12-09-2001
		US 2004126093 A1	01-07-2004
		US 2009316895 A1	24-12-2009
		US 2012027208 A1	02-02-2012
		WO 0165762 A2	07-09-2001
US 2007192810 A1	16-08-2007	NONE	
US 2004139335 A1	15-07-2004	CA 2511914 A1	05-08-2004
		EP 1593265 A1	09-11-2005
		MX PA05007550 A	21-09-2005
		US 2004139335 A1	15-07-2004
		WO 2004066629 A1	05-08-2004

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

The examination is being carried out on the **following application documents**

Description, Pages

1-17	as published

Claims, Numbers

1-15	filed with entry into the regional phase before the EPO

Drawings, Sheets

1/8-8/8	as published

Reference is made to the following documents; the numbering will be adhered to in the rest of the procedure.

D1 EP 0 936 812 A1 (CANAL PLUS SA [FR]) 18 August 1999 (1999-08-18)

D2 WO 01/65762 A2 (TIVO INC [US]; PLATT DAVID C [US]; GOODMAN ANDREW [US]; ZENCHELSKY DAN) 7 September 2001 (2001-09-07)

D3 US 2007/192810 A1 (PRITCHETT THADDEUS C [US] ET AL) 16 August 2007 (2007-08-16)

1 **Independent Claims : Objections under Article 52(1) EPC and Article 54 EPC, lack of novelty**

The subject-matter of independent **claims 1 and 13** is not novel within the meaning of Article 54 EPC and the requirements of Article 52(1) EPC are therefore not met.

1.1 Concerning **claim 1**, document D1 discloses (the references in parentheses applying to this document):

- A method of playing back encrypted video on a playback device, where at least partially encrypted video data 22 is contained within a container file 20 and can be used to create an elementary bitstream 30 using cryptographic information 32, which can be used to decrypt the at least partially encrypted video data 22, the method comprising: (Fig. 3

shows a general overview of a system for transmission and reception of encrypted video data. The high level functioning of the playback of such a system is described in §0073)

- obtaining the cryptographic information 32; (§0134, line 1-3 details that the control word is extracted by the sim card)
- extracting the at least partially encrypted video data 22 from the container file to create an elementary bitstream 30; (§0134, line 7-10, a data stream is created using the scrambled video data and the re-encrypted ECM)
- the method characterized by: enciphering the cryptographic information 32; (§0134, line 6-8 the Control Word is encrypted)
- inserting the cryptographic information 32 in the elementary bitstream 30; (§0134, line 7-10, the previously encrypted Control Word CW is encapsulated in an ECM and inserted in the data stream with the still scrambled video data. §00135 further mentions that the encryption key used to wrap the CW is also encrypted, encapsulated in an EMM and added in the header of the bitstream)
- providing the elementary bitstream 30 to a video decoder 20; (§0136, line 9, the bitstream is descrambled and played)
- extracting the cryptographic information 32 from the elementary bitstream 30 at the video decoder 20; (§0136, line 1-4, the encrypted EMM is extracted from the start of the previously created bitstream)
- deciphering the cryptographic information 32; (§0136, line 4-8, the extracted EMM is used to decrypt the ECM contained in the bitstream)
- decrypting the elementary bitstream with the cryptographic information; and (§0136, line 8-9, the CW decrypted from the ECM is used to decrypt the video content)
- decoding the elementary bitstream for rendering on a display device using the video decoder. (§0136, line 9, the bitstream is played)

1.2 The objections raised for **claim 1** also apply to the corresponding system of **claim 13.** More precisely, document D1 discloses:

- A system for playback of encrypted video, where at least partially encrypted video data 22 is contained within a container file and can be used to create an elementary bitstream 30 using cryptographic information 32, which can be used to decrypt the at least partially encrypted video data 22, comprising a demultiplexer and a video decoder 20; the system characterized by: (Fig. 3 shows a general overview of a system for transmission and reception of encrypted video data. The high level functioning of the playback of such a system is described in §0073 where both a demultiplexer and a decoder are mentioned)
- the demultiplexer being configured to extract the at least partially encrypted video data 22 from the container file to create an elementary bitstream 30;(§0134, line 7-10, a data stream is created using the still scrambled video data and the re-encrypted ECM)
- the video decoder being configured to decrypt the elementary bitstream 30 using the cryptographic information 32 and (§0136, line 8-9, the CW decrypted from the ECM is used to decrypt the video content)
- decode the elementary bitstream 30 for rendering on a display device; wherein (§0136, line 9, the bitstream is played)
- the demultiplexer is configured to encipher the cryptographic information 32 and (§0134, line 6-8 the Control Word is encrypted)
- insert the enciphered cryptographic information in the elementary bitstream 30; and wherein (§0134, line 7-10, the previously encrypted Control Word CW is encapsulated in an ECM and inserted in the data stream with the still scrambled video data. §00135 further mentions that the encryption key used to wrap the CW is also encrypted, encapsulated in an EMM and added in the header of the bitstream)

- the video decoder 20 is configured to extract enciphered cryptographic information from an elementary bitstream 30 and (§0136, line 1-4, the encrypted EMM is extracted from the start of the previously created bitstream)
- to decipher the cryptographic information 32. (§0136, line 4-8, the extracted EMM is used to decrypt the ECM contained in the bitstream)

1.3 For the sake of completeness, it should be noted that:

1.3.1 D2 also discloses an encryption system where the encrypted transport stream transmitted by a headend is encrypted a second time in the receiver before being passed through memories or processors to avoid potential attack (page 3, line 15-25 and page 5, line 18-24). The stream can be partially encrypted and contains various informations and pointers to access the content (page 3, line 35-38). The encryption keys can be embedded in the stream (page 6, line 17-19) to be used prior to display to decrypt the two layers of encryption (page 7, line 17-19).

1.4 D3 also discloses a tuner device which receives content and works in collaboration with a controller device to manage access to the received content using DRM licenses and keys (Fig. 2, §0020 line 10-14, §0023 line 3-7). The media content received is a bitstream with headers containing various required information (§0021, line 1-7 and §0031, line 13-15) and the DRM keys can be added to this stream as inline data to be communicated to the controller (§0026, line 1-3). The media content, which can be partially encrypted, and the DRM keys can be further encrypted using an AES encryptor present in the tuner (§0032, line 16-21, §0018 line 1-2, §0026, line 6-10). The AES encryption used can be time varying to prevent repetition of encrypted data (§0018, line 5-14). The reverse decryption can be used to enable decoding and playing back the content (§0040 line 3-6, §0041line 1-7).

2 Dependent claims, lack of novelty

2.1 Dependent **claims 2-12 and 14-15** do not appear to contain any additional features which, in combination with the features of any claim to which they refer, meet the requirements of the EPC with respect to novelty.

2.2 **Claim 2** discloses that the cryptographic information is obtained from the container file. This can be readily contemplated in D1 where the bitstream N at the bottom of Fig. 3 shows the ECM in a container with (among others) Video data. The same ECM is received and re-encrypted in the decoder side as mentioned in §0134-§0136 of D1 (cf. detailed citations in section 1 above).

2.3 D1 further discloses in §0132 that (the EMM and) the ECM contains information necessary to check if the viewer have the required right to access the program and if so, to obtain the Control Word from the ECM. As such the cryptographic information comprises key information and information concerning the encrypted video data. **Claim 3** is thus not novel.

2.4 The ECM of D1 are mentioned to change on average every 10 seconds in line 1-3 of §0074. This ECM, which refers to a restricted block of video frames, is then used to obtain the CW associated with a particular section of the scrambled content in §0136, line 5-8. **Claim 4**, which mentions that the information is a reference to a block of encrypted data, is thus not novel.

2.5 The EMM and ECM inserted in the stream in D1 are inserted in headers and §0136, line 1-3 further mentions that the header containing the EMM is at the start or the file. The cryptographic information is thus delimited by an identifier and inserted before the encrypted video data as claimed in **claim 5**.

2.6 The extraction of the information using the identifier claimed in **claim 6** is also disclosed in §0136, line 1-5 where the EMM are extracted from the headers.

2.7 As mentioned in section 2.4 above, the ECM are used to obtain the CW associated with a particular section of the scrambled content which can then be descrambled and played (§0136, line 5-9). **Claim 7** is thus not novel.

2.8 As mentioned in section 2.4 above, the ECM are inserted in the stream and change every 10 seconds. This implies that the insertion in different locations of different key information disclosed in **claim 8** is not novel.

2.9 **Claim 9** discloses that the encrypted video data contains frames of encoded video which are encrypted. This can be readily contemplated in §0073, line 1-3 where the digital data, in this case MPEG-2 stream of compressed video (§0047, line 3-6), are scrambled by control word CW.

2.10 **Claim 10** mentions that the enciphering and deciphering are synchronised to avoid the cryptographic information being recoverable after a given time period. A similar method is disclosed in D1 §0133, line 3-5 where a session key is used to protect the CW. It is also mentioned in §0106, line 10-12 that such a session key may be changed periodically. As such **claim 10** is not novel.

2.11 D1 further mentions in §0125, line 4-6 that the session key mentioned in 2.10 is taken from any one of a number of such keys stored for this purpose in the system manager zone. As such it is considered that these key form a sequence of keys. The sequence of scrambling processes disclosed in **claim 11** is thus not novel.

2.12 D1 mentions in §0126, line 3-6 that the same session key required for the decryption is also present in the sim card. As such the inverse scrambling sequence can be decoded and **claim 12** is not novel.

2.13 The arguments detailed in 2.2 and 2.3 above for **claims 2 and 3** apply respectively to the system of **claims 14 and 15**. As such **claims 14 and 15** are not novel.

3 To the procedure

3.1 It is not clear what inventive contribution has been made to the art by the present application. However, if the applicants can identify such a contribution and wish to continue with the present application, they are invited to file a new set of claims directed towards such matter with suitable supporting arguments.

3.2 When revising the application, the applicants should also meet the following points:

(a) new independent claims which set out clearly and concisely the inventive contribution should be presented in a number which satisfies the requirement of Rule 43(2) EPC;

(b) the new claims should be divided in the appropriate two-part form, with a first portion correctly reflecting the nearest prior art disclosed by document D1 (Rule 43(1)(a) and (b) EPC);

(d) to meet the requirements of Rule 42(1)(b) EPC, document D1 should be identified in the description and the relevant background art disclosed therein should be briefly discussed;

(f) the spirit and scope clause on page 17 §0073 should be deleted (see the Guidelines for Examination, F-IV 4.4).

(g) The "incorporation by reference" close page 8 §0048 is not foreseen in the EPC. This statement implies that the scope of the application documents extend beyond that which was originally filed. The statement should therefore be removed in accordance with the Guidelines for Examination, F-III, 8.

3.3 The applicants are welcome to provide amended claims where added material is shown in underlined type and deleted material is shown in ~~strike-out type~~. An additional "clean" version should then also be provided to facilitate any potential office-internal scanning/OCR and printing process.

3.4 In order to meet the requirements of Rule 137(4) EPC and to facilitate the examination of the conformity of the amended application with the requirements of Article 123(2) EPC, the applicants are requested to clearly identify the amendments carried out, irrespective of whether they concern amendments by addition, replacement or deletion, and to indicate the passages of the application as filed on which these amendments are based.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1776
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Supported Media Formats

This document describes the media codec, container, and network protocol support provided by the Android platform.

As an application developer, you are free to make use of any media codec that is available on any Android-powered device, including those provided by the Android platform and those that are device-specific. **However, it is a best practice to use media encoding profiles that are device-agnostic.**

IN THIS DOCUMENT

Network Protocols
Core Media Formats
Video Encoding
Recommendations

SEE ALSO

Multimedia and Camera

KEY CLASSES

MediaPlayer
MediaRecorder

Network Protocols

The following network protocols are supported for audio and video playback:

- RTSP (RTP, SDP)
- HTTP/HTTPS progressive streaming
- HTTP/HTTPS live streaming draft protocol:
  - MPEG-2 TS media files only
  - Protocol version 3 (Android 4.0 and above)
  - Protocol version 2 (Android 3.x)
  - Not supported before Android 3.0

> **Note:** HTTPS is not supported before Android 3.1.

Core Media Formats

The table below describes the media format support built into the Android platform. Note that any given mobile device may provide support for additional formats or file types not listed in the table.

> **Note:** Media codecs that are not guaranteed to be available on all Android platform versions are accordingly noted in parentheses—for example "(Android 3.0+)".

Table 1. Core media format and codec support.

Type	Format / Codec	Encoder	Decoder	Details	Supported File Type(s) / Container Formats
	AAC LC	•	•	Support for mono/stereo/5.0/5.1 content with standard sampling rates from 8 to 48 kHz.	• 3GPP (.3gp) • MPEG-4 (.mp4, .m4a) • ADTS raw AAC (.aac, decode in Android 3.1+, encode in Android 4.0+, ADIF not supported) • MPEG-TS (.ts, not seekable, Android 3.0+)
	HE-AACv1 (AAC+)	• (Android 4.1+)	•		
	HE-AACv2 (enhanced AAC+)		•	Support for stereo/5.0/5.1 content with standard sampling rates from 8 to 48 kHz.	
	AAC ELD (enhanced	•	•	Support for mono/stereo content	

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1777
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

	low delay AAC)	(Android 4.1+)	(Android 4.1+)	with standard sampling rates from 16 to 48 kHz	
	AMR-NB	•	•	4.75 to 12.2 kbps sampled @ 8kHz	3GPP (.3gp)
	AMR-WB	•	•	9 rates from 6.60 kbit/s to 23.85 kbit/s sampled @ 16kHz	3GPP (.3gp)
Audio	FLAC		• (Android 3.1+)	Mono/Stereo (no multichannel). Sample rates up to 48 kHz (but up to 44.1 kHz is recommended on devices with 44.1 kHz output, as the 48 to 44.1 kHz downsampler does not include a low-pass filter). 16-bit recommended; no dither applied for 24-bit.	FLAC (.flac) only
	MP3		•	Mono/Stereo 8-320Kbps constant (CBR) or variable bit-rate (VBR)	MP3 (.mp3)
	MIDI		•	MIDI Type 0 and 1. DLS Version 1 and 2. XMF and Mobile XMF. Support for ringtone formats	• Type 0 and 1 (.mid, .xmf, .mxmf) • RTTTL/RTX (.rttl, .rtx)
				RTTTL/RTX, OTA, and iMelody	• OTA (.ota) • iMelody (.imy)
	Vorbis		•		• Ogg (.ogg) • Matroska (.mkv, Android 4.0+)
	PCM/WAVE	• (Android 4.1+)	•	8- and 16-bit linear PCM (rates up to limit of hardware). Sampling rates for raw PCM recordings at 8000, 16000 and 44100 Hz.	WAVE (.wav)
	JPEG	•	•	Base+progressive	JPEG (.jpg)
	GIF		•		GIF (.gif)
	PNG	•	•		PNG (.png)
Image	BMP		•		BMP (.bmp)
	WebP	• (Android 4.0+) (Lossless, Transparency, Android 4.2.1+)	• (Android 4.0+) (Lossless, Transparency, Android 4.2.1+)		WebP (.webp)
	H.263	•	•		• 3GPP (.3gp) • MPEG-4 (.mp4) • 3GPP (.3gp)

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1778
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Video	H.264 AVC	• (Android 3.0+)	•	Baseline Profile (BP)	• MPEG-4 (.mp4) • MPEG-TS (.ts, AAC audio only, not seekable, Android 3.0+)
	MPEG-4 SP		•		3GPP (.3gp)
	VP8	• (Android 4.3+)	• (Android 2.3.3+)	Streamable only in Android 4.0 and above	• WebM (.webm) • Matroska (.mkv, Android 4.0+)

Video Encoding Recommendations

Table 2, below, lists examples of video encoding profiles and parameters that the Android media framework supports for playback in the H.264 Baseline Profile codec. While table 3 lists examples that the framework supports for playback in the VP8 media codec.

In addition to these encoding parameter recommendations, a device's available *video recording* profiles can be used as a proxy for media playback capabilities. These profiles can be inspected using the `CamcorderProfile` (`//reference/android/media/CamcorderProfile.html`) class, which is available since API level 8.

Table 2. Examples of supported video encoding parameters for the H.264 Baseline Profile codec.

	SD (Low quality)	SD (High quality)	HD 720p (N/A on all devices)
Video resolution	176 x 144 px	480 x 360 px	1280 x 720 px
Video frame rate	12 fps	30 fps	30 fps
Video bitrate	56 Kbps	500 Kbps	2 Mbps
Audio codec	AAC-LC	AAC-LC	AAC-LC
Audio channels	1 (mono)	2 (stereo)	2 (stereo)
Audio bitrate	24 Kbps	128 Kbps	192 Kbps

Table 3. Examples of supported video encoding parameters for the VP8 codec.

	SD (Low quality)	SD (High quality)	HD 720p (N/A on all devices)	HD 1080p (N/A on all devices)
Video resolution	320 x 180 px	640 x 360 px	1280 x 720 px	1920 x 1080 px
Video frame rate	30 fps	30 fps	30 fps	30 fps
Video bitrate	800 Kbps	2 Mbps	4 Mbps	10 Mbps

For video content that is streamed over HTTP or RTSP, there are additional requirements:

- For 3GPP and MPEG-4 containers, the `moov` atom must precede any `mdat` atoms, but must succeed the `ftyp` atom.
- For 3GPP, MPEG-4, and WebM containers, audio and video samples corresponding to the same time offset may be no more than 500 KB apart. To minimize this audio/video drift, consider interleaving audio and video in smaller chunk sizes.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1779
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ISO/IEC JTC 1/SC 29

Date: 2006-10-27

ISO/IEC 14496-18:2004/Cor.1:2006(E)

ISO/IEC JTC 1/SC 29/WG 11

Secretariat:

Information technology — Coding of audio-visual objects — Part 18: Font compression and streaming, TECHNICAL CORRIGENDUM 1

Élément introductif — Élément central — Partie 18: Titre de la partie

Document type: International Standard
Document subtype: Technical Corrigendum
Document stage: (60) Publication
Document language: E

STD Version 2.1

3NSCOCID: <XP_____3001S158A__I_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1780
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

© ISO/IEC 2006 — All rights reserved

INSCOCID: <XP\.....3001515\8A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1781
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Foreword

Technical Corrigendum 1 to ISO/IEC 14496-18:2004 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 29, Coding of Audio, Picture, Multimedia and Hypermedia Information.

© ISO/IEC 2006 — All rights reserved

INSDOCID: <KP_____3001518A__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1782
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

BNSDOCID: <XP_____3001515BA_1_>

Information technology — Coding of audio-visual objects — Part 18: Font compression and streaming, TECHNICAL CORRIGENDUM 1

In Subclause 4.2.4.7 "Triplet encoding of X and Y Coordinates", replace the first twenty lines of the "Table 9 – Coordinate Encoding Table" with the following:

0	2	0	8	N/A	0	N/A	+
1					0		–
2					256		+
3					256		–
4					512		+
5					512		–
6					768		+
7					768		–
8					1024		+
9					1024		–
10	2	8	0	0	N/A	+	N/A
11				0		–	
12				256		+	
13				256		–	
14				512		+	
15				512		–	
16				768		+	
17				768		–	
18				1024		+	
19				1024		–	

In subclause 5.2.1 "Overview", replace the text of the first paragraph:

A font data access unit conveys data for one single font, or a subset thereof. Two access unit formats are specified, discriminated by the DecoderSpecificInfo for the font data stream (see subclause 5.4). Only one of these access unit formats can be selected for a given font data stream.

with the following text:

A font data access unit conveys data for one single font, or a subset thereof. Fonts may support a wide variety of different languages and may have a large number of glyphs. While there are some fonts that support, for example, full Latin character set, there are other fonts that support only English alphabet, which can be considered a subset of Latin characters.

When a font is used for text rendering, only a limited number of glyphs can be utilized, and the transmission of the whole font may not be necessary. In order to reduce the amount of data sent, and to improve the efficiency of the font data streams, fonts can be reduced in size (subsetted) by keeping only those glyphs that are utilized in a particular text fragment, and the additional required general font information. Therefore, the font subsets are complete, self-sufficient and fully functional fonts containing a limited number of glyphs. A font subset shall be conformant to the `fontFormat` specified in the header of the access unit. When the value of `fontFormat` is `0x01`, `0x02` or `0x03`, the content of

© ISO/IEC 2006 — All rights reserved

ENSDOCID: <XP_____3001618A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1784
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

`fontData` shall be conformant to OpenType specification, containing, at least, all required tables specified by the OpenType specification.

Two access unit formats are specified, discriminated by the DecoderSpecificInfo for the font data stream (see subclause 5.4). Only one of these access unit formats can be selected for a given font data stream.

In subclause 5.2.1 "Overview", replace the text of the second and third paragraphs:

The basic access unit format permits to convey font data without any header information. The `DecoderSpecificInfo` provides the identification of the font and, possibly, the subset thereof that is conveyed. The content of subsequent basic access units in such a font data stream shall replace the font or font subset data of previous basic access units.

The enhanced access unit format provides a self-contained format where an access unit header contains all necessary information about the font data conveyed in this access unit.

with the following text:

The basic access unit format permits to convey font data without any header information. The `DecoderSpecificInfo` provides the identification of the font and, possibly, the subset thereof that is conveyed. The content of subsequent basic access units in such a font data stream shall replace the font or font subset data of previous basic access units with the same `fontName` and `fontSubsetID` (if applicable).

The enhanced access unit format provides a self-contained format where an access unit header contains all necessary information about the font data conveyed in this access unit. The content of subsequent enhanced access unit replaces or complements the font or font subset of the previous basic or enhanced access unit with the same `fontName` and `fontSubsetID`, based on the `fontSubsetExtensionFlag`, as described in subclause 5.2.3.This functionality allows incremental updates of font data and efficient font streaming implementations.

In subclause 5.2.3 "Semantics", replace the current description of:

`fontSubsetExtensionFlag` – If set to one indicates that the font data in this access unit is an extension of the previously sent font subset with the same `fontName` and `fontSubsetID`. Otherwise the font subset replaces any previous font subset with the same `fontName` and `fontSubsetID`.

with the following text:

`fontSubsetExtensionFlag` – If set, indicates that the font data in this access unit is an extension of the previously sent font subset with the same `fontName` and `fontSubsetID`. The data presented in the current access unit extends the previously sent font subset data, i.e. a new font subset provides additional glyphs that complement previously sent font subset with the same font name and font subset ID. These two subsets can either be merged in the decoder, or can be used concurrently as two different font subsets of the same original font. If `fontSubsetExtensionFlag` is not set, the font subset replaces any previous font subset with the same `fontName` and `fontSubsetID`.

© ISO/IEC 2006 – All rights reserved
INSDOCID: <XP_____3001S156A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1785
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

In Annex A "Patent Statements", replace the table containing company names with the following:

Company
Apple Computer
Microsoft Corporation
Monotype Imaging Inc.

© ISO/IEC 2006 — All rights reserved

BNSDOCID: <XP_____30015138A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1786
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

© ISO/IEC 2004 — All rights reserved

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC 1/SC 29/WG 11 **N6215**

December 2003, Hawaii, USA

Source:	Systems
Title:	Text of ISO/IEC FDIS 14496-18 Font Compression and Streaming
Status:	Approved at 67th meeting
Editor:	Vladimir Levantovsky

Document type: International Standard
Document subtype:
Document stage: (50) Approval
Document language: E

INSDOCID: <XP_____30013169A_1_>

ISO/IEC JTC 1/SC 29

Date: 2004-01-7

ISO/IEC FDIS 14496-18:2004(E)

ISO/IEC JTC 1/SC 29/WG 11

Secretariat:

Information technology — Coding of audio-visual objects — Part 18: Font compression and streaming

Technologies de l'information — Codage des objets audiovisuels — Partie 18: Compression et transmission de polices de caractères

BNSDOCID: <XP_____3001815GA_1_>

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

ISDCC:ID: <XP_____30018158A_I_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1790
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Content

1 Scope 1

2 Normative References 1

3 Font Data Format 1

4 Font Compression Technology 2
4.1 Overview 2
4.2 Compressed Font Format Specification 2
4.2.1 Introduction 2
4.2.2 Data Types 2
4.2.3 Compressed Font Format 2
4.2.4 Glyph Data Compression 5

5 Font Data Stream 12
5.1 Structure of the Font Data Stream 12
5.2 Access Unit Definition 12
5.2.1 Overview 12
5.2.2 Syntax 12
5.2.3 Semantics 13
5.3 Time Base for Font Data Streams 13
5.4 Font Data Decoder Configuration 13
5.4.1 Syntax 13
5.4.2 Semantics 14
5.5 Accessing the Font Data 14

6 Text Profiles and Levels 15
6.1 Simple Text Profile and Levels 15
6.1.1 Application area 15
6.1.2 List of tools/functionalities 15
6.1.3 Comparison with existing profiles 15
6.1.4 Simple Text Profile @ Level 1 Definition 15
6.1.5 Simple Text Profile @ Level 2 Definition 15
6.1.6 Simple Text Profile @ Level 3 Definition 15
6.2 Advanced Simple Text Profile and Levels 16
6.2.1 Application area 16
6.2.2 List of tools/functionalities 16
6.2.3 Comparison with existing profiles 16
6.2.4 Advanced Simple Text Profile @ Level 1 Definition 16
6.2.5 Advanced Simple Text Profile @ Level 2 Definition 16
6.2.6 Advanced Simple Text Profile @ Level 3 Definition 16
6.3 Main Text Profile and Levels 17
6.3.1 Application area 17
6.3.2 List of tools/functionalities 17
6.3.3 Comparison with existing profiles 17
6.3.4 Main Text Profile @ Level 1 Definition 17
6.3.5 Main Text Profile @ Level 2 Definition 17
6.3.6 Main Text Profile @ Level 3 Definition 17

© ISO/IEC 2004 — All rights reserved

IBNSDOCID: <XP_____30018158A__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1791
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14496-18 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 29, *Coding of Audio, Picture, Multimedia and Hypermedia Information*.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

— *Part 1: Systems*

— *Part 2: Visual*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Delivery Multimedia Integration Framework (DMIF)*

— *Part 7: Optimized software for MPEG-4 visual tools*

— *Part 8: Carriage of ISO/IEC 14496 content over IP networks*

— *Part 9: Reference hardware description*

— *Part 10: Advanced video coding*

— *Part 11: Scene description and application engine*

— *Part 12: ISO base media file format*

— *Part 13: Intellectual Property Management and Protection (IPMP) extentions*

— *Part 14: MP4 file format*

— *Part 15: Advanced Video Coding (AVC) file format*

© ISO/IEC 2004 — All rights reserved

NSDOCID: <XP_____3001315A__I_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1792
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

— *Part 16: Animation framework extension (AFX)*

— *Part 17: Streaming text format*

— *Part 18: Font compression and streaming*

— *Part 19: Synthesized texture stream*

© ISO/IEC 2004 — All rights reserved

BNSDOCID: <XP_____3001315GA__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1793
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Introduction

ISO/IEC 14496 specifies a system for the communication of interactive audio-visual scenes. The specification includes the following elements:

1. the coded representation of natural or synthetic, two-dimensional (2D) or three-dimensional (3D) objects that can be manifested audibly and/or visually (audio-visual objects) (specified in part 1,2 and 3 of ISO/IEC 14496);

2. the coded representation of the spatio-temporal positioning of audio-visual objects as well as their behaviour in response to interaction (scene description, specified in part 11 of ISO/IEC 14496);

3. the coded representation of information related to the management of data streams (synchronization, identification, description and association of stream content, specified in part 11 of ISO/IEC 14496);

4. a generic interface to the data stream delivery layer functionality (specified in part 6 of ISO/IEC 14496);

5. an application engine for programmatic control of the player: format, delivery of downloadable Java byte code as well as its execution lifecycle and behaviour through APIs (specified in part 11 of ISO/IEC 14496); and

6. a file format to contain the media information of an ISO/IEC 14496 presentation in a flexible, extensible format to facilitate interchange, management, editing, and presentation of the media.

The information representation, specified in ISO/IEC 14496-1 and in ISO/IEC 14496-11, describes the means to create an interactive audio-visual scene in terms of coded audio-visual information and associated scene description information. The encoded content is presented to a terminal as the collection of elementary streams. Elementary streams contain the coded representation of either audio or visual data or scene description information or user interaction data. Elementary streams may as well themselves convey information to identify streams, to describe logical dependencies between streams, or to describe information related to the content of the streams. Each elementary stream contains only one type of data.

Elementary streams are decoded using their respective stream-specific decoders. The audio-visual objects are composed according to the scene description information and presented by the terminal's presentation device(s). All these processes are synchronized according to the systems decoder model (SDM) using the synchronization information provided at the synchronization layer.

The scene description stream identifies different types of objects, such as audio, visual, 2D and 3D graphics, etc. that define a scene composition of the content. Among these objects, the essential part of almost any multimedia presentation is text objects that are created utilizing specific custom fonts. Font selection determines the appearance of a text in multimedia content and it's the most critical factor that assures text legibility and readability. It also plays critical role in the overall scene composition since the metric properties of a font are used for textual parts of multimedia content layout. Many thousands of fonts are available today for use in content creation and in order to assure correct appearance and layout of a content the font data have to be included (embedded) with the text objects as part of the multimedia presentation.

Font data compression and streaming technology presented in this document provide efficient mechanism to embed font data in MPEG-4 encoded presentations.

© ISO/IEC 2004 — All rights reserved

ISOCOD: <XP_____30013183A__I_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1794
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Information technology — Coding of audio-visual objects — Part 18: Font compression and streaming

1 Scope

This part of ISO/IEC 14496 specifies functionalities for the communication of font data as part of the MPEG-4 encoded audio-visual presentation. More specifically, it defines:

1. Font format representation that is utilized for font data encoding (OpenType);
2. Font compression technology for TrueType and OpenType fonts with TrueType outlines; and
3. The coded representation of information in font data streams.

2 Normative References

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this part of ISO/IEC 14496. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 14496 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 14496-1, *Information technology -- Coding of audio-visual objects -- Systems*

ISO/IEC 14496-11, *Information technology -- Coding of audio-visual objects -- Scene description and application engine*

The OpenType Specification - available on the Microsoft Typography website at <http://www.microsoft.com/typography/otspec/default.htm> or the Adobe Solutions Network website at <http://partners.adobe.com/asn/tech/type/opentype/index.jsp>

3 Font Data Format

In order to guarantee the original appearance of the content, to preserve corporate branding and identity in streaming multimedia presentations and to provide support for all languages, MPEG-4 supports text rendering utilizing rich formatting capabilities and custom fonts.

MPEG-4 adopts OpenType®[i], version 1.4, as its font data format for the purposes of uniform font data transmission and predictable text rendering. OpenType has emerged as the font solution for high-quality text processing, multimedia applications and cross platform internet document portability. OpenType is a full-featured font format that enables the highest quality of text rendering on low-resolution displays, advanced typographic features and international character support. It is fully compatible with the existing and widely adopted TrueType™[ii] fonts.

MPEG-4 requires fonts to contain a Unicode character map ('cmap') table.

© ISO/IEC 2004 — All rights reserved

SDOCID: <XP_____3001515A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1796
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4 Font Compression Technology

4.1 Overview

Transmission of the font data through bandwidth-limited channels requires efficient compression techniques be applied that allow reducing font data size without sacrificing quality of text rendering and advanced typographic features supported by the font. MPEG-4 adopts MicroType® Express[iii] 2.0 font compression technology, which allows lossless compression of OpenType fonts with TrueType outlines and provides the capability to render the text without decompressing the whole font, which significantly reduces memory requirements. The specification of this compression mechanism is presented below.

4.2 Compressed Font Format Specification

4.2.1 Introduction

This part of the document describes the compressed font format that is designed for compact binary representation of lossless-compressed TrueType and OpenType with TrueType outlines fonts (TTF) and font collections (TTC). This font compression technology for OpenType TrueType fonts is designed to minimize memory usage for font data storage in resource-constrained environments, where both ROM and RAM memory savings are critical, and build relatively small font files without sacrificing the quality, features and capabilities of TrueType and OpenType fonts.

The compressed font format provides random access capabilities into the font data and allows font rasterizer to render glyphs directly from the compressed font tables without decompressing the whole font while maintaining the highest character quality. The selection and number of compressed tables in a font file may differ from one font to another. Any number of tables can be compressed in the same font (from only one to all), which allows application developers to optimize "memory savings – performance of font rendering engine" trade-off.

Compression algorithms, applied to font tables, take advantage of advanced knowledge of OpenType font tables structure and meaning of data (text, TrueType instructions, glyph data, etc.). Separation of these objects into groups with distinctly different statistical distributions allows employing a number of compression techniques, which results in producing most compact lossless-compressed data. These techniques are especially effective when compressing of the glyph data ('glyf' table), which is typically the biggest table in the TTF file. The details of the glyph data compression are described in subclause 4.2.4 of this specification.

4.2.2 Data Types

The compressed font format follows the same SFNT structure (used in both TrueType and OpenType) that provides a "wrapper" for a collection of tables in a general and extensible manner. Rasterizers use a combination of tables contained in the font to render glyph outlines. In order to simplify the process of traversing the tables in a font, the beginning of each table is aligned on the 4-byte boundary and each table data is padded with zeros.

Most tables have version numbers, and the version number for the entire font is contained in the Table Directory. The compressed font format specification uses the same data types that are used in the OpenType specification. For all details on data types and version number format please see the OpenType font file specification.

4.2.3 Compressed Font Format

4.2.3.1 The Compressed Font Header

The compressed font file format is the same as an OpenType font file, which always begins with the Offset Table followed by the Table Directory entries. If the font file is TTC file, the offset to each Table Directory is defined in TTC Header. If the font file contains only one font, the Offset Table will begin at byte 0 of the file.

© ISO/IEC 2004 — All rights reserved
BNSDOCID: <XP_____30018168A__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1797
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

For details about an OpenType font Table directory and TTC header structure please see "Organization of an OpenType Font" in the OpenType font file specification.

4.2.3.2 OpenType Tables

The OpenType font file specification defines the set of required tables to be used in an OpenType font with TrueType outlines. When this font compression mechanism is used with OpenType fonts with TrueType outlines, the ACT3 additional table described in the following section is required. The tag describing this table is '**act3**'.

Other tables may be defined but are not required. OpenType fonts may also contain bitmaps of glyphs, in addition to outlines, as well as other optional tables that support vertical layout and advanced typographic features. For a complete list of OpenType font tables, their definitions and data structures see The OpenType font file specification.

4.2.3.3 ACT3 Table Structure

The 'ACT3' table defines the mapping of all tables, present in the font, to the collection of compressed and uncompressed blocks of data in a font file. Since the herein compressed font file format allows a mixture of both compressed and uncompressed data blocks to be present in the file, the mapping of the data blocks relative to the original OpenType TrueType table structure will be as follows:

- all consecutive uncompressed tables of the original font file will be referenced as a contiguous block of uncompressed data by a single entry in the ACT3 table;
- each compressed table of the original font file will be referenced as a block of compressed data by a single entry in the ACT3 table.

NOTE: The 'act3' table is required to be present in the compressed font OpenType Table Directory. If the font file is TrueType collection, then the table will only be part of the first font Table Directory and will contain mapping information for all fonts in TrueType Collection font file.

Table 1 – 'act3' table syntax

Data Type	Syntax	Description and Comments
	/* ACT3 Table Header*/	
USHORT	TableLength	Length of the 'act3' table padded to 4-byte boundary
ULONG	CompFontSize	Size of the compressed font file
ULONG	UncompFontSize	Original size of the uncompressed font file after CTF compression was applied to glyph data.
ULONG	GlyfStart	Offset to the start of compressed 'glyf' table in uncompressed font file
ULONG	GlyfEnd	Offset to the position immediately following the end of compressed 'glyf' table in uncompressed font file
USHORT	NumEntries	Number of different compressed/uncompressed data blocks in the font file
	/* ACT3 Table Entries*/	
	for (i=0; i<numEntries; i++)	
	{	
ULONG	BlockStart	Offset to the start of uncompressed data block in original font file
ULONG	BlockLength	Length of uncompressed data block in original font file
ULONG	CompBlockStart	Offset to the start of data block in the compressed font file
USHORT	IsCompressed	Flag indicating whether block is compressed (non-zero if block is

© ISO/IEC 2004 — All rights reserved

NSDOCID: <XP_____30013153A_i_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1798
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
                                                compressed, INTERVAL =
                                                isCompressed )
                }
```

4.2.3.4 Structure of Compressed Block

Each compressed table in the original OpenType TrueType font or font collection file is represented by the single block of data in the ACT3 table. The data is losslessly compressed using canonical Huffman codes. When the table is compressed, an additional set of data is generated in order to enable the decoder perform random access into compressed data stream and selectively decompress only those chunks of data that are requested by the font rendering engine.

Random access in the compressed dataset is achieved by providing additional information on the corresponding positions of the start of Huffman codeword in the compressed data stream (bitmarks) and the start of a token in the uncompressed data. This relationship is established through predefined intervals between reference points in the original data.

The size of the interval between reference points affects both the compression efficiency of the encoder (a smaller interval would require more data be added to the output compressed data stream) and font rendering performance (a bigger interval would require font engine to decompress larger dataset in order to read the requested data). Therefore, a careful consideration should be given to selection of the interval between random access entry points. The analysis of the TrueType features, their sizes and empirical rendering performance results for Latin fonts showed that the optimal placement of reference points is achieved at the **INTERVAL** = 128 bytes.

The block of compressed data has **RAC3** tag and consists of the following data segments:

- The dictionary of tokens which were defined by the encoder;
- Definition of the canonical Huffman codes;
- Locations of the random access pointers to the compressed bitstream and original data;
- Compressed data bitstream.

Table 2 – RAC3 block format

Data Type	Syntax	Description and Comments
	`/* RAC3 Block Header */`	
TAG	`RACtag`	Compressed data ID string: 'R''A''C'3 (0x52414303)
ULONG	`compDataSize`	Size of the compressed data
ULONG	`uncompDataSize`	Original size of the uncompressed block of data
	`/* RAC3 Dictionary */`	
ULONG	`numTokens`	Number of tokens in the dictionary.
BYTE	`tokenLength[numTokens]`	Array of length for each token in the dictionary (in bytes).
BYTE	`offsetLength`	Length of the tokenOffset from the beginning of the tokenString array to the start of the token (in bytes)
	`if (offsetLength ==0)`	Tokens are non-overlapping, each
	`{`	tokenOffset into the tokenString can be inferred from the array of tokenLength[].
ULONG	`  tokenStringSize`	Length of the tokenString in bytes.
BYTE	`  tokenString[tokenStringSize]`	Dictionary of the tokens.
	`} else`	We have overlapping tokens, offsets
	`{`	into the tokenString are explicitly specified in the tokenOffset[].
BYTE	`  tokenOffset [(numTokens*offsetLength)]`	Array of offsets for each token in the tokenString. Each element of the

© ISO/IEC 2004 — All rights reserved
BNSDOCID: <XP_____8001316SA_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1799
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

		array is offsetLength bytes long.
ULONG	tokenStringSize	Length of the tokenString in bytes.
BYTE	tokenString[tokenStringSize]	Dictionary of the tokens.
	]	
	/* Canonical Huffman Codes */	
BYTE	numGroups	Number of Huffman code length groups (can be from 1 to 32)
BYTE	codeLength[numGroups]	Array of code length in each Huffman group.
ULONG	firstGroupCode[numGroups]	Array of the first code in each Huffman group.
ULONG	lastGroupCode[numGroups]	Array of the last code in each Huffman group.
	/* Random Access Pointers */	
ULONG	numIntervals	Number of **INTERVAL** byte intervals in the uncompressed source data
ULONG	bitMarks[numIntervals]	Bit offset to the start of the nearest Huffman codeword from the beginning of compressed data block corresponding to n^{th} interval.
BYTE	deltaBytes[numIntervals]	Offset to the start of the nearest Huffman token from the beginning of uncompressed source data = **INTERVAL** * n – deltaBytes[n]
ULONG	BitStringOffset	Bit offset to the start of the compressed bitString[] from the beginning of the compressed data block.
BYTE	BitString[compDataSize-bitStringOffset/8]	Compressed data bitstream.

4.2.4 Glyph Data Compression

4.2.4.1 Overview

The 'glyf' table contains all the information about simple and composite glyphs in OpenType TrueType font. Simple glyphs are recorded as a collection of data segments representing number of contours, bounding box, TTF instructions for glyph scaling and arrays of data point coordinates. If composite characters are present, they contain the array of simple glyph indexes along with the scaling information and TTF instructions for the composite glyph. These datasets have distinctly different statistical distributions of values and, sometimes, significant levels of redundancy. The use of internal dependencies of glyph data segments and knowledge of TrueType rasterizer behaviour allows the application of additional compression algorithms to reduce glyph data set without any affect on the functionality of glyph rendering and the output quality of TrueType rasterizer.

4.2.4.2 Additional Data Types

TrueType (SFNT) file format often uses USHORT and SHORT data types to provide enough storage space for numerical data. However, in the majority of cases the stored number is rather small and can be recorded within one byte. In order to reduce storage size for TTF glyph data while keeping byte alignment of data records we introduce **255USHORT** and **255SHORT** data type that records the numerical **Value** as follow:

Table 3 – 255USHORT Data Type

Data Type	Syntax	Description and Comments
BYTE	Code	**Value** = Code; /* [0..255] */
	if (Code == 255)	
BYTE	Value1	**Value** = 253 + Value1; /* [253..508] */

© ISO/IEC 2004 — All rights reserved

ISDOCID: <XP_____30013153A__I_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1800
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

	`else if (Code == 254)`	
`BYTE`	`Value1`	**Value** = 506 + Value1; /* [506..761] */
	`else if (Code == 253)`	
`USHORT`	`Value`	**Value**; /* [0..65535] */

Table 4 – 255SHORT Data Type

Data Type	Syntax	Description and Comments
`BYTE`	`Code`	Code; /* [0..255] */
	`if (Code < 250) {}`	**Value** = Code; /* [0..249] */
	`else if (Code == 250)`	Flip sign code.
`BYTE`	`Value1`	**Value** = (SHORT)(~(Value1 + 1));
	`else if (Code == 253)`	Word data code. Codes 251 and 252 are reserved for "Hop Codes", see below and subclause 4.2.4.3 for details.
`SHORT`	`Value`	**Value**; /* [-32768..32767] */
	`else if (Code >= 254)`	
`BYTE`	`Value1`	**Value** = Value1+256*(255-Code)+250;
	`else if (Code==251 || Code==252)`	**Value** = Value1+256*(255-Code)+250;
	`{}`	Reserved **HopCode** values;

4.2.4.3 Glyph Records

4.2.4.3.1 Overview

Glyph records are located in the 'glyf' table of OpenType font. Glyphs are recorded as a set of bounding box values, TrueType instructions, flags and outline data points. The glyph record size can be reduced with no effect on the quality of the glyph rendering. This compressed font format converts simple TTF glyph table records to lossless Compact Table Format (CTF).

4.2.4.3.2 Bounding Box

In most cases, the bounding box information can be computed from the data point coordinates and, therefore, bounding box information may be omitted in glyph record.

4.2.4.3.3 Contour End Points

TrueType glyphs store actual numbers of the end point for each contour. CTF stores the number of points in each contour instead, which is typically less then 255. Translation to the endpoint number is simply the cumulative sum of point count in all preceding contours.

4.2.4.3.4 Instructions

TrueType glyphs contain instructions for glyph scaling. Glyph programs typically start with a burst of various PUSH instructions to initialize the stack for a TrueType instructions belonging to that glyph. TrueType provides four different types of PUSH instructions that occupy 18 different opcodes. In order to optimize data storage size, the CTF format stores only data to be pushed onto the stack. The translation process back to TTF generates a new sequence of optimized TrueType PUSH instructions that results in the identical functionality of glyph program.

The storage format of the initial push data utilizes additional **Hop Codes** that make use of the specific data patterns observed in TrueType data. It capitalizes on a specific distribution of the data that a traditional compression technologies do not take advantage of. A typical TrueType data sequence contains value stream that has the same value repeating periodically among other values, e.g. A, x1, A, x2, A, x3, etc. Each value is

© ISO/IEC 2004 — All rights reserved
BNSDOCID: <XP_____3001816SA__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1801
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

of type SHORT and occupies 2 bytes of storage. An additional set of Hop Codes allows the reduction of the dataset size by applying the following transformation:

- **A, x1, A, x2, A** – translated to: **A, x1, Hop3Code, x2**. (Hop3Code = 0xFB);
- **A, x1, A, x2, A, x3, A** – translated to **A, x1, Hop4Code, x2, x3**, (Hop4Code = 0xFC).

4.2.4.3.5 Glyph Program Table

The following table describes the glyph program in CTF format:

Table 5 – Glyph Program Table

Data Type	Syntax	Description and Comments
255USHORT	pushCount	Number of data items in the push data stream.
255SHORT	pushData[pushCount]	See sample C function for HopCodes processing.
255USHORT	codeSize	Glyph program remaining code size.
BYTE	instructions[codeSize]	Remaining TrueType instructions.

4.2.4.3.6 Translation of HopCodes

The following C code can be used to translate sequence of HopCodes:

```
CHAR *input;
int i;
   BYTE code;
   SHORT A;
SHORT pushCount = Read255Short(input);
SHORT *pushData  = (SHORT *) malloc(sizeof(SHORT) * pushCount);
for ( i = 0; i < pushCount; ) {
    code = *input;
    if (code == Hop3Code ) {
        A = pushData[i-2];
        input++;                    /* Hop3Code */
        pushData[i++] = A;          /* A */
        pushData[i++] = Read255Short(input); /* X2 */
        pushData[i++] = A;          /* A */
    } else if (code == Hop4Code ) {
        A = pushData[i-2];
        input++;                    /* Hop4Code */
        pushData[i++] = A;          /* A */
        pushData[i++] = Read255Short(input); /* X2 */
        pushData[i++] = A;          /* A */
        pushData[i++] = Read255Short(input); /* X3 */
        pushData[i++] = A;          /* A */
    } else {
        pushData[i++] = Read255Short(input); /* Data */
    }
}
```

4.2.4.4 CTF Glyph Data

The following table contains information that describes the glyph data in CTF format.

Table 6 – CTF Glyph Data Table

Data Type	Syntax	Description and Comments

© ISO/IEC 2004 — All rights reserved

NSDOCID: <XP_____3001315A__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1802
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

SHORT	numContours	Number of contours in the glyph, also used as a flag.
	if (numContours < 0)	If numContours == -1, the glyph is composite glyph.
	{	
SHORT	xMin	Minimum X value for coordinate data
SHORT	yMin	Minimum Y value for coordinate data
SHORT	xMax	Maximum X value for coordinate data
SHORT	yMax	Maximum Y value for coordinate data
	/* Composite CTF Glyph Table */	
	} else {	
	if (numContours == 0x7FFF)	This is a flag, read new value and bounding box information below.
	{	
SHORT	numContours	Actual number of contours in the glyph.
SHORT	xMin	Minimum X value for coordinate data
SHORT	yMin	Minimum Y value for coordinate data
SHORT	xMax	Maximum X value for coordinate data
SHORT	yMax	Maximum Y value for coordinate data
	} else	Bounding box information omitted and should be computed from the glyph data coordinates.
	{}	
	/* Simple CTF Glyph Table */	
	}	

4.2.4.5 Simple CTF Glyph Data

A simple glyph defines all the contours and points that are used to create the glyph outline. Each point is presented in a (flag, xCoordinate, yCoordinate) triplet that is stored with the variable length encoding consuming 2 to 5 bytes per triplet. The flag value is always stored in the first byte of triplet and defines the format and properties of the point coordinates.

The most significant bit of the flag indicates whether the point is on- or off-curve point. The remaining 7 bits specify 128 possible combinations which define data format and value for X and Y coordinates of a point and specify the combination of the following properties:

- Total number of bytes used for this triplet (byte count);
- Number of bits used for describing X coordinate value (X bits);
- Number of bits for describing Y coordinate value (Y bits);
- An additional amount to be added to X coordinate value (delta X);
- An additional amount to be added to Y coordinate value (delta Y);
- The sign of X coordinate value (X sign);
- The sign of Y coordinate value (Y sign).

Each of the 128 combinations identify a unique set of these seven properties which have been assigned after careful consideration has been given to the typical statistical distribution of data found in TrueType files. As a result, the dataset size in CTF format is significantly reduced compared to the native TrueType format. However, the data is left byte aligned so that an Huffman entropy coder can be easily applied on the second stage.

Unlike the native TrueType formats that allows X and Y coordinates values be defined either in relative or absolute values, the CTF format always defines X and Y coordinates as relative values to the previous point. The first point is relative to (0,0). The information on data formats of xCoordinates[] and yCoordinates[] of glyph data can be found in the subclause 4.2.4.7 of this specification. This table describes simple glyph data in CTF format.

Table 7 – Simple CTF Glyph Table

Data Type	Syntax	Description and Comments
	/* Data */	

© ISO/IEC 2004 — All rights reserved

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1803
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

255USHORT	contourPoints[numContours]	Number of points in each contour, cumulative sum should be calculated to define total number of points.
BYTE	flags[numPoints]	Array of flags for each coordinate in glyph outline.
Variable length bit field	xCoordinates[numPoints]	Array of X coordinates for glyph outline points.
Variable length bit field	yCoordinates[numPoints]	Array of Y coordinates for glyph outline points.
	/* Glyph Program Table */	

4.2.4.6 Composite CTF Glyph Data

This table describes composite glyph data in CTF format.

Table 8 – Composite CTF Glyph Table

Data Type	Syntax	Description and Comments
	/* Data */	
	do {	/* for all glyph components */
USHORT	flags	Flags for each component in composite glyph.
	. . . . .	See "glyf" table specification for more information.
	} while (flags & MORE_COMPONENTS);	
	if (flags & WE_HAVE_INSTR) {	
	/* Glyph Program Table */	
	}	

The information on data formats of flags[], xCoordinates[] and yCoordinates[] of glyph data section can be found in the 'glyf' table of OpenType font format specification.

4.2.4.7 Triplet Encoding of X and Y Coordinates

The following table presents information on Triplet Encoding (flag[], xCoordinates[], yCoordinates[]) of simple glyph outline points.
Please note – "Byte Count" field reflects total size of the triplet, including 'flags'.

Table 9 – Coordinate Encoding Table

Index	Byte Count	X bits	Y bits	Delta X	Delta Y	X sign	Y sign
0	2	8	0	0	N/A	+	N/A
1				0		–	
2				256		+	
3				256		–	
4				512		+	
5				512		–	
6				768		+	
7				768		–	
8				1024		+	
9				1024		–	

© ISO/IEC 2004 – All rights reserved

ISDOCID: <KF_____3001818A__I_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1804
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Index	Byte Count	X bits	Y bits	Delta X	Delta Y	X sign	Y sign
10	2	0	8	N/A	0	N/A	+
11					0		-
12					256		+
13					256		-
14					512		+
15					512		-
16					768		+
17					768		-
18					1024		+
19					1024		-
20	2	4	4	1	1	+	+
21					1	+	-
22					1	-	+
23					1	-	-
24					17	+	+
25					17	+	-
26					17	-	+
27					17	-	-
28					33	+	+
29					33	+	-
30					33	-	+
31					33	-	-
32					49	+	+
33					49	+	-
34					49	-	+
35					49	-	-
36	2	4	4	17	1	+	+
37					1	+	-
38					1	-	+
39					1	-	-
40					17	+	+
41					17	+	-
42					17	-	+
43					17	-	-
44					33	+	+
45					33	+	-
46					33	-	+
47					33	-	-
48					49	+	+
49					49	+	-
50					49	-	+
51					49	-	-
52	2	4	4	33	1	+	+
53					1	+	-
54					1	-	+
55					1	-	-
56					17	+	+
57					17	+	-
58					17	-	+
59					17	-	-
60					33	+	+
61					33	+	-
62					33	-	+
63					33	-	-
64					49	+	+
65					49	+	-
66					49	-	+
67					49	-	-

© ISO/IEC 2004 — All rights reserved
BNSDOCID: <XP_____3001315GA__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1805
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Index	Byte Count	X bits	Y bits	Delta X	Delta Y	X sign	Y sign
68	2	4	4	49	1	+	+
69					1	+	-
70					1	-	+
71					1	-	-
72					17	+	+
73					17	+	-
74					17	-	+
75					17	-	-
76					33	+	+
77					33	+	-
78					33	-	+
79					33	-	-
80					49	+	+
81					49	+	-
82					49	-	+
83					49	-	-
84	3	8	8	1	1	+	+
85					1	+	-
86					1	-	+
87					1	-	-
88					257	+	+
89					257	+	-
90					257	-	+
91					257	-	-
92					513	+	+
93					513	+	-
94					513	-	+
95					513	-	-
96	3	8	8	257	1	+	+
97					1	+	-
98					1	-	+
99					1	-	-
100					257	+	+
101					257	+	-
102					257	-	+
103					257	-	-
104					513	+	+
105					513	+	-
106					513	-	+
107					513	-	-
108	3	8	8	513	1	+	+
109					1	+	-
110					1	-	+
111					1	-	-
112					257	+	+
113					257	+	-
114					257	-	+
115					257	-	-
116					513	+	+
117					513	+	-
118					513	-	+
119					513	-	-
120	4	12	12	0	0	+	+
121						+	-
122						-	+
123						-	-
124	5	16	16	0	0	+	+
125						+	-

© ISO/IEC 2004 — All rights reserved

«SDOCID: <XP_____3001B153A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1806
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Index	Byte Count	X bits	Y bits	Delta X	Delta Y	X sign	Y sign
126						-	+
127						-	-

The coordinate values of simple glyph outline points are calculated as follows:

```
Xcoord[i] = (SHORT)((Xsign)*(xCoordinate[i] + DeltaX[index]);
Ycoord[i] = (SHORT)((Ysign)*(yCoordinate[i] + DeltaY[index]);
```

5 Font Data Stream

5.1 Structure of the Font Data Stream

Font data streams are streams of streamType FontDataStream 0x0C (see ISO/IEC 14496-1, Table 7), identified by a specific objectTypeIndication 0x06 (see ISO/IEC 14496-1, Table 6). A font data stream consists of one or more access units with font data.

5.2 Access Unit Definition

5.2.1 Overview

A font data access unit conveys data for one single font, or a subset thereof. Two access unit formats are specified, discriminated by the DecoderSpecificInfo for the font data stream (see subclause 5.4). Only one of these access unit formats can be selected for a given font data stream.

The basic access unit format permits to convey font data without any header information. The DecoderSpecificInfo provides the identification of the font and, possibly, the subset thereof that is conveyed. The content of subsequent basic access units in such a font data stream shall replace the font or font subset data of previous basic access units.

The enhanced access unit format provides a self-contained format where an access unit header contains all necessary information about the font data conveyed in this access unit.

Access units in font data streams shall be labelled and time-stamped by suitable means. This shall be done via the related flags and the composition time stamps, respectively, in the SL packet header (see ISO/IEC 14496-1, 10.2.4). The composition time indicates the point in time at which a font data access unit becomes valid, i.e., when the embedded font information becomes available for font rendering. Decoding and composition time for a font data access unit shall always have the same value.

For all basic access units and those enhanced access units that have `fontSubsetExtensionFlag` set to '0' the `randomAccessPointFlag` in the SL packet header shall be set to '1' for this access unit. Otherwise, the `randomAccessPointFlag` shall be set to '0'.

5.2.2 Syntax

```
class BasicFontAccessUnit() {
   bit(8)  fontData[sizeOfInstance];
}

class EnhancedFontAccessUnit() {
   bit(7)  fontFormat;
   bit(1)  storeFont;
   bit(8)  fontNameLength;
   bit(8)  fontName[fontNameLength];
   bit(7)  fontSubsetID;
   bit(1)  fontSubsetExtensionFlag;
   bit(8)  fontSpecInfoLength;
   bit(8)  fontSpecInfo[fontSpecInfoLength];
   bit(8)  fontData[sizeOfInstance - fontNameLength - fontSpecInfoLength - 5]
}
```


© ISO/IEC 2004 — All rights reserved
BNSDOCID: <XP_____3001915BA__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1807
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

5.2.3 Semantics

`fontFormat` – indicates the format of the access unit as follows.

Table 10 – `fontFormat` values for EnhancedAccessUnit

`fontFormat`	Access unit content
0x00	Forbidden
0x01	OpenType with TrueType outlines (uncompressed)
0x02	OpenType with TrueType outlines compressed using the compression mechanism described in subclause 4.2
0x03	OpenType CFF
0x04 – 0x3F	ISO reserved
0x40 – 0x7F	User private

`storeFont` – if set to '1', the terminal shall store the font information contained within the access unit at least until the end of the current session. Otherwise, the font information becomes unavailable as soon as the font data stream becomes unavailable.

NOTE — Details of persistent storage are beyond the scope of this specification.

`fontNameLength` – specifies the length in characters of the `fontName` field.

`fontName` – a Unicode (ISO/IEC 10646-1) encoded string that indicates the name of the font.

`fontSubsetID` – specifies a numerical ID for a subset of the font conveyed in this access unit. If `fontSubsetID` equals 0x00, the access unit contains the complete font data. The value of 0x7F shall not be used.

`fontSubsetExtensionFlag` – if set to one indicates that the font data in this access unit is an extension of the previously sent font subset with the same `fontName` and `fontSubsetID`. Otherwise the font subset replaces any previous font subset with the same `fontName` and `fontSubsetID`.

`fontSpecInfoLength` - defines the length in bytes of the `fontSpecInfo` field.

`fontSpecInfo` – is an opaque container with information for a specific font handler.

`fontData` – contains the font data in the specified `fontFormat`.

5.3 Time Base for Font Data Streams

The time base associated with a font data stream shall be indicated by suitable means. This shall be done by the use of object clock reference time stamps in the SL packet headers (see ISO/IEC 14496-1, 10.2.4) for this stream or by indicating the elementary stream from which this font data stream inherits the time base (see ISO/IEC 14496-1, 10.2.3). All time stamps in the SL-packetized font data stream refer to this time base.

5.4 Font Data Decoder Configuration

5.4.1 Syntax

```
class FontDataDecoderConfiguration extends DecoderSpecificInfo  : bit(8)
tag=DecSpecificInfoTag {
   bit(7)  fontFormat;
   if (fontFormat != 0x00) {
      bit(1)  storeFont;
      bit(8)  fontNameLength;
      bit(8)  fontName[fontNameLength];
```

© ISO/IEC 2004 — All rights reserved

NBDCCIO: <XP_____30010158A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1808
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
        bit(7)  fontSubsetID;
        bit(1)  reserved = 1;
        bit(8)  fontSpecInfo[sizeOfInstance - fontNameLength - 4];
    }
}
```

5.4.2 Semantics

`fontFormat` – indicates the format of the access unit as follows.

Table 11 – `fontFormat` values for FontDataDecoderConfiguration

`fontFormat`	Access unit content
0x00	enhanced access unit format
0x01	OpenType with TrueType outlines (uncompressed)
0x02	OpenType with TrueType outlines compressed using the compression mechanism described in section 4.2
0x03	OpenType CFF
0x04 – 0x3F	ISO reserved
0x40 – 0x7F	User private

A `fontFormat` of 0x00 indicates the use of the enhanced access unit format for this stream. In this format, all information about the font data is conveyed within each access unit.

`storeFont` – if set to '1', the terminal shall store the font information contained within the access unit at least until the end of the current session. Otherwise, the font information becomes unavailable as soon as the font data stream becomes unavailable.

NOTE — Details of persistent storage are beyond the scope of this specification.

`fontNameLength` – specifies the length in characters of the `fontName` field.

`fontName` – a Unicode (ISO/IEC 10646-1) encoded string that indicates the name of the font.

`fontSubsetID` – specifies a numerical ID for a subset of the font conveyed in this access unit. If `fontSubsetID` equals 0x00, the access unit contains the complete font data. The value of 0x7F shall not be used.

`fontSpecInfoLength` - defines the length in bytes of the `fontSpecInfo` field.

`fontSpecInfo` – is an opaque container with information for a specific font handler.

5.5 Accessing the Font Data

A font presented in either `BasicFontAccessUnit` or `EnhancedFontAccessUnit` of a `FontDataStream` can be identified by a number of different attributes. One common identifier for fonts is the font name. However, a font can be presented either as a whole font, or as an arbitrary subset of an original font. In both cases the font name will be the same while the actual font data included in access unit may vary. In order to explicitly identify a font (or a subset thereof), the following syntax shall be used as the font name:

"`<fontName>[;OD:<odid>][;FSID:<fsid>]`", where :

- `<fontName>` is a Unicode encoded string that indicates the name of the font, as conveyed by `fontName` field in either `BasicFontAccessUnit` or `EnhancedFontAccessUnit` of associated font data stream;
- `<odid>` is the numeric value of the `objectDescriptorID` of the associated font data stream;
- `<fsid>` is the numeric value of the requested font (or a font subset) as conveyed by `fontSubsetID` within the `BasicFontAccessUnit` or `EnhancedFontAccessUnit` of associated font data stream.

For example, a fontName "`Times New Roman; OD:16; FSID:11`" would indicate that a font data presented in the font data stream associated with the `objectDescriptorID` = 16 and is conveyed by the `FontAccessUnit` with

© ISO/IEC 2004 — All rights reserved
<NSDOCID: <XP_____80013163A_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1809
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

`fontSubsetID` = 11 shall be used, or, if font data stream is not present, a font `"Times New Roman"` shall be used for text rendering, if available in the MPEG-4 terminal.

6 Text Profiles and Levels

6.1 Simple Text Profile and Levels

6.1.1 Application area

Proposed structure of the Simple profile is designed to satisfy the needs of the majority of languages that do not require support for complex script processing and advanced typographic features. It's also aimed to minimize the implementation complexity of the terminals, limiting font rendering capabilities to fonts with TrueType outlines only. These limitations allow using all existing TrueType fonts and OpenType fonts with TrueType outlines containing the set of required tables only and embedded bitmaps. The functionality of the Simple profile can be implemented utilizing previously defined **Text** and **FontStyle** BIFS nodes and does not require the use of new **XFontStyle** node (see ISO/IEC 14496-11 for details).

6.1.2 List of tools/functionalities

The Simple Text Profile represents the collection of tools that are required to guarantee the predictable and faithful reproduction of text objects utilizing TrueType fonts and simple language scripts (e.g. Western European and some Asian languages). The profile also provides tools for font compression and streaming.

6.1.3 Comparison with existing profiles

The Simple Text Profile is the lowest level profile defined for text support. It requires support for rendering of fonts with TrueType outlines, which is available on the majority of software platforms and operations systems.

6.1.4 Simple Text Profile @ Level 1 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines (only required tables defined in the OpenType specification and embedded bitmaps are supported).

6.1.5 Simple Text Profile @ Level 2 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines (only required tables defined in the OpenType specification and embedded bitmaps are supported);
- Streaming of uncompressed fonts only.

6.1.6 Simple Text Profile @ Level 3 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines (only required tables defined in the OpenType specification and embedded bitmaps are supported);
- Streaming of compressed and uncompressed fonts.

© ISO/IEC 2004 — All rights reserved

NSDOCID: <XP_____3001315A_!_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1810
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.2 Advanced Simple Text Profile and Levels

6.2.1 Application area

Proposed structure of the Advanced Simple Text Profile is designed to satisfy the needs of those languages that either require or would benefit from the support for advanced typographic features (e.g. stylistic alternatives and historical forms in Latin languages, support for both lowercase and small cap characters, support for ligatures and glyph substitution and positioning, etc.). It's also aimed to reduce the implementation complexity of the terminals, limiting font rendering capabilities to support TrueType outlines only. This limitation allows using all existing TrueType fonts and OpenType fonts with TrueType outlines.

6.2.2 List of tools/functionalities

The Advanced Simple Text Profile represents the collection of tools that are required to guarantee the predictable and faithful reproduction of text objects utilizing OpenType fonts with TrueType outlines and complex language scripts (e.g. Arabic and Indic scripts, as well as many other language scripts). The profile also provides tools for font compression and streaming.

6.2.3 Comparison with existing profiles

The Advanced Simple Profile extends the Simple Text Profile by adding the capability to support advanced text layout and advanced typographic features (e.g. bi-directional text rendering, ligatures, glyph substitution and positioning, etc.)

6.2.4 Advanced Simple Text Profile @ Level 1 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines;
- Support for OpenType Layout and advanced features.

6.2.5 Advanced Simple Text Profile @ Level 2 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines;
- Support for OpenType Layout and advanced features;
- Streaming of uncompressed fonts only.

6.2.6 Advanced Simple Text Profile @ Level 3 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType outlines;
- Support for OpenType Layout and advanced features;
- Streaming of compressed and uncompressed fonts.

© ISO/IEC 2004 — All rights reserved
BNSDOCID: <XP_____3001315SA_1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1811
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.3 Main Text Profile and Levels

6.3.1 Application area

Proposed structure of the Main Text Profile is designed to satisfy the needs of those languages that either require or would benefit from the support for advanced typographic features. It also allows using all existing TrueType fonts and OpenType fonts with both TrueType and CFF outlines.

6.3.2 List of tools/functionalities

The Main Text Profile represents the collection of tools that are required to guarantee the predictable and faithful reproduction of text objects utilizing OpenType fonts with either TrueType or CFF outlines and complex language scripts (e.g. Arabic and Indic scripts, as well as many other language scripts). The profile also provides tools for font compression and streaming.

6.3.3 Comparison with existing profiles

The Main Text Profile extends the Advanced Simple Text Profile by adding the capability to render OpenType fonts with CFF/Type 1 outlines with full support for advanced text layout and advanced typographic features (e.g. bi-directional text rendering, ligatures, glyph substitution and positioning, etc.)

6.3.4 Main Text Profile @ Level 1 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType and CFF outlines;
- Support for OpenType Layout and advanced features.

6.3.5 Main Text Profile @ Level 2 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType and CFF outlines;
- Support for OpenType Layout and advanced features;
- Streaming of uncompressed fonts only.

6.3.6 Main Text Profile @ Level 3 Definition

- Support for resident fonts;
- Rendering of OpenType fonts with TrueType and CFF outlines;
- Support for OpenType Layout and advanced features;
- Streaming of compressed and uncompressed fonts.

---

[i] OpenType is a registered trademark of Microsoft Corporation.
[ii] TrueType is a trademark of Apple Computer Incorporated.
[iii] MicroType is a registered trademark of Agfa Monotype Corporation.

© ISO/IEC 2004 — All rights reserved

«ISDOCID: <XP_____3061153A__1_>

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1812
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

SSME (Smooth Streaming Media Element) config.xml review (Smooth Streaming Client configuration file) - The Official Microsoft IIS Site

Home > Forums > IIS 7 > IIS7 - Media > SSME (Smooth Streaming Media Element) config.xml review (Smooth Streaming Client c

« Previous

Thread: SSME (Smooth Streaming Media Element) config.xml review (Smooth Streaming Client configuration file)

Last post 08-10-2011 9:35 PM by abarker. 4 replies.

Sort Posts:

Shortcuts
Search | Active Posts | Unanswered Posts | View all users

10-07-2010, 8:35 AM

SGamou

Joined on 10-07-2010, 8:32 AM
Posts 2

SSME (Smooth Streaming Media Element) config.xml review (Smooth Streaming Client configura

Reply Contact

SSME has this little "configPath" property that is a path to an XML file containing some impr configuration nodes.

If it is not set SSME tries to call "config.xml" from its root directory.

I could not find any documentation on this topic but it seams to be really powerful tool for S configuration.

The only place I could find info on one of the config parameters was the Ezequiel Jadib's Bl

http://blogs.southworks.net/ejadib/category/smooth-streaming-media-element/

that explains how to configure exact seek on SSME, i.e. a way to seek to exact frame in me closest keyframe.

Except this blog, no other information could be found yet as far as I concern

So it leaves no choice to the developer that want to use SSME as to investigate it himself.

It sound kind of interesting also. So I post below this XML file with my coments while the le comment investigation reliability would be marked as followed:

(H) for HIGH reliability. Sure enough and tested by me.

(M) for MEDIUM reliability. Not sure enough, needs more testing, hard to test.

(L) for LOW reliability. Only guessing or not tested directly

I should say that SSME heuristics are really complicated as you will see from the XML, so te components is not the easiest thing.
I'm waiting for a lot of inaccuracy places that you guys would find in my research.
Please feel free to notify me on those so I can retest and fix the post if needed or change th

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1814
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

parameter reliability.
This is the real goal of this thread as I see it untill microsoft would release official docs on tl

NOTE! you must remove all the comments before using the below XML configuration file, be

http://forums.iis.net/t/1172146.aspx

SSME (Smooth Streaming Media Element) config.xml review (Smooth Streaming Client configuration file) : The Official Microsoft IIS Site

not support comments inside a tag

```
<?xml version="1.0" encoding="utf-8" ?>
<LiveSmoothStreamingSettings>
  <BufferingEngineSettings

            //(H) Allows seek between keyframes,
            //i.e. to exact frame
            //(Originaly posted by Ezequiel Jadib in his Blog)
      ForceAccurateSeeks="true"

      LiveBackOffTimeIncrementMilliseconds = "250"
      LiveBackOffTimeMilliseconds = "250"
      LiveBufferTimeMilliseconds = "7000"

            //(H) try downloading the same chunk
            //until this number of 404 chunks is achieved
      Max404ErrorsPerChunk = "2"

            //(L) try downloading the same chunk
            // until this number of 412 chunks is achieved
      Max412ErrorsPerChunk = "8"

            //(L) if "ChunkDownloadTimeoutSeconds" is reached
            // while downloading chunk try this more times
      NumRetriesOnTimeout ="1"

            //(L) after "bad" buffering occurs and "RebufferSizeMultiplier "
            // and "RebufferSizeAddendMilliseconds " are applied,
            // time that takes for player to rebuffer should not exceed this number of r
      MaxAllowableTimeForRebuffersAfterSeekMilliseconds = "2000"

            //(L) the difference between quantity of video data
            // and audio data that were buffered must not exceed this number if ms
      MaxAudioVideoBufferDiscrepancyMilliseconds = "5000"


      MaxChunkDownloadRetries = "2"

            //(M) maximum Consecutive 404 error per stream until it finally fails
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1815
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
MaxConsecutiveChunkDownloadErrorsPerStream = "50"

        //(L) maximum Consecutive 412 error per stream until it finally fails
MaxConsecutiveChunkParseErrorsPerStream = "5"


MaxGetNextStreamSampleRetries = "3"

        //(H) maximum number of ms that could be buffered
       // when playing Live or VOD if no rebuffering occurred
MaxPlaybackBufferSizeMilliseconds = "30000"
```

http://forums.iis.net/t/1172146.aspx

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1816
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Transcoding Best Practices

From movideo

Below outlines movideo's recommendation for video ingestion into movideo.

movideo stores a Mezzanine File in the system for transcoding when required. This Mezzanine File is also used for thumbnail image generation.

Contents

- 1 Recommended Ingestion/Mezzanine File Format
- 2 Recommended Output Streaming Formats
  - 2.1 16:9 Format recommended for Adobe Flash Streaming
  - 2.2 4:3 Format recommended for Adobe Flash Streaming
  - 2.3 16:9 Format recommended for Apple iOS Streaming - iPad
  - 2.4 4:3 Format recommended for Apple iOS Streaming - iPad
  - 2.5 16:9 Format recommended for Apple iOS Streaming - iPhone
  - 2.6 4:3 Format recommended for Apple iOS Streaming - iPhone
  - 2.7 4:3 Format recommended for Android Phone Streaming
  - 2.8 16:9 Format recommended for Android Phone Streaming
  - 2.9 Notes

Recommended Ingestion/Mezzanine File Format

- **Container:** MP4
- **Video Type:** H.264
- **Size/Ratio:** 1920x1080 (16:9) 1440 x 1080 (4:3)
- **Video Bitrate:** 4096kbps (low motion e.g. news desk reporter) or 6144kbps (high motion e.g. sport)
- **Key Frame Interval:** minimum every 6 seconds
- **Audio Type:** AAC
- **Audio Sample Frequency:** 44.1kHz
- **Audio Bite Rate:** 192kbps

Recommended Output Streaming Formats

movideo uses Adobes Dynamic Streaming to deliver content to its Flash Player and Apples HTTP Live Streaming for deliver to iOS devices.

Both these technologies utilize adaptive bitrates by detecting a user's bandwidth and CPU capacity in real time and

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1817
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

adjusting the quality of a video stream accordingly. This use of adaptive bitrates requires multiple encoded files at various bitrates.

Below outlines the various encoding format recommendations for adaptive bitrate streaming from movideo.

16:9 Format recommended for Adobe Flash Streaming

Name	Width	Height	Audio Bitrate	Video Bitrate	Total Bitrate	Mime Type	Frame Rate	Key Frame Interval	H.264 Profile
1080p - 3Mbps video - 192Kbps Audio (16:9)	1920	1080	192	3,072	3,264	MP4	29.97	90	BASELINE
720p - 2Mbps video - 192Kbps Audio (16:9)	1280	720	192	2048	2,240	MP4	29.97	90	BASELINE
480p - 800Kbps Video - 128Kbps Audio (16:9)	854	480	128	819	947	MP4	29.97	90	BASELINE
360p - 500Kbps video - 128Kbps Audio (16:9)	640	360	128	512	640	MP4	29.97	90	BASELINE
240p - 250Kbps Video - 64Kbps Audio (16:9)	428	240	64	256	320	MP4	29.97	90	BASELINE

4:3 Format recommended for Adobe Flash Streaming

Name	Width	Height	Audio Bitrate	Video Bitrate	Total Bitrate	Mime Type	Frame Rate	Key Frame Interval	H.264 Profile
1080p - 3Mbps video - 192Kbps Audio (4:3)	1440	1080	192	3,072	3,264	MP4	29.97	90	BASELINE
720p - 2Mbps video - 192Kbps Audio (4:3)	960	720	192	2,048	2,240	MP4	29.97	90	BASELINE
480p - 800Kbps Video - 128Kbps Audio (4:3)	640	480	128	819	947	MP4	29.97	90	BASELINE
360p - 500Kbps video - 128Kbps Audio (4:3)	480	360	128	512	640	MP4	29.97	90	BASELINE
240p - 250Kbps Video - 64Kbps Audio (4:3)	320	240	64	256	320	MP4	29.97	90	BASELINE

16:9 Format recommended for Apple iOS Streaming - iPad

Name	Width	Height	Audio Bitrate	Video Bitrate	Total Bitrate	Mime Type	Frame Rate	Key Frame Interval	H.264 Profile
iPad WiFi - 16:9 1240kbps	640	360	40	1,200	1,240	MP4	29.97	90	MAIN
iPad WiFi - 16:9	640	360	40	800	840	MP4	29.97	90	MAIN

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1818
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

840kbps									
iPad WiFi - 16:9 640kbps	640	360	40	600	640	MP4	29.97	90	BASELINE
iPad Cell - 16:9 440kbps	400	224	40	400	440	MP4	29.97	90	BASELINE
iPad Cell Low - 16:9 240kbps	400	224	40	200	240	MP4	15	45	BASELINE
iPad Cell Low - 16:9 140kbps	400	224	40	110	150	MP4	10	30	BASELINE
iPad Cell Very Low - 16:9 64kbps	400	224	40	24	64	MP4	10	30	BASELINE

4:3 Format recommended for Apple iOS Streaming - iPad

Name	Width	Height	Audio Bitrate	Video Bitrate	Total Bitrate	Mime Type	Frame Rate	Key Frame Interval	H.264 Profile
iPad WiFi - 4:3 1240kbps	640	480	40	1,200	1,240	MP4	29.97	90	MAIN
iPad WiFi - 4:3 840kbps	640	480	40	800	840	MP4	29.97	90	MAIN
iPad WiFi - 4:3 640kbps	640	480	40	600	640	MP4	29.97	90	BASELINE
iPad Cell - 4:3 440kbps	400	300	40	400	440	MP4	29.97	90	BASELINE
iPad Cell Low - 4:3 240kbps	400	300	40	200	240	MP4	15	45	BASELINE
iPad Cell Low - 4:3 140kbps	400	300	40	110	150	MP4	10	30	BASELINE
iPad Cell Very Low - 4:3 64kbps	400	300	40	24	64	MP4	10	30	BASELINE

16:9 Format recommended for Apple iOS Streaming - iPhone

Name	Width	Height	Audio Bitrate	Video Bitrate	Total Bitrate	Mime Type	Frame Rate	Key Frame Interval	H.264 Profile
iPhone WiFi - 16:9 640kbps	400	224	40	600	640	MP4	29.97	90	BASELINE
iPhone WiFi - 16:9 440kbps	400	224	40	400	440	MP4	29.97	90	BASELINE
iPhone Cell Low - 16:9 240kbps	400	224	40	200	240	MP4	15	45	BASELINE

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1819
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

iPhone Cell Low - 16:9 140kbps	400	224	40	110	150	MP4	10	30	BASELINE
iPhone Cell Very Low - 16:9 64kbps	400	224	40	24	64	MP4	10	30	BASELINE

4:3 Format recommended for Apple iOS Streaming - iPhone

Name	Width	Height	Audio Bitrate	Video Bitrate	Total Bitrate	Mime Type	Frame Rate	Key Frame Interval	H.264 Profile
iPhone WiFi - 4:3 640kbps	400	300	40	600	640	MP4	29.97	90	BASELINE
iPhone WiFi - 4:3 440kbps	400	300	40	400	440	MP4	29.97	90	BASELINE
iPhone Cell Low - 4:3 240kbps	400	300	40	200	240	MP4	15	45	BASELINE
iPhone Cell Low - 4:3 140kbps	400	300	40	110	150	MP4	10	30	BASELINE
iPhone Cell Very Low - 4:3 64kbps	400	300	40	24	64	MP4	10	30	BASELINE

4:3 Format recommended for Android Phone Streaming

Name	Width	Height	Audio Bitrate	Video Bitrate	Total Bitrate	Mime Type	Frame Rate	Key Frame Interval	H.264 Profile
SD High Quality - 4:3 500kbps	512	384	128	372	500	MP4	29.97	90	BASELINE
SD Mid Quality - 4:3 300kbps	384	288	48	252	300	MP4	29.97	90	BASELINE
SD Low Quality - 4:3 80kbps	256	193	24	56	70	MP4	12	10	BASELINE

16:9 Format recommended for Android Phone Streaming

Name	Width	Height	Audio Bitrate	Video Bitrate	Total Bitrate	Mime Type	Frame Rate	Key Frame Interval	H.264 Profile
SD High Quality - 16:9 500kbps	512	288	128	372	500	MP4	29.97	90	BASELINE
SD Mid Quality - 16:9 300kbps	384	216	48	252	300	MP4	29.97	90	BASELINE
SD Low Quality - 16:9 80kbps	256	144	24	56	70	MP4	12	10	BASELINE

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1820
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Notes

- All Bitrates are shown as kilobits per second (kbit/s, kb/s, or kbps). kbps is a unit of data transfer rate equal to 1000 bits per second or 125 bytes per second.
- Key Frame Intervals are shown in number of frames. For more information see: Key Frame Interval (I-frame)
- The Reason for including a 64kbps encode with iOS is only if a submission to the Apple store is required. Any Application that streams video to a cellular network must use HTTP Live Streaming for any stream that is longer than 10 minutes. In addition, applications using HTTP Live Streaming over the cellular network must include a low quality stream of no more than 64kbps for your app to fall back to when network conditions demand it. For up to date information on submissions to the Apple store please see: Apples App Store Tips (http://developer.apple.com/news/ios/appstoretips/)
- For further reading on the subject of H.264 encoding please read the great paper by Adobe's Kush Amerasinghe H.264 For the rest of us (http://code.movideo.com/images/a/ac/H264_primer.pdf)
- For H.264 encoding for Adobe Flash delivery to Android devices please read the paper by Adobe's Maxim Levkov Mobile Encoding Guidelines for Android (http://code.movideo.com/images/e/ea/Mobile-encoding-android-v2_7.pdf)
- Googles Video Encoding Recommendations for Android (http://developer.android.com/guide/appendix/media-formats.html)
- Video Encoding Cookbook and Profile Guidelines for the Adobe Flash Platform (http://www.eventsadobe.com/cookbook/)

Retrieved from "http://code.movideo.com/Transcoding_Best_Practices"

- Privacy policy

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1821
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Using HTTP Live Streaming

Download the Tools

There are several tools available that can help you set up an HTTP Live Streaming service. The tools include a media stream segmenter, a media file segmenter, a stream validator, an id3 tag generator, and a variant playlist generator.

The tools are frequently updated, so you should download the current version of the HTTP Live Streaming Tools from the Apple Developer website. You can access them if you are a member of the iOS Developer Program. One way to navigate to the tools is to log onto developer.apple.com, then use the search feature.

Media Stream Segmenter

The `mediastreamsegmenter` command-line tool takes an MPEG-2 transport stream as an input and produces a series of equal-length files from it, suitable for use in HTTP Live Streaming. It can also generate index files (also known as playlists), encrypt the media, produce encryption keys, optimize the files by reducing overhead, and create the necessary files for automatically generating multiple stream alternates. For details, verify you have installed the tools, and type `man mediastreamsegmenter` from the terminal window.

Usage example: `mediastreamsegmenter -s 3 -D -f /Library/WebServer/Documents/stream 239.4.1.5:20103`

The usage example captures a live stream from the network at address `239.4.1.5:20103` and creates media segment files and index files from it. The index files contain a list of the current three media segment files (`-s 3`). The media segment files are deleted after use (`-D`). The index files and media segment files are stored in the directory `/Library/WebServer/Documents/stream`.

Media File Segmenter

The `mediafilesegmenter` command-line tool takes an encoded media file as an input, wraps it in an MPEG-2 transport stream, and produces a series of equal-length files from it, suitable for use in HTTP Live Streaming. The media file segmenter can also produce index files (playlists) and decryption keys. The file segmenter behaves very much like the stream segmenter, but it works on existing files instead of streams coming from an encoder. For details, type `man mediafilesegmenter` from the terminal window.

Media Stream Validator

The `mediastreamvalidator` command-line tool examines the index files, stream alternates, and media segment files on a server and tests to determine whether they will work with HTTP Live Streaming clients. For details, type `man mediastreamvalidator` from the terminal window.

Variant Playlist Creator

The `variantplaylistcreator` command-line tool creates a master index file, or playlist, listing

the index files for alternate streams at different bit rates, using the output of the `mediafilesegmenter`. The `mediafilesegmenter` must be invoked with the `-generate-variant-playlist` argument to produce the required output for the variant playlist creator. For details, type `man variantplaylistcreator` from the terminal window.

Metadata Tag Generator

The `id3taggenerator` command-line tool generates ID3 metadata tags. These tags can either be written to a file or inserted into outgoing stream segments. For details, see "Adding Timed Metadata."

Session Types

The HTTP Live Streaming protocol supports two types of sessions: events (live broadcasts) and video on demand (VOD).

VOD Sessions

For VOD sessions, media files are available representing the entire duration of the presentation. The index file is static and contains a complete list of all files created since the beginning of the presentation. This kind of session allows the client full access to the entire program.

VOD can also be used to deliver "canned" media. HTTP Live Streaming offers advantages over progressive download for VOD, such as support for media encryption and dynamic switching between streams of different data rates in response to changing connection speeds. (QuickTime also supports multiple-data-rate movies using progressive download, but QuickTime movies do not support dynamically switching between data rates in mid-movie.)

Live Sessions

Live sessions (events) can be presented as a complete record of an event, or as a sliding window with a limited time range the user can seek within.

For live sessions, as new media files are created and made available, the index file is updated. The new index file lists the new media files. Older media files can be removed from the index and discarded, presenting a moving window into a continuous stream—this type of session is suitable for continuous broadcasts. Alternatively, the index can simply add new media files to the existing list—this type of session can be easily converted to VOD after the event completes.

It is possible to create a live broadcast of an event that is instantly available for video on demand. To convert a live broadcast to VOD, do not remove the old media files from the server or delete their URLs from the index file; instead, add an `#EXT-X-ENDLIST` tag to the index when the event ends. This allows clients to join the broadcast late and still see the entire event. It also allows an event to be archived for rebroadcast with no additional time or effort.

If your playlist contains an `EXT-X-PLAYLIST-TYPE` tag, you should also change the value from `EVENT` to `VOD`.

Content Protection

Media files containing stream segments may be individually encrypted. When encryption is

employed, references to the corresponding key files appear in the index file so that the client can retrieve the keys for decryption.

When a key file is listed in the index file, the key file contains a cipher key that must be used to decrypt subsequent media files listed in the index file. Currently HTTP Live Streaming supports AES-128 encryption using 16-octet keys. The format of the key file is a packed array of these 16 octets in binary format.

The media stream segmenter available from Apple provides encryption and supports three modes for configuring encryption.

The first mode allows you to specify a path to an existing key file on disk. In this mode the segmenter inserts the URL of the existing key file in the index file. It encrypts all media files using this key.

The second mode instructs the segmenter to generate a random key file, save it in a specified location, and reference it in the index file. All media files are encrypted using this randomly generated key.

The third mode instructs the segmenter to generate a new random key file every *n* media segments, save it in a specified location, and reference it in the index file. This mode is referred to as key rotation. Each group of *n* files is encrypted using a different key.

> **Note**: All media files may be encrypted using the same key, or new keys may be required at intervals. The theoretical limit is one key per media file, but because each media key adds a file request and transfer to the overhead for presenting the subsequent media segments, changing to a new key periodically is less likely to impact system performance than changing keys for each segment.

You can serve key files using either HTTP or HTTPS. You may also choose to protect the delivery of the key files using your own session-based authentication scheme. For details, see "Serving Key Files Securely Over HTTPS."

Key files require an initialization vector (IV) to decode encrypted media. The IVs can be changed periodically, just as the keys can.

Caching and Delivery Protocols

HTTPS is commonly used to deliver key files. It may also be used to deliver the media segment files and index files, but this is not recommended when scalability is important, since HTTPS requests often bypass web server caches, causing all content requests to be routed through your server and defeating the purpose of edge network distribution systems.

For this very reason, however, it is important to make sure that any content delivery network you use understands that the `.M3U8` index files are not to be cached for longer than one media segment duration for live broadcasts, where the index file is changing dynamically.

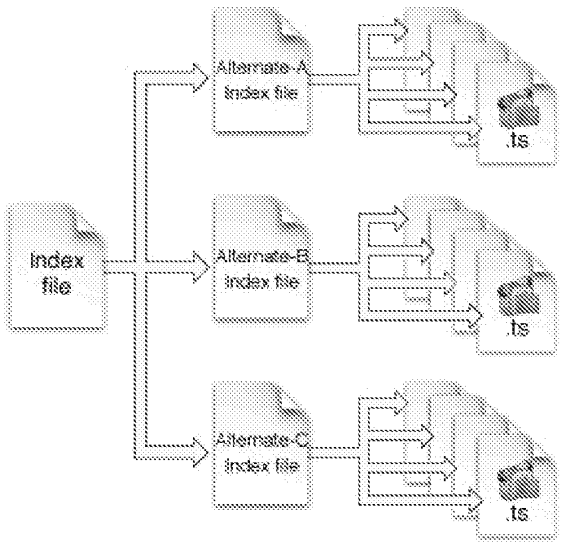
Stream Alternates

A master index file may reference alternate streams of content. References can be used to support delivery of multiple streams of the same content with varying quality levels for different

bandwidths or devices. HTTP Live Streaming supports switching between streams dynamically if the available bandwidth changes. The client software uses heuristics to determine appropriate times to switch between the alternates. Currently, these heuristics are based on recent trends in measured network throughput.

The master index file points to alternate streams of media by including a specially tagged list of other index files, as illustrated in Figure 2-1

Figure 2-1 Stream alternates



Both the master index file and the alternate index files are in `.M3U8` playlist format. The master index file is downloaded only once, but for live broadcasts the alternate index files are reloaded periodically. The first alternate listed in the master index file is the first stream used—after that, the client chooses among the alternates by available bandwidth.

Note that the client may choose to change to an alternate stream at any time, such as when a mobile device enters or leaves a WiFi hotspot. All alternates should use identical audio to allow smooth transitions among streams.

You can create a set of stream alternates by using the `variantplaylistcreator` tool and specifying the `-generate-variant-playlist` option for either the `mediafilesegmenter` tool or the `mediastreamsegmenter` tool (see "Download the Tools" for details).

When using stream alternates, it is important to bear the following considerations in mind:

- The first entry in the variant playlist is played when a user joins the stream and is used as part of a test to determine which stream is most appropriate. The order of the other entries is irrelevant.
- Where possible, encode enough variants to provide the best quality stream across a wide range of connection speeds. For example, encode variants at 150 kbps, 350 kbps, 550 kbps, 900 kbps, 1500 kbps.
- When possible, use relative path names in variant playlists and in the individual `.M3U8` playlist files
- The video aspect ratio on alternate streams *must* be exactly the same, but alternates can have different pixel dimensions, as long as they have the same aspect ratio. For example, two stream alternates with the same 4:3 aspect ratio could have dimensions of 400 x 300 and 800 x 600.

- The `RESOLUTION` field in the `EXT-X-STREAM-INF` should be included to help the client choose an appropriate stream.
- If you are an iOS app developer, you can query the user's device to determine whether the initial connection is cellular or WiFi and choose an appropriate master index file.

  To ensure the user has a good experience when the stream is first played, regardless of the initial network connection, you should have more than one master index file consisting of the same alternate index files but with a different first stream.

  A 150k stream for the cellular variant playlist is recommended.

  A 240k or 440k stream for the Wi-Fi variant playlist is recommended.

  Note: For details on how to query an iOS-based device for its network connection type, see the following sample code: *Reachability*.

- When you specify the bitrates for stream variants, it is important that the `BANDWIDTH` attribute closely match the actual bandwidth required by a given stream. If the actual bandwidth requirement is substantially different than the `BANDWIDTH` attribute, automatic switching of streams may not operate smoothly or even correctly.

Video Over Cellular Networks

When you send video to a mobile device such as iPhone or iPad, the client's Internet connection may move to or from a cellular network at any time.

HTTP Live Streaming allows the client to choose among stream alternates dynamically as the network bandwidth changes, providing the best stream as the device moves between cellular and WiFi connections, for example, or between 3G and EDGE connections. This is a significant advantage over progressive download.

It is strongly recommended that you use HTTP Live Streaming to deliver video to all cellular-capable devices, even for video on demand, so that your viewers have the best experience possible under changing conditions.

In addition, you should provide cellular-capable clients an alternate stream at 64 Kbps or less for slower data connections. If you cannot provide video of acceptable quality at 64 Kbps or lower, you should provide an audio-only stream, or audio with a still image.

A good choice for pixel dimensions when targeting cellular network connections is 400 x 224 for 16:9 content and 400 x 300 for 4:3 content (see "Preparing Media for Delivery to iOS-Based Devices").

Requirements for Apps

Warning: iOS apps submitted for distribution in the App Store must conform to these requirements.

If your app delivers video over cellular networks, and the video exceeds either 10 minutes duration

or 5 MB of data in a five minute period, you are required to use HTTP Live Streaming. (Progressive download may be used for smaller clips.)

If your app uses HTTP Live Streaming over cellular networks, you are required to provide at least one stream at 64 Kbps or lower bandwidth (the low-bandwidth stream may be audio-only or audio with a still image).

These requirements apply to iOS apps submitted for distribution in the App Store for use on Apple products. Non-compliant apps may be rejected or removed, at the discretion of Apple.

Redundant Streams

If your playlist contains alternate streams, they can not only operate as bandwidth or device alternates, but as failure fallbacks. Starting with iOS 3.1, if the client is unable to reload the index file for a stream (due to a 404 error, for example), the client attempts to switch to an alternate stream.

In the event of an index load failure on one stream, the client chooses the highest bandwidth alternate stream that the network connection supports. If there are multiple alternates at the same bandwidth, the client chooses among them in the order listed in the playlist.

You can use this feature to provide redundant streams that will allow media to reach clients even in the event of severe local failures, such as a server crashing or a content distributor node going down.

To support redundant streams, create a stream—or multiple alternate bandwidth streams—and generate a playlist file as you normally would. Then create a parallel stream, or set of streams, on a separate server or content distribution service. Add the list of backup streams to the playlist file, so that the backup stream at each bandwidth is listed after the primary stream. For example, if the primary stream comes from server ALPHA, and the backup stream is on server BETA, your playlist file might look something like this:

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=200000, RESOLUTION=720x480
http://ALPHA.mycompany.com/lo/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=200000, RESOLUTION=720x480
http://BETA.mycompany.com/lo/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=500000, RESOLUTION=1920x1080
http://ALPHA.mycompany.com/md/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=500000, RESOLUTION=1920x1080
http://BETA.mycompany.com/md/prog_index.m3u8
```

Note that the backup streams are intermixed with the primary streams in the playlist, with the backup at each bandwidth listed after the primary for that bandwidth.

You are not limited to a single backup stream set. In the example above, ALPHA and BETA could be followed by GAMMA, for instance. Similarly, you need not provide a complete parallel set of streams. You could provide a single low-bandwidth stream on a backup server, for example.

Adding Timed Metadata

You can add various kinds of metadata to media stream segments. For example, you can add the album art, artist's name, and song title to an audio stream. As another example, you could add the current batter's name and statistics to video of a baseball game.

If an audio-only stream includes an image as metadata, the Apple client software automatically displays it. Currently, the only metadata that is automatically displayed by the Apple-supplied client software is a still image accompanying an audio-only stream.

If you are writing your own client software, however, using either `MPMoviePlayerController` or `AVPlayerItem`, you can access streamed metadata using the `timedMetaData` property.

If you are writing your own segmenter, you can read about the stream format for timed metadata information in *Timed Metadata for HTTP Live Streaming*.

If you are using Apple's tools, you can add timed metadata by specifying a metadata file in the `-F` command line option to either the stream segmenter or the file segmenter. The specified metadata source can be a file in ID3 format or an image file (JPEG or PNG). Metadata specified this way is automatically inserted into every media segment.

This is called timed metadata because it is inserted into a media stream at a given time offset. Timed metadata can optionally be inserted into all segments after a given time.

To add timed metadata to a live stream, use the `id3taggenerator` tool, with its output set to the stream segmenter. The tool generates ID3 metadata and passes it the stream segmenter for inclusion in the outbound stream.

The tag generator can be run from a shell script, for example, to insert metadata at the desired time, or at desired intervals. New timed metadata automatically replaces any existing metadata.

Once metadata has been inserted into a media segment, it is persistent. If a live broadcast is re-purposed as video on demand, for example, it retains any metadata inserted during the original broadcast.

Adding timed metadata to a stream created using the file segmenter is slightly more complicated.

1. First, generate the metadata samples. You can generate ID3 metadata using the `id3taggenerator` command-line tool, with the output set to file.
2. Next, create a **metadata macro file**—a text file in which each line contains the time to insert the metadata, the type of metadata, and the path and filename of a metadata file.

   For example, the following metadata macro file would insert a picture at 1.2 seconds into the stream, then an ID3 tag at 10 seconds:

   ```
   1.2 picture /meta/images/picture.jpg

   10 id3 /meta/id3/title.id3
   ```

3. Finally, specify the metadata macro file by name when you invoke the media file segmenter, using the `-M` command line option.

For additional details, see the `man` pages for `mediastreamsegmenter`, `mediafilesegmenter`, and `id3taggenerator`.

Adding Closed Captions

HTTP Live Streaming supports closed captions within streams. If you are using the stream segmenter, you need to add CEA-608 closed captions to the MPEG-2 transport stream (in the main video elementary stream) as specified in ATSC A/72. If you are using the file segmenter, you should encapsulate your media in a QuickTime movie file and add a closed captions track (`'clcp'`). If you are writing an app, the AVFoundation framework supports playback of closed captions.

Live streaming also supports multiple subtitle and closed caption tracks in Web Video Text Tracks (WebVTT) format. The sample in Listing 2-1 shows two closed captions tracks in a master playlist.

Listing 2-1 Master playlist with multiple closed captions tracks

```
#EXTM3U


#EXT-X-MEDIA:TYPE=CLOSED-CAPTIONS,GROUP-
ID="cc",NAME="CC1",LANGUAGE="en",DEFAULT=YES,AUTOSELECT=YES,INSTREAM-ID="CC1"

#EXT-X-MEDIA:TYPE=CLOSED-CAPTIONS,GROUP-
ID="cc",NAME="CC2",LANGUAGE="sp",AUTOSELECT=YES,INSTREAM-ID="CC2"


#EXT-X-STREAM-INF:BANDWIDTH=1000000,SUBTITLES="subs",CLOSED-CAPTIONS="cc"

x.m3u8
```

In the encoding process, the WebVTT files are broken into segments just as audio and video media. The resulting media playlist includes segment durations to sync text with the correct point in the associated video.

Advanced features of live streaming subtitles and closed captions include semantic metadata, CSS styling, and simple animation.

For more information on implementation of WebVTT is available in *WWDC 2012: What's New in HTTP Live Streaming* and WebVTT: The Web Video Text Tracks Format specification.

Preparing Media for Delivery to iOS-Based Devices

The recommended encoder settings for streams used with iOS-based devices are shown in the following four tables. For live streams, these settings should be available from your hardware or software encoder. If you are re-encoding from a master file for video on demand, you can use a video editing tool such as Compressor.

File format for the file segmenter can be a QuickTime movie, MPEG-4 video, or MP3 audio, using the specified encoding.

Stream format for the stream segmenter must be MPEG elementary audio and video streams, wrapped in an MPEG-2 transport stream, and using the following encoding. The "Audio Technologies" in *iOS Technology Overview* and "Video Technologies" in *iOS Technology Overview* list supported compression formats.

- Encode video using H.264 compression
  - H.264 Baseline 3.0: All devices
  - H.264 Baseline 3.1: iPhone 3G and later, and iPod touch 2nd generation and later.
  - H.264 Main profile 3.1: iPad (all versions), Apple TV 2 and later, and iPhone 4 and later.
  - H.264 Main Profile 4.0: Apple TV 3 and later, iPad 2 and later, and iPhone 4S and later

    - H.264 High Profile 4.0: Apple TV 3 and later, iPad 2 and later, and iPhone 4S and later.
    - H.264 High Profile 4.1: iPad 2 and later and iPhone 4S and later.

- A frame rate of 10 fps is recommended for video streams under 200 kbps. For video streams under 300 kbps, a frame rate of 12 to 15 fps is recommended. For all other streams, a frame rate of 29.97 is recommended.
- Encode audio as either of the following:
    - HE-AAC or AAC-LC, stereo
    - MP3 (MPEG-1 Audio Layer 3), stereo
- Optionally include an AC-3 audio track for Apple TV (or AirPlay to an Apple TV) when used with a surround sound receiver or TV that supports AC-3 input.
- A minimum audio sample rate of 22.05 kHz and audio bit rate of 40 kbps is recommended in all cases, and higher sampling rates and bit rates are strongly encouraged when streaming over Wi-Fi.

Table 2-1 Baseline profile encoder settings, 16:9 aspect ratio

Connection	Dimensions	Total bit rate	Video bit rate	Keyframes
Cellular	400 x 224	64 kbps	audio only	none
Cellular	400 x 224	150 kbps	110 kbps	30
Cellular	400 x 224	240 kbps	200 kbps	45
Cellular	400 x 224	440 kbps	400 kbps	90
WiFi	640 x 360	640 kbps	600 kbps	90

Table 2-2 Baseline profile encoder settings, 4:3 aspect ratio

Connection	Dimensions	Total bit rate	Video bit rate	Keyframes
Cellular	400 x 300	64 kbps	audio only	none
Cellular	400 x 300	150 kbps	110 kbps	30
Cellular	400 x 300	240 kbps	200 kbps	45
Cellular	400 x 300	440 kbps	400 kbps	90
WiFi	640 x 480	640 kbps	600 kbps	90

Table 2-3 Additional main profile encoder settings, 16:9 aspect ratio

Connection	Dimensions	Total bit rate	Video bit rate	Keyframes
WiFi	640 x 360	1240 kbps	1200 kbps	90
WiFi	960 x 540	1840 kbps	1800 kbps	90
WiFi	1280 x 720	2540 kbps	1500 kbps	90
WiFi	1280 x 720	4540 kbps	4500 kbps	90

Table 2-4 Additional main profile encoder settings, 4:3 aspect ratio

Connection	Dimensions	Total bit rate	Video bit rate	Keyframes
WiFi	640 x 480	1240 kbps	1200 kbps	90
WiFi	960 x 720	1840 kbps	1800 kbps	90
WiFi	960 x 720	2540 kbps	2500 kbps	90
WiFi	1280 x 960	4540 kbps	4500 kbps	90

Table 2-5 Additional high profile encoder settings, 16:9 aspect ratio

Connection	Dimensions	Total bit rate	Video bit rate	Keyframes
WiFi	1920 x 1080	12000 kbps	11000 kbps	90
WiFi	1920 x 1080	25000 kbps	24000 kbps	90
WiFi	1920 x 1080	40000 kbps	39000 kbps	90

Sample Streams

There are a series of HTTP streams available for testing on Apple's developer site. These examples show proper formatting of HTML to embed streams, `.M3U8` files to index the streams, and `.ts` media segment files. The streams can be accessed from the HTTP Live Streaming Resources.

Copyright © 2014 Apple Inc. All Rights Reserved. Terms of Use | Privacy Policy | Updated: 2014-02-11

Video Manager and Video Title Set IFO file headers

Offset	size	VMG IFO Contents
0000	12	DVDVIDEO-VMG
000C	4	last sector of VMG set (last sector of BUP)
001C	4	last sector of IFO
0020	2	version number
0022	4	VMG category byte1=prohibited region mask
0026	2	number of volumes
0028	2	volume number
002A	1	side ID
003E	2	number of title sets
0040	32	Provider ID
0060	8	VMG POS
0080	4	end byte address of VMGI_MAT
0084	4	start address of FP_PGC (First Play program chain)
00C0	4	start sector of Menu VOB
00C4	4	sector pointer to TT_SRPT (table of titles)
00C8	4	sector pointer to VMGM_PGCI_UT (Menu Program Chain table)
00CC	4	sector pointer to VMG_PTL_MAIT (Parental Management masks)
00D0	4	sector pointer to VMG_VTS_ATRT (copies of VTS audio/sub-picture attributes)
00D4	4	sector pointer to VMG_TXTDT_MG (text data)

File Structures		
VMG Files		
00000000	**Video_ts.ifo**	
(001C)->		
(00C0)->	**Video_ts.vob**	
	Video_ts.bup	
(000C)->		
VTS Files		
Vts_*xx*_0.ifo	00000000	
	<-(001C)	
Vts_*xx*_0.vob	<-(00C0)	
Vts_*xx*_1.vob : **Vts_*xx*_9.vob**	<-(00C4)	
Vts_*xx*_0.bup		
	<-(000C)	

VTS IFO Contents
DVDVIDEO-VTS
last sector of title set (last sector of BUP)
last sector of IFO
version number
VTS category 0=unspecified, 1=Karaoke
end byte address of VTS_MAT
start sector of Menu Vob
start sector of Title Vob
sector pointer to VTS_PTT_SRPT (table of Titles and Chapters)
sector pointer to VTS_PGCI (Title Program Chain table)
sector pointer to VTSM_PGCI_UT (Menu Program Chain table)
sector pointer to VTS_TMAPTI (time map)

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1832
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

00D8	4	sector pointer to VMGM_C_ADT (menu cell address table)	sector pointer to VTSM_C_ADT (menu cell address table)
00DC	4	sector pointer to VMGM_VOBU_ADMAP (menu VOBU address map)	sector pointer to VTSM_VOBU_ADMAP (menu VOBU address map)
00E0	4		sector pointer to VTS_C_ADT (title set cell address table)
00E4	4		sector pointer to VTS_VOBU_ADMAP (title set VOBU address map)
0100	2	video attributes of VMGM_VOBS	video attributes of VTSM_VOBS
0102	2	number of audio streams in VMGM_VOBS	number of audio streams in VTSM_VOBS
0104	8*8	audio attributes of VMGM_VOBS	audio attributes of VTSM_VOBS
0144	16		
0154	2	number of subpicture streams in VMGM_VOBS (0 or 1)	number of subpicture streams in VTSM_VOBS (0 or 1)
0156	6	subpicture attributes of VMGM_VOBS	subpicture attributes of VTSM_VOBS
015A	164	reserved	reserved
0200	2		video attributes of VTS_VOBS
0202	2		number of audio streams in VTS_VOBS
0204	8*8		audio attributes of VTS_VOBS
0144	16		
0254	2		number of subpicture streams in VTS_VOBS
0256	32*6		subpicture attributes of VTS_VOBS
0316	2		
0318	8*24		multichannel extension
03D8			

Version Number

byte	7	6	5	4	3	2	1	0
0	reserved, should be 0							

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1833
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

1	Major version number	Minor version number

Video Attributes

byte	7	6	5	4	3	2	1	0
0	Coding mode 0 = Mpeg-1 1 = Mpeg-2		Standard 0 = NTSC 1 = PAL		Aspect 0 = 4:3 1,2 = reserved 3 = 16:9		Automatic Pan/Scan 1 = disallowed	Automatic Letterbox 1 = disallowed
1	CC for line 21 field 1 in GOP (NTSC only)	CC for line 21 field 2 in GOP (NTSC only)	Resolution NTSC (PAL) 0 = 720x480 (720x576) 1 = 704x480 (704x576) 2 = 352x480 (352x576) 3 = 352x240 (352x288)		0 = Not letterboxed 1 = Letterboxed		Unknown	If Standard = PAL 0 = camera 1 = film

Audio Attributes

byte	7	6	5	4	3	2	1	0
0	Coding mode 0 AC3 1 ??? 2 Mpeg-1 3 Mpeg-2ext 4 LPCM 5 ??? 6 DTS 7 ???			Multichannel extension present	Language type 0 = unspecified 1 = per bytes 2&3		Application mode 0 = unspecified 1 = karaoke 2 = surround	
1	Quantization/DRC If Coding mode=2,3 0 = No DRC 1 = DRC If Coding mode=4 0 = 16bps 1 = 20bps 2 = 24bps, (DRC = dynamic range control)			Sample rate 0 = 48Kbps			channels-1	
2	1st character of language code							
3	2nd character of language code							
4	reserved for language code extension							

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1834
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

5	code extension, 0 = unspecified, 1 = normal, 2 = for visually impaired, 3 = director's comments, 4 = alternate director's comments - see SPRM #17				
6					
7	Application information karaoke mode				
	0	Channel assignments 0 = 1+1 (not valid), 1 = 1/0 (not valid), 2 = 2/0 L,R , 3 = 3/0 L,M,R , 4 = 2/1 L,R,V1 , 5 = 3/1 L,M,R,V1 , 6 = 2/2 L,R,V1,V2 , 7 = 3/2 L,M,R,V1,V2	Karaoke version	MC intro present	0=solo, 1=duet
	surround mode				
	reserved		1 = suitable for Dolby surround decoding	reserved	

MultiChannel Extension - Karaoke mode

byte (hex)	7	6	5	4	3	2	1	0
00	reserved							ACH0 Guide Melody exists
01	reserved							ACH1 Guide Melody exists
02	reserved				ACH2 Guide Vocal 1 exists	ACH2 Guide Vocal 2 exists	ACH2 Guide Melody 1 exists	ACH2 Guide Melody 2 exists
03	reserved				ACH3 Guide Vocal 1 exists	ACH3 Guide Vocal 2 exists	ACH3 Guide Melody A exists	ACH3 Sound Effect A exists
04	reserved				ACH4 Guide Vocal 1 exists	ACH4 Guide Vocal 2 exists	ACH4 Guide Melody B exists	ACH4 Sound Effect B exists
05-17								

Subpicture Attributes

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1835
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

byte	7	6	5	4	3	2	1	0
0	Coding mode 0 = 2-bit rle						Language type 0 = unspecified 1 = per bytes 2&3	
1								
2	1st character of language code							
3	2nd character of language code							
4	reserved for language code extension							
5	code extension - see SPRM #19							

VMGM_C_ADT
VTSM_C_ADT
VTS_C_ADT

This table contains pointers to all the cells in the menu or title set.

offset	0	1	2	3	4	5	6	7
0	number of VOB IDs		reserved		end address (last byte of last entry)			
8...	12-byte entries							

each entry

Offset	size	Contents
0	2	VOBidn
2	1	CELLidn
3	1	
4	4	starting sector within VOB
8	4	ending sector within VOB

VMGM_VOBU_ADMAP
VTSM_VOBU_ADMAP
VTS_VOBU_ADMAP

This table contains pointers to all the VOBUs in the menu or title set.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1836
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

offset	0	1	2	3
0	end address (last byte of last entry)			
4	starting sector within VOB of first VOBU			
8 ...	possible additional entries			


DVD-Video home

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1837
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

WHAT IS
Blu-ray
DVD
VCD
Glossary
HOW TO
All guides
Articles
Author
Capture
Convert
DVD Backup
Edit
Play
LISTS
Capture Cards
DVD Media
DVD Players
DVD Hacks
DVD Recorders
DVD Writers
Tools
OTHER
Forum
F.A.Q
Search
About
Advertise
Contact us
Affiliates
DVD-R Blank Media
DVD Duplicators
DVDFab

What is DVD?

Search... Search



DVD Sizes, DVD-5, DVD-9
DVD Write and read speeds
Download DVD Samples
DVD-Video Technical Info
DVD File/Folder structure
Video file comparison
More info

DVD stands for Digital Versatile/Video Disc, **DVDR** stands for DVD Recordable and **DVDRW** for DVD ReWriteable. If you're familiar with regular audio/music CDs or regular DVD-Video discs, then you will know what a recordable DVD looks like. A recordable DVD stores up to 2 hours of very good quality DVD-Video, including several audio tracks in formats like stereo, Dolby Digital or DTS and also advanced menu systems, subtitles and still pictures that can be played by many standalone DVD Players and most computer DVD-ROMs. If you choose to lower the video quality it is possible to store several hours video on a recordable DVD using low bitrates and low resolution with video quality more like VHS, SVHS, SVCD, CVD or VCD. It is also possible to have up to 4.37* GB ordinary data or mix DVD-Video and data on a recordable DVD that can be played by most computer DVD-ROMs.

There are three competing DVD Recording standards, **DVD-R/DVD-RW** and **DVD+R/DVD+RW** have pretty similiar features and are compatible with many standalone DVD Players and most DVD-ROMs while **DVD-RAM** has less DVD Player and DVD-ROM compatibility but better recording features.

DVD-R and **DVD-RW**
DVD-R was the first DVD recording format released that was compatible with standalone DVD Players.
DVD-R is a non-rewriteable format and it is compatible with about 93% of all DVD Players and most DVD-ROMs.
DVD-RW is a rewriteable format and it is compatible with about 80% of all DVD Players and most DVD-ROMs.
DVD-R and DVD-RW supports single side 4.37 computer GB* DVDs(called DVD-5) and double sided 8.75 computer GB* DVDs(called DVD-10).
These formats are supported by DVDForum.

DVD+R and **DVD+RW**
DVD+R is a non-rewritable format and it is compatible with about 89% of all DVD Players and most DVD-ROMs.
DVD+RW is a rewritable format and is compatible with about 79% of all DVD Players and most DVD-ROMs.
DVD+R and DVD+RW supports single side 4.37 computer GB* DVDs(called DVD-5) and double side 8.75 computer GB* DVDs(called DVD-10).
These formats are supported by the DVD+RW Alliance.

DVD+R DL
DVD+R DL or called DVD+R9 is a Dual Layer writeable DVD+R. The dual layered discs can hold 7.95 computer GB* (called DVD-9) and

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1838
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Blank dvd Media
Free Codecs
CD Freaks

Site layout:
Default
Blue
Center
Change
Cookies must be enabled.

Printer Friendly

©1999-2010 videohelp.com dvdrhelp.com vcdhelp.com

Contact Us
Privacy Policy

dual layered double sides 15.9* computer GB (called dvd-18).

DVD-R DL
DVD-R DL or called DVD-R9 is a Dual Layer writeable DVD-R. The dual layered discs can hold 7.95 computer GB* (called DVD-9) and dual layered double sides 15.9* computer GB (called dvd-18).

DVD-RAM
DVD-RAM has the best recording features but it is **not** compatible with most DVD-ROM drives and DVD-Video players. Think more of it as a removable hard disk. DVD-RAM is usually used in some DVD Recorders.
This format is supported by DVDForum.

Read our DVD Writers and Recorders list and read also our DVD Players Compatibility list to see what types your standalone DVD Player supports. You find more detailed comparisons between the formats in the more info section here.

DVD Sizes

The DVD sizes can be a bit confusing. There are basicly 4 different DVD Sizes,

DVD-5, holds around 4 700 000 000 bytes and that is 4.37 computer GB where 1 kbyte is 1024 bytes* . DVD+R/DVD+RW and DVD-R/DVD-RW supports this format. Also called Single Sided Single Layered. **This is the most common DVD Media, often called 4.7 GB Media.**

DVD-10, holds around 9 400 000 000 bytes and that is 8.75 computer GB. DVD+R/DVD+RW and DVD-R/DVD-RW supports this format. Also called Double Sided Single Layered.

DVD-9, holds around 8 540 000 000 bytes and that is 7.95 computer GB. DVD+R supports this format. Also called Single Sided Dual Layered. **This media is called DVD-R9, DVD-R DL, DVD+R9, DVD+R DL or 8.5 GB Media**.

DVD-18, holds around 17 080 000 000 bytes and that is 15.9 computer GB. DVD+R supports this format. Also called Double Sided Dual Layered.

** In the computer world is 1 KB data = 1024 bytes so 4 700 000 000 bytes / 1024 = 4 589 843KB / 1024 = 4482MB / 1024 = 4.37GB. See section 3.3 in the DVDDemystified FAQ here.*

DVD+R/DVD+RW/DVD+R DL and DVD-R/DVD-RW exact sizes
DVD-R/DVD-RW = 4 706 074 624 bytes (4488 MB)
DVD+R/DVD+RW = 4 700 372 992 bytes (4482 MB)
DVD+R DL = 8 547 993 600 bytes (8152 MB)

DVD Write and read speeds

Single Layer(4.7GB) write speeds
1x (CLV) = about 58 minutes

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1839
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

2x (CLV) = about 29 minutes
2.4x (CLV) = about 24 minutes
4x (CLV) = about 14.5 minutes
6x (CLV/ZCLV) = about 10-12 minutes
8x (PCAV/ZCLV) = about 8-10 minutes
12x (PCAV/ZCLV) = about 6.5-7.5 minutes
16x (CAV/ZCLV) = about 6-7 minutes

Dual/Double Layer(8.5GB) write speeds
1x CLV = about 105 minutes
2.4x CLV = about 44 minutes
4x CLV = about 27 minutes

Single Layer (4.7GB) read speeds
1x read speed is 1.321MB/s = ~56 minutes
6x CAV (avg. ~4x) read speed is max 7.93MB/s = ~14 minutes
8x CAV (avg. ~6x) read speed is max 10.57MB/s = ~10 minutes
12x CAV (avg. ~8x) read speed is max 15.85MB/s = ~7 minutes
16x CAV (avg. ~12x) read speed is max 21.13MB/s = ~5 minutes

** write speed time and read speed time is not the same because writing requires some extra steps and also does the faster writing above 6x usually use lower write speeds for some parts of the dvd. 4x DVD speed = 36x CD speed. See section 4.2 in the DVDDemystified FAQ here.*

Other non-standard special DVD formats:

DVD-VCD
is basicly a VCD authored on a DVD. DVD supports the VCD resolution but the audio has to be resampled to 48 khz. If the audio is resampled to 48 khz it is standard DVD-Video. Read more here how to make a such.

DVD-SVCD
is basicly a SVCD authored on a DVD. DVD do not supports the SVCD resolution but it may anyway work and the audio has to be resampled to 48 khz like the DVD-VCD. Read more here how to make a such and download a DVD Sample including a DVD-SVCD here.

DVD-MP3
is MP3s burned on a DVD but very few MP3 capable standalone DVD Players supports it because most Players verify DVD as DVD-Video only. Check our DVD Players list for compatibility.

DVD-ISO
is MPEG ,MPG, VOB files burned on a DVD without any DVD Authoring(making the vob,ifo files) but very few standalone DVD Players supports it because most Players verify DVD as DVD-Video only. Check our DVD Players list for compatibility.

What is **miniDVD / cDVD** then?
Read here on the miniDVD/cDVD page.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1840
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

DVD Samples, DVD-SVCD Samples

DVD Trailers, Dolby Digital, DTS, THX Trailers and some other DVD VOB Trailers

To play the DVD Trailers on your computer use WinDVD and just open the VOB file.

If you would like to burn the VOB Trailers on a DVD you can use Tmpgenc DVD Author, add the vob files and burn.

DVD-SVCD Samples, NTSC and PAL DVD-SVCD samples.

Technical Info for DVD-Video

PAL

Video:
Up to 9.8 Mbps* (9800 kbps*) MPEG2 video
Up to 1.856 Mbps (1856 kbps) MPEG1 video
720 x 576 pixels MPEG2 (Called Full-D1)
704 x 576 pixels MPEG2
352 x 576 pixels MPEG2 (Called Half-D1, same as the CVD Standard)
352 x 288 pixels MPEG2
352 x 288 pixels MPEG1 (Same as the VCD Standard)
25 fps*
16:9 Anamorphic (only supported by 720x576)

Audio:
48000 Hz
32 - 1536 kbps
Up to 8 audio tracks containing Dolby Digital, DTS, PCM(uncompressed audio), MPEG-1 Layer2. One audio track must have MPEG-1, DD or PCM Audio.

Extras:
Motion menus, still pictures, up to 32 selectable subtitles, seamless branching for multiple storylines, 9 camera angles. And also additional DVD-ROM / data files that only can be read by computer DVD drives.

Total:
Total bitrate including video, audio and subs can be max 10.08 Mbps (10080 kbps)

** Mbps = million bits per second*
** kbps = thousand bits per second*
** fps = frames per second*

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1841
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

For more technical DVD-Video details read the DVDDemystified DVD FAQ section 3.4

NTSC (NTSC Film)

Video:
Up to 9.8 Mbps* (9800 kbps*) MPEG2 video
Up to 1.856 Mbps (1856 kbps) MPEG1 video
720 x 480 pixels MPEG2 (Called Full-D1)
704 x 480 pixels MPEG2
352 x 480 pixels MPEG2 (Called Half-D1, same as the CVD Standard)
352 x 240 pixels MPEG2
352 x 240 pixels MPEG1 (Same as the VCD Standard)
29,97 fps*
23,976 fps with 3:2 pulldown = 29,97 playback fps (NTSC Film, this is only supported by MPEG2 video)
16:9 Anamorphic (only supported by 720x480)

Audio:
48000 Hz
32 - 1536 kbps
Up to 8 audio tracks containing DD (Dolby Digital/AC3), DTS, PCM(uncompressed audio), MPEG-1 Layer2. One audio track must have DD or PCM Audio.

Extras:
Motion menus, still pictures, up to 32 selectable subtitles, seamless branching for multiple storylines, 9 camera angles. And also additional DVD-ROM / data files that only can be read by computer DVD drives.

Total:
Total bitrate including video, audio and subs can be max 10.08 Mbps (10080 kbps)

** Mbps = million bits per second*
** kbps = thousand bits per second*
** fps = frames per second*

For more technical DVD-Video details read the DVDDemystified DVD FAQ section 3.4

DVD File/Folder Structure

Explanation:

. BUP = Backup files of the IFO files.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1842
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

. IFO = The IFO files includes information such as chapters, subtitle tracks and audio tracks.

. VOB = The VOB files contains the actual video,audio,subtitles and menus.

Folder	Files	Explanation
AUDIO_TS	(undefined)	DVD Audio
VIDEO_TS	VIDEO_TS.BUP	
	VIDEO_TS.IFO	**The first video play item, IFO,** usally a copyright notice or a menu
	VIDEO_TS.VOB	**The first video play item, VOB**
	VTS_01_0.BUP	
	VTS_01_0.IFO	**Title Set 01, IFO**, usually the main movie
	VTS_01_0.VOB	**Title Set 01, VOB 0, the menu for this title**
	VTS_01_1.VOB	**Title Set 01, VOB 1, the video for this title**
	VTS_01_2.VOB	**Title Set 01, VOB 2,** if larger than 1 GB it will be splitted into several vobs
	VTS_01_3.VOB	**Title Set 01, VOB 3**
	VTS_01_4.VOB	**Title Set 01, VOB 4,** up to 10(0-9) VOB files if necassary
	VTS_02_0.BUP	
	VTS_02_0.IFO	**Title Set 02, IFO**, usually movie extras
	VTS_02_0.VOB	**Title Set 02, VOB 0, the menu for this title**
	VTS_02_1.VOB	**Title Set 02, VOB 1, the video for this title**
	VTS_xx_x.BUP	
	VTS_xx_x.IFO	And so on
	VTS_xx_x.VOB	
	VTS_xx_x.VOB	
	VTS_99_9.VOB	Up to 99(1-99) titles with max 10(0-9) VOB files each

Advanced DVD-Video file specification here

Video File Comparison

Format	VCD	SVCD	DVD	HDDVD HDTV (WMVHD)	AVI DivX XviD	MOV Quick-Time	RM Real-Media	AVI DV

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1843
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

					WMV			
Resolution NTSC/PAL	352x240 352x288	480x480 480x576	720x480² 720x576²	1920x1080² 1280x720²	640x480²	640x480²	320x240²	720x480 720x576
Video Compression	MPEG1	MPEG2	MPEG2, MPEG1	MPEG2 (WMV-MPEG4)	MPEG4	Sorenson, Cinepak, MPEG4 ...	RM	DV
Video bitrate	1150kbps	~2000kbps	~5000kbps	~20Mbps (~8Mbps)	~1000kbps	~1000kbps	~350kbps	25Mbps
Audio Compression	MP1	MP1	MP1, MP2, AC3, DTS, PCM	MP1, MP2, AC3, DTS, PCM	MP3, WMA, OGG, AAC, AC3	QDesign Music, MP3 ...	RM	DV
Audio bitrate	224kbps	~224kbps	~448kbps	~448kbps	~128kbps	~128kbps	~64kbps	~1500kbps
Size/min	10 MB/min	10-20 MB/min	30-70 MB/min	~150MB/min (~60MB/min)	4-10 MB/min	4-20 MB/min	2-5 MB/min	216MB/min
Min/74min CD	74min	35-60min	10-20min	~4min (~10min)	60-180min	30-180 min	120-300 min	3min
Hours/DVD	N/A	N/A	1-2hrs (2-5hrs[a])	~30min (~1hrs)	7-18hrs	3-18hrs	14-35hrs	20min
Hours/ DualLayerDVD	N/A	N/A	2-4hrs (5-9hrs[a])	~55min (~2hrs)	13-30hrs	6-30hrs	25-65hrs	37min
DVD Player Compatibility	Great	Good	Excellent	None	Few	None	None	None
Computer CPU Usage	Low	High	Very High	Super high	Very High	High	Low	High
Quality	Good	Great*	Excellent*	Superb*	Great*	Great*	Decent*	Excellent

kbps = thousand bits per second
Mbps = million bits per second
² approximately resolution, it can be higher or lower
~ approximately bitrate, it can be higher or lower
[a] DVD with lower video quality, similiar to VCD/SVCD video quality
** the video quality depends on the bitrate and the video resolution, higher bitrate and higher resolution generally means better video quality but bigger file size*

More Info

General DVD Info

The DVDDemystified DVD FAQ

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1844
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

DVD-Video file specification

Comparisons between the difference formats, DVD-R vs DVD+R, DVD+RW vs DVD-RW vs DVD-RAM

What's the difference between the various DVD recordable formats, and what should I buy?

DVDDemystified DVD FAQ comparison

Statistics	Latest guides	Latest Tools	Latest Media	Latest DVD Hacks	Latest DVD Writers
586 users over the past 5 minutes	Basics of MeGUI's Avisynth Script Creator	BD Rebuilder	Office Depot 16 x DVD-R	Samsung BD-P1500	LG GP08NU20
VideoHelp server uptime statistics	Womble MPEG Video Wizard - Basic Video Edi...	foobar2000	Memorex 2.4 x DVD+R	Samsung BD-P1600	Pioneer DVR-118L
	Blu-ray Disc standalone results/tests on D...	multiAVCHD	Datawrite Mach4 16 x DVD-R	Sony BDP-S350	Pioneer DVR-218L
		ProgDVB	Ritek Excellent 16 x DVD-R	LG DVX482H	Pioneer DVR-117
		PopCorn MKV AudioConverter		HarmonTec DV-105	Pioneer DVR-217

About Advertise Forum Archive RSS Feeds Statistics

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1845
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

VOB files

An overview to show the logical structure of VOB files, and help you see where all the pieces fit.

MPEG-2 System Stream

A VOB file is an MPEG-2 system stream. This means that it complies 100% with the MPEG-2 system level standard, ISO 13818-1. However, VOB files are a very strict subset of the standard. So while all VOB files are MPEG-2 system streams, not all MPEG-2 system streams comply with the definition for a VOB file.

Pack/sector size

DVD sectors contain 2048 bytes of data, this is also the size of one pack. In MPEG-2 packs are used mainly to group together elements (such as audio and video) that are to be presented simultaneously, and their size is variable. The pack header can also contain timing information used for synchronization.
In DVD-Video each sector is one pack. This adds some overhead, but makes random access to the stream much easier.

Pack contents

Each pack begins with a pack header and contains one or two packets, and no more. The information in one pack is all of one kind, which may be navigation data (a NAV Pack), video, audio, or subpicture.
The NAV pack contains the system header and two fixed length packets called Presentation Control Information (PCI) and Data Search Information (DSI).
The video, audio, and subpicture packs contain only the Packetized Elementary Stream (PES) for the content, and, if needed, a padding packet.

Non-standard stuff

The MPEG-2 system fortunately left provisions for non-standard data in the form of *private* streams. There are two private stream types, only one has timing information in the form of Presentation Time Stamps (PTS) and Decoder Time Stamps (DTS). The actual content of a private stream is determined by the application, in our case, DVD-Video.
Private Stream 1 is the one that has the timing information, and so DVD-Video uses this stream for subpictures and all the additional audio systems (AC3, DTS, LPCM, etc) which are not MPEG. The actual content of each private stream packet is determined by the *sub-stream* number.
The other stream, Private Stream 2, is used for the navigation packets found in the NAV pack.

The VOBU

The next higher logical structure is called the Video OBject Unit, or VOBU. Each VOBU starts with a NAV pack and contains approximately half a second of the program. The size of the VOBU is determined by the video coding unit called a Group Of Pictures (GOP). A VOBU will contain one or more complete GOP, as needed. The last video pack in each VOBU is padded if needed with either a padding stream or stuffing bytes. Audio and subpictures with DTS values within the same range of values as the video are included in each VOBU. Audio is not padded until the end of the cell, therefore audio frames can span VOBUs.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1846
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The Cell

Cells are the next higher logical structure, containing any number of whole VOBUs. Their length and placement is entirely arbitrary and depends on the overall organization of the program (movie). Chapters, multiple angles, titles, and even how the "prev" and "next" buttons on a remote act all dictate the placement of cells.

The VOB

The VOB is a collection of one or more cells. An entire title could use just one VOB, but they usually use more. Sometimes the use is arbitrary, usually along the lines of a new VOB for each chapter, and within the VOB cells for each scene. This is not a requirement. In fact, there is only one place where seperate VOBs are required, and that is multiple angles.

Several 1GB files

All the content for one title set (VTS) is contiguous on the DVD, but broken up into 1GB files in the computer compatible file systems *for the convenience of the various operating systems.* You can see that there really is no break by examining the second or later file and looking at the Logical Block Address (LBA), contained in NAV packs.
The files are broken up without regard to content, which is why it is difficult to process any file but the first, since it most likely will not start at a VOBU (start with a NAV pack). The usual split point is at 524,287 sectors (1,048,574 KB, 1,073,739,776 bytes). In hexadecimal this is 7FFFF sectors (2^{19}-1), or 3FFFF800 bytes.


DVD-Video info home

Copyright © 2001 - 2009 MPUCoder, all rights reserved.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1847
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

What's on a DVD?

What exactly is on a DVD disc? What are VOB, IFO and BUP files? All this will be explained here.

When you access the DVD drive you'll see at least 2 directories:

22.08.2000 03:20 <DIR> AUDIO_TS
22.08.2000 05:53 <DIR> VIDEO_TS

AUDIO_TS is used for DVD Audio and contains the audio files, whereas VIDEO_TS is used to store all data for the movie. There might be more directories which contain DVD-ROM features for your PC.

Now let's have a look at the VIDEO_TS directory:

22.08.2000 05:53 <DIR> .
22.08.2000 03:20 <DIR> ..
22.08.2000 05:23 12'288 VIDEO_TS.BUP
22.08.2000 05:23 12'288 VIDEO_TS.IFO
22.08.2000 05:23 333'824 VIDEO_TS.VOB
22.08.2000 05:23 59'392 VTS_01_0.BUP
22.08.2000 05:23 59'392 VTS_01_0.IFO
22.08.2000 05:23 8'192 VTS_01_0.VOB
22.08.2000 05:27 1'073'643'520 VTS_01_1.VOB
22.08.2000 05:31 1'073'631'232 VTS_01_2.VOB
22.08.2000 05:32 104'785'920 VTS_01_3.VOB
22.08.2000 05:32 88'064 VTS_02_0.BUP
22.08.2000 05:32 88'064 VTS_02_0.IFO
22.08.2000 05:32 59'379'712 VTS_02_0.VOB
22.08.2000 05:36 1'073'436'672 VTS_02_1.VOB
22.08.2000 05:40 1'073'549'312 VTS_02_2.VOB
22.08.2000 05:45 1'073'502'208 VTS_02_3.VOB
22.08.2000 05:49 1'073'371'136 VTS_02_4.VOB
22.08.2000 05:53 1'073'555'456 VTS_02_5.VOB
22.08.2000 05:57 810'952'704 VTS_02_6.VOB

As you can see there's 3 types of files on a DVD: .VOB, .IFO and .BUP:

VOBs - Video OBjects

A VOB contains several streams multiplexed together: Video, Audio and Subtitles. Video is MPEG-2, audio can be AC-3, Linear PCM, Mpeg 2 multichannel or MPEG1 layer2 2 channel audio. AC3 is pretty much the standard and MPEG-2 multichannel can only

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1848
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

be found on very few discs (one example is "In the line of fire, PAL edition") as this format was initially considered to be the standard format in Region2 (Europe and Japan) but was later dropped. PCM is mostly found on music DVDs and MP2 on cheaper productions. PCM is high quality uncompressed audio which takes a lot of space, hence it's not an ideal choice for full length movies with extras and possibly multiple languages. AC3 streams have a bitrate between 192 and 448kbit/s. 192kbit/s is used for 2 channel sound, and 384-448kbit/s for 5.1channel surround.

A VOB can contain one main video stream and several multiangle streams, allowing you to switch (as an example) the perspective during the movie. This feature is mostly used to display storyboards or other extra features during playback. The maximum bitrate of the video stream is 9.8mbit/s. Together, video and audio stream must be below 10mbit/s at any given moment. It's possible to have up to 9 different audio streams and you can usually switch the audio stream during playback (this feature can be disabled during the authoring phase of a DVD). It's also possible to have up to 32 different subtitle streams. Subtitles are 4 color bitmaps which are overlayed over the video stream, they're usually not encoded into the video stream.

Let's have a closer look at a VOB using our old favorite vStrip:

```
Scanning for stream id's, press control-c to quit...
Found 0xBF = Private 2 [@LBA 0]
Found VOB-ID: 01/CELL-ID: 01 [@LBA 0]
Encountered encrypted sector, attempting key recovery [@LBA 1]
Deduced key: 0xC00374C61C (2/2 vkey(s))
Found 0xE0 = Video 0 [PTS 0:00:00.290 @LBA 1]
Width = 720
Height = 480
Aspect-ratio = [3] 16:9 display
Frame-rate = [4] 29.97 (30000/1001) fps
Found 0xBD = Private 1, sub 0x80 [PTS 0:00:00.224 @LBA 99]
Found 0xBD = Private 1, sub 0x81 [PTS 0:00:00.224 @LBA 100]
Found 0xBD = Private 1, sub 0x82 [PTS 0:00:00.224 @LBA 101]
Found 0xBD = Private 1, sub 0x83 [PTS 0:00:00.224 @LBA 102]
Found 0xBE = Padding [@LBA 117]
Found 0xBD = Private 1, sub 0x20 [PTS 0:00:00.724 @LBA 169]
Found 0xBD = Private 1, sub 0x21 [PTS 0:00:00.724 @LBA 170]
Found VOB-ID: 02/CELL-ID: 01 [@LBA 378]
```

As said before there's usually just one video stream and it always has ID 0xE0. PTS is the Program Time Stamp and states when a certain stream starts with respect to the beginning of the VOB. As you can see the resolution is 720x480 so it's an NTSC DVD. The frame-rate of 29.97 basically states the same fact. The DAR of the stream is 16:9. Then there's 4 audio streams: 0x80 - 0x83. Note that these are AC3 streams. All AC3 streams are in stream 0xBD, substreams 0x8x. When there's a DTS stream it usually has stream id 0xBD 0x88 or 0xBD 0x89. PCM streams are also located in stream 0xBD, they have stream IDs 0xAx, from 0xA0 up to 0xA9. MP2 audio finally is located in the stream 0xCx. Here's an example from a PCM and MP2 stream:

```
Found 0xBD = Private 1, sub 0xA0 [PTS 0:11:01.479 @LBA 4]
```


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1849
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Found 0xC0 = Audio 0 [PTS 0:00:00.440 @LBA 25]

There are also 2 substreams, 0x20 and 0x21. All subtitle streams are located in the 0xBD stream as well.

Furthermore there are usually padding streams (0xBE) but they don't concern us.

You might also have noticed the VOB and Cell IDs. A Cell is the smallest unit on a DVD. On a simple DVD this usually represents a chapter but it gets more complicated in many discs. If you're interested about Cells on a DVD I suggest you start learning how to use Scenarist, because only then will you truly understand how these works. When it comes to VOB IDs I still haven't found how they are created exactly and to what kind of structure they correspond. Usually each unit on its own has its own VOB ID, for instance the main movie and the trailer. In multiangle and seamless branching titles each angle has its own VOB ID.

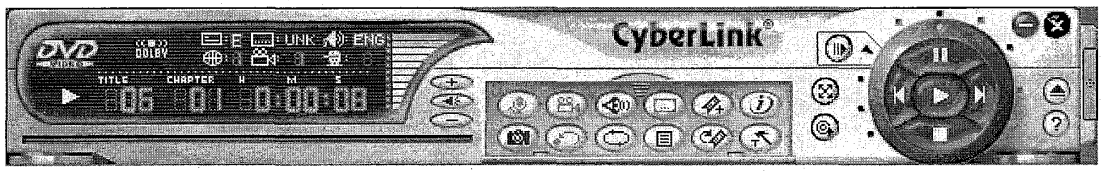
IFOs - InFOrmation

IFO Files give the player important navigational information, like where a chapter starts, where a certain audio or subtitle stream is located, etc. This is the reason why it's only possible to rip certain parts of a movie (like a chapter) with a ripper which can read this files.

Or old friend vStrip can and let's have a glimpse at the output we get when using ifo parsing (sample command line: vStrip f:\video_ts\vts_02_1.vob -if:\video_ts\vts_02_0.ifo)

Parsing "f:\video_ts\vts_02_0.ifo"...
0. Length: 02:07:15:24 in 15 cell(s).
1. Length: 00:00:01:01 in 1 cell(s).
Scanning for stream id's, press control-c to quit...

As you can see there's 2 PGCs or ProGram Chains in this movie. PGC 0 represents the main movie whereas PGC 1 the studio logo at the beginning. This structure might be a lot more complicated but usually it's as easy as shown above. Multiangle movies will give you several PGCs having the same length, and seamless branching movies several PGC with different length. By selecting the right PGC you can get the right version of the movie. Also.. the PGC corresponds to the Title number being displayed in your player. Here's just an example (Matrix - follow the white rabbit feature = Title 6 or in other words PGC 5 since the PC starts counting at 0 internally).



Last but not least IFO files are not encrypted.

BUPs - BackUP

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1850
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

BUP files are just backup files off the IFOs. As their counterparts they are not encrypted.

Look closer

Now that we know what each filetype is for let's look at the actual example again:

22.08.2000 05:23 12'288 VIDEO_TS.BUP
22.08.2000 05:23 12'288 VIDEO_TS.IFO
22.08.2000 05:23 333'824 VIDEO_TS.VOB
22.08.2000 05:23 59'392 VTS_01_0.BUP
22.08.2000 05:23 59'392 VTS_01_0.IFO
22.08.2000 05:23 8'192 VTS_01_0.VOB
22.08.2000 05:27 1'073'643'520 VTS_01_1.VOB
22.08.2000 05:31 1'073'631'232 VTS_01_2.VOB
22.08.2000 05:32 104'785'920 VTS_01_3.VOB
22.08.2000 05:32 88'064 VTS_02_0.BUP
22.08.2000 05:32 88'064 VTS_02_0.IFO
22.08.2000 05:32 59'379'712 VTS_02_0.VOB
22.08.2000 05:36 1'073'436'672 VTS_02_1.VOB
22.08.2000 05:40 1'073'549'312 VTS_02_2.VOB
22.08.2000 05:45 1'073'502'208 VTS_02_3.VOB
22.08.2000 05:49 1'073'371'136 VTS_02_4.VOB
22.08.2000 05:53 1'073'555'456 VTS_02_5.VOB
22.08.2000 05:57 810'952'704 VTS_02_6.VOB

The VIDEO_TS.* files represent the first play item. This is an item being set during authoring and is the first thing being played when the disc is inserted in the player. Usually this is just a copyright notice, but it could also be a menu where to select the language of the menu, some trailers, etc. The video_ts.vob contains the video and audio data, the video_ts.ifo the navigational data and video_ts.bup is the backup for video_ts.ifo.

There's 2 more ifo files: vts_01_0.ifo and vts_02_0.ifo. The first 2 numbers in the name tell the title number. vts_01_* is title 1, vts_02_* title 2, etc. Obviously there can be 99 titles at the maximum (vts_00_* does not exist). Each title can have 10 VOB files, whereas the first VOB (vts_XX_0.vob) always contains the menu for that title. Each title has at least 2 VOBs (one for the menu and one for the main feature) and exactly one ifo and bup file.

In our case title 1 contains the behind the scenes featurette and has its own menu. Title 2 contains the main movie, also with its own menu.

As you can see the maximum size of a VOB file is 1GB. In fact the files must be smaller than or exactly one GB (1024x1024x1024Bytes), but there are certain which aren't and these can't be played on a PC.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1851
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Now the last question: how to locate the main movie on a disc? The largest .ifo file will give you the title number and then you need all the VOB files except for the menu VOB (unless you want the menu as well, of course). Most Rippers (if they don't do ifo parsing) proceed according to this rule to select the main movie and the ifo to be copied.

Inside a VOB

DVD video content is broken into titles (movies or albums) and parts of titles (chapters or songs). Titles are made up of cells linked together by one or more Program Chains (PGCs). A cell is the smallest video unit on a DVD. Often a cell is equal to a chapter but not always. Cells are sometimes grouped together using VOB IDs. Examples are Warner Brothers movies where the main movie has one VOB ID on one layer and another one on another layer, multiangle movies (at the branching points each angle gets its own VOB ID, and then when they get back together we have the next VOB ID, etc., and seamless branching movies - where we have different VOB IDs for the different parts of movies that are selected to play when selecting a certain playback option).

This document was last updated on October 9, 2003

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1852
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/224,298	09/01/2011	Kourosh Soroushian	D1-023538	9140

71897 7590 05/19/2014

KPPB LLP
2400 EAST KATELLA AVENUE
SUITE 1050
ANAHEIM, CA 92806

EXAMINER
ELAHI, SHAWKAT

ART UNIT	PAPER NUMBER
2486	

NOTIFICATION DATE	DELIVERY MODE
05/19/2014	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

docketing@cpaglobal.com
PAIR@KPPB.COM

Office Action Summary	Application No. 13/224,298	Applicant(s) SOROUSHIAN ET AL.	
	Examiner SHAN ELAHI	Art Unit 2486	AIA (First Inventor to File) Status No

-- *The MAILING DATE of this communication appears on the cover sheet with the correspondence address* --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTHS FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03/19/2014*.

☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on _____.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

4)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims*

5)☒ Claim(s) *1-18* is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6)☐ Claim(s) _____ is/are allowed.

7)☒ Claim(s) *1-18* is/are rejected.

8)☐ Claim(s) _____ is/are objected to.

9)☐ Claim(s) _____ are subject to restriction and/or election requirement.

* If any claims have been determined allowable, you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov.

Application Papers

10)☐ The specification is objected to by the Examiner.

11)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

Certified copies:

a)☐ All b)☐ Some** c)☐ None of the:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

** See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☒ Information Disclosure Statement(s) (PTO/SB/08a and/or PTO/SB/08b) Paper No(s)/Mail Date *3/9/2014, 12/18/2013*.

3) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.

4) ☐ Other: _____.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1854
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

DETAILED ACTION

1. The claims and only the claims form the metes and bounds of the invention. "Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure. In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023, 1027-28 (Fed. Cir.1997). Limitations appearing in the specification but not recited in the claim are not read into the claim. In re Prater, 415 F.2d 1393, 1404-05, 162 USPQ 541, 550-551 (CCPA 1969)" (MPEP p 2100-8, c 2, l 45-48; p 2100-9, c 1, l 1-4). The Examiner has full latitude to interpret each claim in the broadest reasonable sense. The Examiner will reference prior art using terminology familiar to one of ordinary skill in the art. Such an approach is broad in concept and can be either explicit or implicit in meaning.

Status of Claims

2. This Office Action is in response to the Remarks filed on March 19th 2014. Claims 1-18 are pending examination. The present application is being examined under the pre-AIA first to invent provisions

Information Disclosure Statement

3. The information disclosure statement (IDS) submitted on 3/19/2014, 12/18/2013 is noted. The submission is in compliance with the provisions of 37 CFR 1.97 and

1.98. Accordingly, the information disclosure statement is being considered by the examiner.

Response to Amendments/Arguments

4. Applicant's arguments/amendments (see remarks filed on March 19th 2014) with respect to the 35 U.S.C § 112 rejections (claim 18) have been fully considered and are not persuasive. Therefore, 35 U.S.C § 112 rejections in regards to claim 18 is maintained.

5. Applicant's arguments/amendments see remarks, filed on March 19th 2014, with respect to 35 U.S.C § 101 rejections have been fully considered and are not persuasive. Therefore, 35 USC 101 rejections in regards to claim 18 is maintained.

6. Applicant's arguments/amendments (see remarks filed on March 19th 2014) with respect to the Double Patenting rejections (claims 1-18) have been fully considered and are not persuasive. Therefore, Double Patenting rejections in regards to claims 1-18 are maintained.

7. Applicant's arguments/amendments see remarks, filed on March 19th 2014, with respect to 35 U.S.C § 112 rejections have been fully considered and are not persuasive. Therefore, 35 USC 112 rejections in regards to claim 18 is maintained.

8. Applicant's remarks/arguments filed on March 19th 2014 with respect to rejection of claims 1-18 under 35 U.S.C. §103 have been fully considered but are not persuasive. The Examiner would like to point out that this action is made final (See MPEP 706.07A).

9. On pages 4-5 applicant argues that Hamilton does not teach the associating of a font file with a container file including a subtitle stream as recited in claim. While the applicant's points are understood, the examiner respectfully disagrees. See for example Zoest Fig.2.1, paragraph [0161]-[0164]. There Zoest discloses the "hdrl' (e.g. header list) list chunk is defined in the AVI file format and two subtitle tracks. Furthermore, see Eerenberg paragraphs [0014]-[0016] and paragraphs [0040]-[0047]. There Eerenberg discloses the MPEG encoding standard and each GOP comprising an intra frame encoding pictures. Furthermore, see Hamilton, Figure 2 to Figure 13 and Fig.15-Fig.17, Col.9 lines 4-67 and Col.14 lines 58 to Col.15 lines 26. There Hamilton discloses font files and how characters of a font with compressed and appears in screen views of a display

unit driven by a display subsystem. Therefore the rejection (claims 1-18) has been maintained.

Double Patenting

10. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory double patenting rejection is appropriate where the claims at issue are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).
A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the reference application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint

research agreement. A terminal disclaimer must be signed in compliance with 37 CFR 1.321(b).

The USPTO internet Web site contains terminal disclaimer forms which may be used. Please visit http://www.uspto.gov/forms/. The filing date of the application will determine what form should be used. A web-based eTerminal Disclaimer may be filled out completely online using web-screens. An eTerminal Disclaimer that meets all requirements is auto-processed and approved immediately upon submission. For more information about eTerminal Disclaimers, refer to http://www.uspto.gov/patents/process/file/efs/guidance/eTD-info-I.jsp.

11. Claims 1-18 are rejected on the ground of nonstatutory double patenting as being unpatentable over claims 1-25 of Co-pending application 13/224,295. Although the claims at issue are not identical, they are not patentably distinct from each other because the claims in the current application are broader than the claims of the copending application 13/224295 substantially teaches (Claim 1 limitation) a method of playing back encoded media, where the media is encoded as a plurality of alternative streams and at least one subtitle stream, the method comprising: requesting and buffering portions of video from at least one of the alternative streams using a playback device; requesting information indicative of a font utilized by a font-rendering engine to render text from a selected subtitle stream; downloading at least one font file when the font is not present on the playback device; requesting and buffering at least a portion of the selected

subtitle stream; decoding the buffered portions of video using a decoder on the playback device; rendering the portions of the subtitle stream corresponding to the buffered portion of video using a font-rendering engine configured by the at least one downloaded font file; and performing synchronized playback of the decoded video and rendered subtitles using the playback device.

Claim Rejections - 35 USC § 112

12. Use of the word "means" (or "step for") in a claim with functional language creates a rebuttable presumption that the claim element is to be treated in accordance with 35 U.S.C. § 112(f) (pre-AIA 35 U.S.C. 112, sixth paragraph). The presumption that § 112(f) (pre-AIA § 112, sixth paragraph) is invoked is rebutted when the function is recited with sufficient structure, material, or acts within the claim itself to entirely perform the recited function.

Absence of the word "means" (or "step for") in a claim creates a rebuttable presumption that the claim element is not to be treated in accordance with 35 U.S.C. § 112(f) (pre-AIA 35 U.S.C. 112, sixth paragraph). The presumption that § 112(f) (pre-AIA § 112, sixth paragraph) is not invoked is rebutted when the claim element recites function but fails to recite sufficiently definite structure, material or acts to perform that function.

Claim elements in this application that use the word "means" (or "step for") are presumed to invoke § 112(f) except as otherwise indicated in an Office action. Similarly, claim elements that do not use the word "means" (or "step for") are

presumed not to invoke § 112(f) except as otherwise indicated in an Office action.

Limitation of the claim 18 recite(s) "a processor configure by" are being treated in accordance with 112(f). A processor configured by a source encoding application does not have support whether the processor is a general purpose processor or microprocessor or software or the combination of the hardware or software (see applicant's specification paragraphs 29, 50. Because the claim 18 does not have support in the specification therefore, there is 112(b) indefiniteness.

13. The following is a quotation of 35 U.S.C. 112(b):
(b) CONCLUSION.—The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the inventor or a joint inventor regards as the invention.

The following is a quotation of 35 U.S.C. 112 (pre-AIA), second paragraph:
The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

14. Regarding claim 18, the phrase "a processor configure by" renders the claim indefinite because it is unclear whether the limitation(s) following the phrase are part of the claimed invention. See MPEP § 2173.05(d).

Claim Rejections - 35 USC § 101

15. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claim 18 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter, as it does not fall under any of the

statutory classes of inventions. Claim 18, for example, recite(s) "a source encoder configure to...: a processor configure by ..." but these components, according to paragraph [0050] of the of Applicant's specification, are directed towards software/programs. Therefore, since the specification explicitly states the source encoder is a server. Therefore, claim 18 does not fall under the statutory classes of invention set forth (according to MPEP2111).

Claim Rejections - 35 USC § 103

16. The following is a quotation of pre-AIA 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

17. The factual inquiries set forth in *Graham v. John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under pre-AIA 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.

2. Ascertaining the differences between the prior art and the claims at issue.

3. Resolving the level of ordinary skill in the pertinent art.

4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

18. Claims 1-11, and 15-18 are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over Zoest et al hereafter Zoest (US Publication No 2005/0207442 A1) in view of Eerenberg et al hereafter Eerenberg (US Publication No 2003/0231863 A1) and further in view of Hamilton (Patent No 5,404,436).

Regarding Claim 1, Zoest disclose a method of encoding source media including at least one subtitle track as a plurality of alternative video streams and at least one subtitle stream using a source encoder, the method comprising: selecting a portion of the source media using the source encoder (paragraph [0004] multimedia files and system for generating, distributing, decoding and encoding video tracks);

transcoding the selected portion of the source media into a plurality of alternative portions of encoded video using the source encoder (paragraph [0257] the video source stored and then encoded using a video encoder and paragraph [0258] various encoders, decoders and transcoder, the software used to perform the encoding, decoding and transcoding and paragraph [0266] video decoder and video encoder), where each alternative portion is encoded using a different set of encoding parameters (paragraph [0445] a one pass rate control techniques commences by initializing the encoder with a desired bit rate, the video frame rate and variety of other parameters (i.e. encoding parameters)),

writing each of the alternative portions of encoded video to a separate container file using the source encoder (paragraph [0262] meta data statement are written

into an XML configuration file (i.e. container files) paragraph [0427] encode files in accordance with MPEG-4 standard);

writing a subtitle stream segment from each subtitle track (abstract multimedia files can contain multiple video tracks, multiple audio tracks, multiple subtitle, data that can be used to generate a menu interface to access the content of the file and meta data and paragraphs [0036]-[0037], [0044] and paragraph [0550]) corresponding to the selected portion of the source media to a separate container file using the source encoder (paragraph [0090] XML data, the data types indicates the type of the value information and the language contains information identifying a specific language, and paragraph [0149] an AVI file and the AVI format and paragraph [0153] media in compliant AVI file format and paragraph [0161] single subtitle track and two subtitle track contains global information for the entire file such as the number stream within the file and width and height of the video contained in the multimedia file); and

Although Zoest does not explicitly disclose commences with an intra frame at the start of a closed group of pictures (GOP). However Eerenberg discloses a digital video information processing, reproduction, and recording based on the MPEG standards to retrieve intra encoded picture and to generate inter encoded picture and encoded picture comprising groups of pictures (i.e. commences with an intra frame) (see paragraphs [0001], [0011]-[0013]). It would have been obvious to one of the ordinary skill in the art to modify the method for multimedia files generation, distribution, decoding, and encoding of Zoest to process the video

signal as taught by Eerenberg. The motivation/suggestion would have been to encode and decode video and audio track with the video information (multimedia files) using the MPEG standards (Eerenberg: paragraphs [0003], [0011]-[0015]).

Zoest and Eerenberg do not explicitly disclose associating a font file with at least one of the container files containing subtitle stream segments. However Hamilton discloses method and apparatus employs font metrics data related to font bitmaps for characters of a desire font (Abstract and Col.3 lines 23 to Col.4 lines 55 and Col. 9 line 10 the bitmap font files provides bitmaps of compressed capital letters and Col.15 lines 1-16). It would have been obvious to one of the ordinary skill in the art to modify the method Zoest and Eerenberg to compress composite characters and decompress for display on computer display systems as taught by Hamilton. The motivation/suggestion would have been to compress and decompress composite characters and to display on the computer display system having no character height restriction (Hamilton: Col.3 line 23 to Col. 4 lines 55).

The Applicant is reminded that the preamble is generally not accorded any patentable weight where it merely recites the purpose of a process or the intended use of a structure, and where the body of the claim does not depend on the preamble for completeness but, instead, the process steps or structural limitations are able to stand alone. See In re Hirao, 535 F.2d 67, 190 USPQ 15 (CCPA 1976) and Kropa v. Robie, 187 F.2d 150, 152, 88 USPQ 478, 481 (CCPA 1951).

Regarding Claim 2, Zoest and Eerenberg disclose the method of claim 1. Zoest and Eerenberg do not explicitly disclose further comprising compressing the font file. However Hamilton discloses a computer method to compress and decompress bit map font characters (Col.3 lines 23-36 and Col.4 lines 5-29). It would have been obvious to one of the ordinary skill in the art to modify the method Zoest and Eerenberg to compress composite characters and decompress for display on computer display systems as taught by Hamilton. The motivation/suggestion would have been to compress and decompress composite characters and to display on the computer display system having no character height restriction (Hamilton: Col.3 line 23 to Col. 4 lines 55).

Regarding Claim 3, Zoest and Eerenberg disclose the method of claim 1. Zoest and Eerenberg do not explicitly disclose further comprising reducing the size of the font file by subsetting the font file, where the subsetted font file includes all of the characters within the subtitle stream segments within the container file with which the font file is associated. However Hamilton discloses a computer method to compress and decompress bit map font characters (Col.3 lines 23-36 and Col.4 lines 5-29, Col.6 lines 43-67 in the font metrics file 28 each characters is defined). It would have been obvious to one of the ordinary skill in the art to modify the method Zoest and Eerenberg to compress composite characters and decompress for display on computer display systems as taught by Hamilton. The

motivation/suggestion would have been to compress and decompress composite characters and to display on the computer display system having no character height restriction (Hamilton: Col.3 line 23 to Col. 4 lines 55).

Regarding Claim 4, Zoest and Eerenberg disclose the method of claim 1. Zoest and Eerenberg do not explicitly disclose further comprising breaking the font file into a plurality of font files, where at least one of the font files includes characters displayed during a subtitle stream segment. However Hamilton discloses a computer method to compress and decompress bit map font characters and files are used by the application 10 to determined how characters will be sized when they are printed and the font metrics file of a font is divided into three section which contains characters metrics data (Col.3 lines 23-36 and Col.4 lines 5-29, Col.6 lines 35-67 font metrics files 28 provide data and how characters will be rendered and in the font metrics file 28 each characters is defined). It would have been obvious to one of the ordinary skill in the art to modify the method Zoest and Eerenberg to compress composite characters and decompress for display on computer display systems as taught by Hamilton. The motivation/suggestion would have been to compress and decompress composite characters and to display on the computer display system having no character height restriction (Hamilton: Col.3 line 23 to Col. 4 lines 55).

Regarding Claim 5, Zoest disclose the method of claim 4, corresponding to

different subtitle stream segments (paragraph [0534] generating video chunks and segmenting video into chunks 1 to N paragraph [0535]).

Zoest does not explicitly disclose wherein breaking the font file into a plurality of font files comprises generating font files. However Hamilton discloses a computer method to compress and decompress bit map font characters and files are used by the application 10 to determined how characters will be sized when they are printed and the font metrics file of a font is divided into three section which contains characters metrics data (Col.3 lines 23-36 and Col.4 lines 5-29, Col.6 lines 35-67 font metrics files 28 provide data and how characters will be rendered and in the font metrics file 28 each characters is defined). It would have been obvious to one of the ordinary skill in the art to modify the method Zoest to compress composite characters and decompress for display on computer display systems as taught by Hamilton. The motivation/suggestion would have been to compress and decompress composite characters and to display on the computer display system having no character height restriction (Hamilton: Col.3 line 23 to Col. 4 lines 55).

Regarding Claim 6, Zoest disclose the method of claim 5, wherein is assembled as attachments to a single container file (paragraphs [0565] multimedia files that combines video, audio and subtitles can requires selection of a particular audio track, video track and subtitle track and paragraph [0284] combining the XML configuration file and AVI files).

Zoest does not explicitly disclose the plurality of font files. However Hamilton discloses a computer method to compress and decompress bit map font characters and the software program running on a digital processor which process the conversion of the bitmaps locally to a desire computer display system have no restrictions (Col.3 lines 23-36 and Col.4 lines 5-29, Col.6 lines 43-67 in the font metrics file 28 each characters is defined Col.14 lines 59 to Col. 15 lines 17). It would have been obvious to one of the ordinary skill in the art to modify the method Zoest and Eerenberg to compress composite characters and decompress for display on computer display systems as taught by Hamilton. The motivation/suggestion would have been to compress and decompress composite characters and to display on the computer display system having no character height restriction (Hamilton: Col.3 line 23 to Col. 4 lines 55).

Regarding Claim 7, Zoest disclose the method of claim 5. Zoest does not explicitly disclose wherein starting and ending timecodes are used to indicate a time period over which each font file is used for rendering subtitles. However Hamilton discloses a computer method to compress and decompress bit map font characters and the software program running on a digital processor which process the conversion of the bitmaps locally to a desire computer display system have no restrictions and that data renders the character on a screen display (Col.3 lines 23-36 and Col.4 lines 5-29, Col.6 lines 43-67 in the font metrics file 28 each characters is defined Col.14 lines 59 to Col. 15 lines 17 and

Col. 9 lines 14-32 the composite character ... and how data renders the character, see Fig. 6, Fig. 8 and Fig. 10 for rendering of the character on a screen display). It would have been obvious to one of the ordinary skill in the art to modify the method Zoest and Eerenberg to compress composite characters and decompress for display on computer display systems as taught by Hamilton. The motivation/suggestion would have been to compress and decompress composite characters and to display on the computer display system having no character height restriction (Hamilton: Col.3 line 23 to Col. 4 lines 55).

Regarding Claim 8, Zoest disclose the method of claim 5. Zoest does not explicitly disclose further comprising generating a union font file using the plurality of font files that includes characters that appear in at least two of the plurality of font files and subsetting the plurality of font files to remove the characters that are in the union font file. However Hamilton discloses a computer method to compress and decompress bit map font characters and the software program running on a digital processor which process the conversion of the bitmaps locally to a desire computer display system have no restrictions and that data renders the character on a screen display and the font metrics file of a font is divided into three sections ... and the first entry in that line is the code for the character, the code used in the illustration is the International Standards Organization ISO Latin 1 encoding scheme (Col.3 lines 23-36 and Col.4 lines 5-29, Col.6 lines 43-67 in the font metrics file 28 each characters is defined and

Col. 7 lines 1-14 and Col.14 lines 59 to Col. 15 lines 17 and Col. 9 lines 14-32 the composite character ... and how data renders the character, see Fig. 6, Fig. 8 and Fig. 10 for rendering of the character on a screen display). It would have been obvious to one of the ordinary skill in the art to modify the method of Zoest and Eerenberg to compress composite characters and decompress for display on computer display systems as taught by Hamilton. The motivation/suggestion would have been to compress and decompress composite characters and to display on the computer display system having no character height restriction (Hamilton: Col.3 line 23 to Col. 4 lines 55).

Regarding Claim 9, Zoest as modified disclose the method of claim 5, further comprising subsetting each font file so that characters used in a previous subtitle stream segments are removed from the font file (Hamilton: Col. 8 lines 36-51 command screen display of text and Col.9 lines 4-13).

Regarding Claim 10, Zoest discloses the method of claim 1, wherein the selected portions of source media are selected based upon their duration (paragraph [0243] the value "duration" specified the time at the starting and ending point of the subtitle).

Regarding 11, Zoest disclose the method of claim 10, wherein the selected portions of source media each have a duration of two seconds (paragraph [0243]

for timing).

Regarding Claim 15, Zoest disclose the method of claim 1, further comprising creating a top level index file that identifies each of the container files containing the alternative portions of encoded video and identifies the container files that include the at least one subtitle stream (paragraph [0151] a multimedia files and paragraph [0249] the 'idxl' chunk 44 is optional chunk that can be used to index the 'data' and paragraph [0388]).

Regarding Claim 16, Zoest disclose the method of claim 1, wherein the font file associated with the at least one of the container files containing subtitle stream segments is attached to the container file (paragraph [0537]-[0538] generating subtitle) and the container file includes a separate element containing metadata describing (paragraph [0564] meta data included within the multimedia files).

Zoest does not explicitly disclose the font file. However Hamilton discloses method and apparatus employs font metrics data related to font bitmaps for characters of a desire font (Abstract and Col.3 lines 23 to Col.4 lines 55 and Col. 9 line 10 the bitmap font files provides bitmaps of compressed capital letters and Col.15 lines 1-16). It would have been obvious to one of the ordinary skill in the art to modify the method Zoest and Eerenberg to compress composite characters and decompress for display on computer display systems as taught by Hamilton. The motivation/suggestion would have been to compress and decompress

composite characters and to display on the computer display system having no character height restriction (Hamilton: Col.3 line 23 to Col. 4 lines 55).

Regarding Claim 17, Zoest disclose the method of claim 16 wherein the metadata includes the location (paragraph [0079] multimedia files includes meta data and paragraph [0551]).

Zoest does not explicitly disclose the font file within the container file and information describing the file. However Hamilton discloses method and apparatus employs font metrics data related to font bitmaps for characters of a desire font (Abstract and Col.3 lines 23 to Col.4 lines 55 and Col. 9 line 10 the bitmap font files provides bitmaps of compressed capital letters and Col.15 lines 1-16). It would have been obvious to one of the ordinary skill in the art to modify the method Zoest and Eerenberg to compress composite characters and decompress for display on computer display systems as taught by Hamilton. The motivation/suggestion would have been to compress and decompress composite characters and to display on the computer display system having no character height restriction (Hamilton: Col.3 line 23 to Col. 4 lines 55).

Regarding Claim 18, a source encoder of claim 18 implemented to perform the method of claim 1.

Zoest further discloses, a processor configured by a source encoding application to select a portion of the source media (paragraph [0144] a system for encoding, distribution and decoding includes a computing device, and claim 11, paragraph [0259] the entire system can be implemented on a computer using software and connected to a storage device such as hard drive and paragraph [0285] a computer is configured to parse information from the XML configuration file and the entire encoding system can be implemented using a computer configured to perform the various functions paragraph [0299] and paragraph [0301] the processing operations).

The Applicant is reminded that the preamble is generally not accorded any patentable weight where it merely recites the purpose of a process or the intended use of a structure, and where the body of the claim does not depend on the preamble for completeness but, instead, the process steps or structural limitations are able to stand alone. See In re Hirao, 535 F.2d 67, 190 USPQ 15 (CCPA 1976) and Kropa v. Robie, 187 F.2d 150, 152, 88 USPQ 478, 481 (CCPA 1951).

19. Claims 12-14 are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over Zoest et al hereafter Zoest (US Publication No 2005/0207442 A1) in view of Eerenberg et al hereafter Eerenberg (US Publication No 2003/0231863 A1) and Hamilton (Patent No 5,404,436) and further in view of Knowles et al hereafter Knowles (US 2007/0180125 A1).

Regarding Claim 12, Zoest, Eerenberg and Hamilton disclose the method of claim 1 except wherein the container files containing the alternative streams and the subtitle streams are Extensible Binary Markup Language (EBML) container files. However Knowles discloses a method for synchronizing mobile internet browser with proxy server and the data protocol set similar to HTTP, XML and Binary Extensible Markup language for encoded document (paragraphs [0002], [0020]-[0022], [0030]). It would have been obvious to one of the ordinary skill in the art to modify the method Zoest, Eerenberg and Hamilton to synchronizing and transmitting from the browser to the server for secure communication between proxy server and internet browser as taught by Knowles. The motivation/suggestion would have been for a secure communication between proxy server and the internet browser (Knowles: paragraph [0004], [0012]-[0014, [0020]).

Regarding Claim 13, Zoest as modified disclose the method of claim 12, wherein: the element of the EBML container file to which each alternative portion of encoded video is written is a Cluster element and each Cluster element to which each of the alternative portions of encoded video is written is assigned the same time code (Knowles: paragraph [0034] the state data includes the URL of an HTML page ... can compares the time stamps/expiration information, [0045]).

Regarding Claim 14, Zoest as modified disclose the method of claim 12, wherein

the element of the EBML container file to which the subtitle stream segment is written is a Cluster element containing the same time code as the Cluster elements to which the alternative portions of encoded video are written (Knowles: paragraphs [0020]-[0022], [0030]-[0034]).

20. ***References cited, not replied upon**

The prior art made of record and not relied upon is considered pertinent to applicants disclosure. Below the list of the references are:

1. US 2008/0192818 A1 – (see at least abstract and detailed description of the invention).
2. US 2008/0256105 A1 - (see at least abstract).
3. US 8,321,556 B1 – (see at least abstract).
4. US 8,456,380 B2 – (see at least abstract).
5. US 8,289,338 B2 – (see at least abstract for generating and displaying textual information for a multimedia files).

Conclusion

21. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed

within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to SHAN ELAHI whose telephone number is (571)270-1442. The examiner can normally be reached on Monday to Friday 7:30 AM to 5:00 PM (EST).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Mehrdad Dastouri can be reached on (571) 272-7418. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/SHAN ELAHI/
Examiner, Art Unit 2486

/ANDY RAO/
Primary Examiner, Art Unit 2486
May 14, 2014

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/905,804	05/30/2013	Abhishek Shivadas	D1-03350	7019

71897 7590 07/25/2014

KPPB LLP
2400 EAST KATELLA AVENUE
SUITE 1050
ANAHEIM, CA 92806

EXAMINER	
HUANG, JEN-SHI	
ART UNIT	PAPER NUMBER
2423	

NOTIFICATION DATE	DELIVERY MODE
07/25/2014	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

docketing@cpaglobal.com
PAIR@KPPB.COM

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1879
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Office Action Summary	Application No. 13/905,804	Applicant(s) SHIVADAS ET AL.	
	Examiner JEN-SHI HUANG	Art Unit 2423	AIA (First Inventor to File) Status Yes

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTHS FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 5/30/2013.

☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on _____.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

4)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims*

5)☒ Claim(s) 1-25 is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6)☐ Claim(s) _____ is/are allowed.

7)☒ Claim(s) 1-25 is/are rejected.

8)☐ Claim(s) _____ is/are objected to.

9)☐ Claim(s) _____ are subject to restriction and/or election requirement.

* If any claims have been determined allowable, you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov.

Application Papers

10)☐ The specification is objected to by the Examiner.

11)☒ The drawing(s) filed on 5/30/2013 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

Certified copies:

a)☐ All b)☐ Some** c)☐ None of the:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

** See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Information Disclosure Statement(s) (PTO/SB/08a and/or PTO/SB/08b) Paper No(s)/Mail Date _____.

3) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

4) ☐ Other: _____.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1880
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

DETAILED ACTION

1. The present application, filed on or after March 16, 2013, is being examined under the first inventor to file provisions of the AIA.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a)(1) the claimed invention was patented, described in a printed publication, or in public use, on sale or otherwise available to the public before the effective filing date of the claimed invention.

3. Claims 1, 4-10, 13-16, 19 and 22-23 are rejected under 35 U.S.C. 102(a)(1) as being unpatentable over McLaren et al. (6,064,794)

Claim 1, McLaren discloses a method of video playback with trick play (**Fig. 3 and Col. 2, lines 12-15; Fig. 5 and Col. 2, lines 19-21 → A method for selecting between normal or trick play reproduction speed.**), comprising:

downloading (**Col. 7, lines 31-33**) over a communication network (**Fig. 4, el. 511 and 551**), files of encoded video including interspersed key frames (**Col. 7, lines 33-37 → number of trick play speed, look-up tables**);

playing back video from the downloaded files (**Fig. 5, el. 275 and Col. 7, lines 54-64**);

creating a trick play index based on the downloaded files indicating network locations of the key frames (**Col. 4, lines 17-22 → look-up tables may**

be employed which list the file byte offsets for each GOP in each play speed stream); and

receiving a trick play request (**Col. 6, lines 54-59 → commands may include, program manipulation or "trick-play" features**) and, in response thereto,

downloading the key frames from the indicated network locations (**Fig. 5, el. 550 and Col. 8, lines 10-11**), and

playing back video from the downloaded key frames (**Fig. 5, el. 600 and Col. 8, lines 12-14**).

Claim 4, the method of claim 2, **McLaren discloses** wherein:

the playing back video from the downloaded files includes

decoding the non-key frames and the key frames in the downloaded files to recover the video encoded therein (**Col. 1, "decoding the reproduced signals for display of the selected program at the selected new reproduction speed."**), and

presenting the recovered video (**Col. 1, "decoding the reproduced signals for display of the selected program at the selected new reproduction speed."**); and

the playing back video from the downloaded key frames includes

decoding the downloaded key frames to recover the video encoded therein (**Col. 8, "However these tables must be recovered from the**

disk and be stored in an active memory prior to program replay in order to facilitate "trick-play" reproduction."), and

presenting the recovered video (**Col. 8, "During "trick-play" reproduction the disk replay transducer jumps successively between I frames in a sequence derived from the jump-to tables. For example, at seven times forward speed the transducer is directed to jump "over" seven intervening I frames and reproduce only the eighth I frame."**).

Claim 5, the method of claim 1, **McLaren discloses** further comprising:

prior to the downloading the files, receiving a playlist from the network, the playlist listing network addresses where the files are accessible (**Col. 4, "look-up tables, or LUTs may be employed which list the file byte offsets for each GOP in each play speed stream."**),

wherein the downloading the files includes downloading the files from the network addresses listed in the playlist (**Col. 7, lines 11-13 and Col. 8, lines 35-44**).

Claim 6, the method of claim 1, **McLaren discloses** wherein the creating the trick play index includes, for each downloaded file (**Col. 3, lines 45-46 "Determine the current byte offset, and hence the current frame in the current file."**):

determining an offset from a beginning of the file where a key frame in the file begins (**Col. 3, lines 49-52 "Step 2 is complicated by the fact that, for MPEG streams, the entry points into a new stream are limited to those points where a sequence.sub.-- header exists, which is typically at an I frame at the beginning of a group of pictures (GOP)."**);

determining a size of the key frame (**Col. 3, lines 31-36 "Each trick-play and normal-play streams should comprise relatively uniform, short group of pictures (GOP) having a size of, for example, half a second. This GOP size yields a worst case visual continuity error of 0.25 seconds, i.e. time to reach the nearest I frame entry point when switching between bit streams."**); and

storing, in the index (**Col. 1, lines 58-60, 66-67 and Col. 2, lines 1-5**), the offset and the size of the key frame,

wherein the network address of the file and the offset of the key frame together represent one of the indicated network locations (**Fig. 5, el. 375 and Col. 7, lines 54-64 → the new GOP is calculated from firstly reading the current offset address and locate GOP current offset table**).

Claim 7, the method of claim 1, **McLaren discloses** wherein:

the downloaded files, and correspondingly the key frames interspersed therein, are associated with successive time codes (**Col. 6, lines 10-20 and 31-35 → "For example, a user may, in the interest of continuity of**

entertainment, opt to always join the new image stream 1/2 or 1 second prior to the departure point in the first stream, in this way "program" image continuity may be sustained.");

the playing back video from the downloaded files includes playing back the video in a forward direction of increasing time codes (**Col. 2, lines 50-51**); and

the playing back in response to the trick play request includes playing back the downloaded key frames in a rewind direction of decreasing time codes (**Col. 2, lines 51-52**).

Claim 8, the method of claim 7, **McLaren discloses** further comprising:

receiving a subsequent trick play request and, in response thereto (**Col. 8, lines 27-32 → A further test is performed is user desires a new program, therefore, a subsequent trick play is requested.**),

downloading further key frames from their indicated network addresses (**Fig. 5, el. 550 and Col. 8, lines 10-11**); and

playing back video from the downloaded further key frames in the forward direction of increasing time codes (**Col. 2, lines 50-51**).

Claim 9, the method of claim 1, **McLaren discloses** further comprising:

between the creating and the receiving, uploading the trick play index to a network address (**Col. 5, lines 55-67 "To initiate switching between streams**

the current GOP is determined from the current file offset by searching through the current play speed look-up table to find the GOP start point which corresponds to the current file offset (see Table 2). Once the current GOP is known, the new GOP, gop.sub.-- new, may be calculated from the old GOP, gop.sub.-- old, using equations 1 and 2, and the following parameters, speed.sub.-- new, speed.sub.-- old, gop.sub.-- size and first.sub.-- gop.sub.-- size;"); and

between the receiving and the downloading the key frames, downloading the trick play index from its network address (**Col. 6, lines 1-9 "Having calculated the new GOP, the look-up table appropriate to the new speed is searched to find the file offset which corresponds to the new GOP. The new stream may then be played starting at this new offset point. The relative simplicity of this system results in efficient switching between different streams. However, this method is based on real time calculation of the new GOP with the assumption that the streams contain GOPs which produce constant numbers of displayed frames (denoted by gop.sub.-- size).**").

Claim 10, an apparatus is analyzed with respect to claim 1.

Claim 13, the apparatus of claim 10 is analyzed with respect to claim 5.

Claim 14, the apparatus of claim 10 is analyzed with respect to claim 6.

Claim 15, the apparatus of claim 10 is analyzed with respect to claim 7.

Claim 16, the apparatus of claim 10 is analyzed with respect to claim 8.

Claim 19, a non-transitory computer readable medium is analyzed with respect to claim 1.

Claim 22, the computer program product of claim 19 is analyzed with respect to claim 5.

Claim 23, the computer program product of claim 19 is analyzed with respect to claim 6.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103 which forms the basis for all obviousness rejections set forth in this Office action:

> A patent for a claimed invention may not be obtained, notwithstanding that the claimed invention is not identically disclosed as set forth in section 102 of this title, if the differences between the claimed invention and the prior art are such that the claimed invention as a whole would have been obvious before the effective filing date of the claimed invention to a person having ordinary skill in the art to which the claimed invention pertains. Patentability shall not be negated by the manner in which the invention was made.

5. Claims 2-3, 11-12 and 20-21 are rejected under 35 U.S.C. 103 as being unpatentable over McLaren et al. (6,064,794) in view of Kelly et al. (US 2001/0036355)

Claim 2, the method of claim 1, **Mclaren discloses** wherein the encoded video comprises successive encoded video frames (**Col. 8, "These look up tables may contain I frame track addresses which enable the disk replay transducer to jump successively between I frames in order to generate the desired "trick-play" reproduction speed."**),

Mclaren does not disclose non-key frames each encoded based on one or more other video frames of the encoded video; and the key frames interspersed among the non-key frames, each of the key frames encoded independent of other video frames

of the encoded video, and thereby capable of being decoded independent of the other video frames.

Kelly discloses wherein the encoded video comprises successive encoded video frames, including:

non-key frames each encoded based on one or more other video frames of the encoded video (**Par. [0021], "a number of frames which are intra-coded without reference to any other frame of the sequence"**); and

the key frames interspersed among the non-key frames, each of the key frames encoded independent of other video frames of the encoded video, and thereby capable of being decoded independent of the other video frames (**Par. [0021], "number of frames which are inter-coded with reference to at least one further frame of the sequence"**).

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify McLaren's method by offering trick mode has to select only those certain "I" or "Intra" frames which are independent for display, and requires analysis of the digital stream either at the time of playback or in advance, as taught by Kelly, in order to yield a predictable result such as to decoded independently (Kelly: **Par. [0021] and [0069]**).

Claim 3, the method of claim 2, **McLaren in view of Kelly discloses** wherein the key frames include intra-coded I-frames (Kelly: **Par. [0069], "Therefore a group of pictures GOP will typically comprise: an intra-coded "I" frame, which is coded**

without reference to other pictures;") and the non-key frames include inter-coded P-frames (Kelly: **Par. [0069], "two or three "P" (predictive) coded pictures which are coded using motion vectors based on a preceding I frame"**).

Claim 11, the apparatus of claim 10 is analyzed with respect to claim 2.

Claim 12, the apparatus of claim 10 is analyzed with respect to claim 3.

Claim 20, the computer program product of claim 19 is analyzed with respect to claim 2.

Claim 21, the computer program product of claim 19 is analyzed with respect to claim 3.

6. Claims 17-18 and 24-25 are rejected under 35 U.S.C. 103 as being unpatentable over McLaren et al. (6,064,794) in view of Nelson (US 2007/0178933)

Claim 17, the apparatus of claim 10, **Mclaren discloses** further comprising:

a communication system to communicate with the communication network (**Fig. 4 and Col. 6, "A system employing the various inventive digital video source selection methods is depicted in FIG. 4."**);

a user interface system (**Fig. 4, el. 500 and Col. 6, "An interface unit 500, provides a control communication stream 551 between the user's apparatus and a digital video source 10. Interface unit 500 also decodes a compressed digital video signal 511, derived from source 10 to produce audio and video signals which are coupled to display device 1000."**).

Mclaren does not disclose housing.

Nelson discloses a housing to house the processor system, the communication system, and the user the interface system (**Fig. 1, el. 11 and Par. [0043], "Typically, phone 10 also includes an audio interface including a microphone and a speaker, radio transceiver circuitry, an antenna, a battery, and a microprocessor system including associated software and data memory for radio communication, all contained within housing 11."**).

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify McLaren's apparatus by having housing, as taught by Nelson, in order to yield a predictable result such as to improve the appearance of a mobile phone.

Claim 18, the apparatus of claim 17, **McLaren in view of Nelson discloses** wherein:

the communication system includes a wireless communication system (Nelson: **Par. [0043], "FIG. 1 illustrates a wireless communications device in the form of a mobile phone 10."**); and

the housing includes a mobile hand-held housing to house the processor system, the communication system, the user interface system, and a battery (Nelson: **Fig. 1, el. 11 and Par. [0043]**).

Claim 24, the computer program product of claim 19 is analyzed with respect to claim 17.

Claim 25, the computer program product of claim 19 is analyzed with respect to claim 18.

Conclusion

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to JEN-SHI HUANG whose telephone number is (571)272-3872. The examiner can normally be reached on Monday - Friday, alt Friday, 8:00am to 5:30pm EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Benjamin Bruckart can be reached on (571) 272-3982. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/J. H./

Examiner, Art Unit 2423

/MARK D FEATHERSTONE/

Primary Examiner, Art Unit 2423

An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP

Saamer Akhshabi
College of Computing
Georgia Institute of Technology
sakhshab@cc.gatech.edu

Ali C. Begen
Video and Content Platforms Research and Advanced Development
Cisco Systems
abegen@cisco.com

Constantine Dovrolis
College of Computing
Georgia Institute of Technology
dovrolis@cc.gatech.edu

ABSTRACT

Adaptive (video) streaming over HTTP is gradually being adopted, as it offers significant advantages in terms of both user-perceived quality and resource utilization for content and network service providers. In this paper, we focus on the rate-adaptation mechanisms of adaptive streaming and experimentally evaluate two major commercial players (Smooth Streaming, Netflix) and one open source player (OSMF). Our experiments cover three important operating conditions. First, how does an adaptive video player react to either persistent or short-term changes in the underlying network available bandwidth? Can the player quickly converge to the maximum sustainable bitrate? Second, what happens when two adaptive video players compete for available bandwidth in the bottleneck link? Can they share the resources in a stable and fair manner? And third, how does adaptive streaming perform with live content? Is the player able to sustain a short playback delay? We identify major differences between the three players, and significant inefficiencies in each of them.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms

Performance, Measurement, Algorithms

Keywords

Experimental evaluation, adaptive streaming, rate-adaptation algorithm, video streaming over HTTP

1. INTRODUCTION

Video has long been viewed as the "next killer application". Over the last 20 years, the various instances of packet video have been thought of as demanding applications that would never work satisfactorily over best-effort IP networks. That pessimistic view actually led to the creation of novel network architectures and QoS mechanisms, which were not deployed in a large-scale, though. Eventually, over the last three to four years video-based applications, and video streaming in particular, have become utterly popular generating more than half of the aggregate Internet traffic. Perhaps, surprisingly though, video streaming today runs over IP without any specialized support from the network. This has become possible through the gradual development of highly efficient video compression methods, the penetration of broadband access technologies, and the development of *adaptive video players* that can compensate for the unpredictability of the underlying network through sophisticated rate-adaptation, playback buffering, and error recovery and concealment methods.

Another conventional wisdom has been that video streaming would never work well over TCP, due to the throughput variations caused by TCP's congestion control and the potentially large retransmission delays. As a consequence, most of the earlier video streaming research has assumed that the underlying transport protocol is UDP (or RTP over UDP), which considerably simplifies the design and modeling of adaptive streaming applications. In practice, however, two points became clear in the last few years. First, TCP's congestion control mechanisms and reliability requirement do not necessarily hurt the performance of video streaming, especially if the video player is able to adapt to large throughput variations. Second, the use of TCP, and of HTTP over TCP in particular, greatly simplifies the traversal of firewalls and NATs.

The first wave of HTTP-based video streaming applications used the simple *progressive download* method, in which a TCP connection simply transfers the entire movie file as quickly as possible. The shortcomings of that approach are many, however. One major issue is that all clients receive the same encoding of the video, despite the large variations in the underlying available bandwidth both across different clients and across time for the same client. This has recently led to the development of a new wave of HTTP-based streaming applications that we refer to as *adaptive streaming over HTTP* (For a general overview of video streaming protocols and adaptive streaming, refer to [2]). Several recent players, such as Microsoft's Smooth

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MMSys'11, February 23–25, 2011, San Jose, California, USA.
Copyright 2011 ACM 978-1-4503-0517-4/11/02 ...$10.00.

Streaming, Adobe OSMF, as well as the players developed or used by Netflix, Move Networks and others, use this approach. In adaptive streaming, the server maintains multiple profiles of the same video, encoded in different bitrates and quality levels. Further, the video object is partitioned in *fragments*, typically a few seconds long. A player can then request different fragments at different encoding bitrates, depending on the underlying network conditions. Notice that it is the player that decides what bitrate to request for any fragment, improving server-side scalability. Another benefit of this approach is that the player can control its playback buffer size by dynamically adjusting the rate at which new fragments are requested.

Adaptive streaming over HTTP is a new technology. It is not yet clear whether the existing commercial players perform well, especially under dynamic network conditions. Further, the complex interactions between TCP's congestion control and the application's rate-adaptation mechanisms create a "nested double feedback loop" - the dynamics of such interacting control systems can be notoriously complex and hard to predict. As a first step towards understanding and improving such video streaming mechanisms, this paper experimentally evaluates two commercial adaptive video players over HTTP (Microsoft's Smooth Streaming and the player used by Netflix) and one open source player (Adobe OSMF). Our experiments cover three important operating conditions. First, how does an adaptive video player react to either persistent or short-term changes in the underlying network available bandwidth? Can the player quickly converge to the maximum sustainable bitrate? Second, what happens when two adaptive video players compete for available bandwidth in the bottleneck link? Can they share that resource in a stable and fair manner? And third, how does adaptive streaming perform with live content? Is the player able to sustain a short playback delay? We identify major differences between the three players, and significant inefficiencies in each of them.

1.1 Related Work

Even though there is extensive previous work on rate-adaptive video streaming over UDP, transport of rate-adaptive video streaming over TCP, and HTTP in particular, presents unique challenges and has not been studied in depth in the past. A good overview of multi-bitrate video streaming over HTTP was given by Zambelli [17], focusing on Microsoft's IIS Smooth Streaming. Adobe has provided an overview of HTTP Dynamic Streaming on the Adobe Flash platform [1]. Cicco et al. [3] experimentally investigated the performance of the Akamai HD Network for Dynamic Streaming for Flash over HTTP. They studied how the player reacted to abrupt changes in the available bandwidth and how it shared the network bottleneck with a greedy TCP flow. Kuschnig et al. [9] evaluated and compared three server-side rate-control algorithms for adaptive TCP streaming of H.264/SVC video. The same authors have proposed a receiver-driven transport mechanism that uses multiple HTTP streams and different priorities for certain parts of the media stream [10]. The end-result is to reduce throughput fluctuations, and thus, improve video streaming over TCP. Tullimas et al. [15] also proposed a receiver-driven TCP-based method for video streaming over the Internet, called *MultiTCP*, aimed at providing resilience against short-term bandwidth fluctuations and controlling the sending rate by using multiple TCP connections. Hsiao et al. [8] proposed a method called Receiver-based Delay Control (RDC) to avoid congestion by delaying TCP ACK generation at the receiver based on notifications from routers. Wang et al. [16] developed discrete-time Markov models to investigate the performance of TCP for both live and stored media streaming. Their models provide guidelines indicating the circumstances under which TCP streaming leads to satisfactory performance. For instance, they show that TCP provides good streaming performance when the achievable TCP throughput is roughly twice the media bitrate, with only a few seconds of startup delay. Goel et al. [7] showed that the latency at the application layer, which occurs as a result of throughput-optimized TCP implementations, could be minimized by dynamically tuning TCP's send buffer. They developed an adaptive buffer-size tuning technique that aimed at reducing this latency. Feng et. al [5] proposed and evaluated a priority-based technique for the delivery of compressed prerecorded video streams across best-effort networks. This technique uses a multi-level priority queue in conjunction with a delivery window to smooth the video frame rate, while allowing it to adapt to changing network conditions. Prangl et al. [13] proposed and evaluated a TCP-based perceptual QoS improvement mechanism. Their approach is based on media content adaptation (transcoding), applied at the application layer at the server. Deshpande [4] proposed an approach that allowed the player to employ single or multiple concurrent HTTP connections to receive streaming media and switch between the connections dynamically.

1.2 Paper Outline

In Section 2, we describe our experimental approach, the various tests we perform for each player, and the metrics we focus on. Sections 3, 4 and 5 focus on the Smooth Streaming, Netflix, and OSMF players, respectively. Section 6 focuses on the competition effects that take place when two adaptive players share the same bottleneck. Section 7 focuses on live video using the Smooth Streaming player. We summarize what we learn for each player and conclude the paper in Section 8.

2. METHODOLOGY AND METRICS

In this section, we give an overview of our experimental methodology and describe the metrics we focus on. The host that runs the various video players also runs a packet sniffer (Wireshark [12]) and a network emulator (DummyNet [14]). Wireshark allows us to capture and analyze offline the traffic from and to the HTTP server. DummyNet allows us to control the *downstream available bandwidth* (also referred to as *avail-bw*) that our host can receive. That host is connected to the Georgia Tech campus network through a Fast Ethernet interface. When we do not limit the avail-bw using DummyNet, the video players always select the highest rate streams; thus, when DummyNet limits the avail-bw to relatively low bitrates (1-5 Mbps) we expect that it is also the downstream path's end-to-end bottleneck.

In the following, we study various throughput-related metrics:

1. The *avail-bw* refers to the bitrate of the bottleneck that we emulate using DummyNet. The TCP connections that transfer video and audio streams cannot exceed (collectively)

that bitrate at any point in time.
2. The *2-sec connection throughput* refers to the download throughput of a TCP connection that carries video or audio traffic, measured over the last two seconds.
3. The *running average of a connection's throughput* refers to a running average of the 2-sec connection throughput measurements. If $A(t_i)$ is the 2-sec connection throughput in the i'th time interval, the running average of the connection throughput is:

$$\hat{A}(t) = \begin{cases} \delta\hat{A}(t_{i-1}) + (1-\delta)A(t_i) & i > 0 \\ A(t_0) & i = 0 \end{cases}$$

In the experiments, we use $\delta = 0.8$.
4. The *(audio or video) fragment throughput* refers to the download throughput for a particular fragment, i.e., the size of that fragment divided by the corresponding download duration. Note that, if a fragment is downloaded in every two seconds, the fragment throughput can be much higher than the 2-sec connection throughput in the same time interval (because the connection can be idle during part of that time interval). As will be shown later, some video players estimate the avail-bw using fragment throughput measurements.

We also estimate the *playback buffer size* at the player (measured in seconds), separately for audio and video. We can accurately estimate the playback buffer size for players that provide a timestamp (an offset value that indicates the location of the fragment in the stream) in their HTTP fragment requests. Suppose that two successive, say video, requests are sent at times t_1 and t_2 ($t_1 < t_2$) with timestamps $t_1^{'}$ and $t_2^{'}$ ($t_1^{'} < t_2^{'}$), respectively (all times measured in seconds). The playback buffer duration in seconds for video at time t_2 can be then estimated as:

$$B(t_2) = [B(t_1) - (t_2 - t_1) + (t_2^{'} - t_1^{'})]^+$$

where $[x]^+$ denotes the maximum of x and 0. This method works accurately because, as will be clear in the following sections, the player requests are not pipelined: a request for a new fragment is sent only after the previous fragment has been fully received.

We test each player under the same set of avail-bw conditions and variations. In the first round of tests, we examine the behavior of a player when the avail-bw is not limited by DummyNet; this "blue-sky" test allows us to observe the player's start-up and steady-state behavior - in the same experiments we also observe what happens when the user skips to a future point in the video clip. In the second round of tests, we apply persistent avail-bw variations (both increases and decreases) that last for tens of seconds. Such variations are common in practice when the cross traffic in the path's bottleneck varies significantly due to arriving or departing traffic from other users. A good player should react to such variations by decreasing or increasing the requested bitrate. In the third round of tests, we apply positive and negative spikes in the path's avail-bw that last for just few seconds - such variations are common in 802.11 WLANs for instance. For such short-term drops, the player should be able to maintain a constant requested bitrate using its playback buffer. For short-term avail-bw increases, the player could be conservative and stay at its current rate to avoid unnecessary bitrate variations. Due to space constraints, we do not show results from all these experiments for each player; we select only those results that are more interesting and provide new insight.

All experiments were performed on a Windows Vista Home Premium version 6.0.6002 laptop with an Intel(R) Core(TM)2 Duo P8400 2.26 GHz processor, 3.00 GB physical memory, and an ATI Radeon Graphics Processor (0x5C4) with 512 MB dedicated memory.

3. MICROSOFT SMOOTH STREAMING

In the following experiments, we use Microsoft Silverlight Version 4.0.50524.0. In a Smooth Streaming manifest file, the server declares the available audio and video bitrates and the resolution for each content (among other information). The manifest file also contains the duration of every audio and video fragment. After the player has received the manifest file, it generates successive HTTP requests for audio and video fragments. Each HTTP request from the player contains the name of the content, the requested bitrate, and a timestamp that points to the beginning of the corresponding fragment. This timestamp is determined using the per-fragment information provided in the manifest. The following is an example of a Smooth Streaming HTTP request.

```
GET (..)/BigBuckBunny720p.ism/
QualityLevels(2040000)/Fragments(video=400000000)
HTTP/1.1
```

In this example, the requested bitrate is 2.04 Mbps and the fragment timestamp is 40 s.

The Smooth Streaming player maintains two TCP connections with the server. At any point in time, one of the two connections is used for transferring audio and the other for video fragments. Under certain conditions, however, the player switches the audio and video streams between the two connections - it is not clear to us when/how the player takes this decision. This way, although at any point in time one connection is transferring video fragments, over the course of streaming, both connections get the chance to transfer video fragments. The benefit of such switching is that neither of the connections would stay idle for a long time, keeping the server from falling back to slow-start. Moreover, the two connections would maintain a large congestion window.

Sometimes the player aborts a TCP connection and opens a new one - this probably happens when the former connection provides very low throughput. Also, when the user jumps to a different point in the stream, the player aborts the existing TCP connections, if they are not idle, and opens new connections to request the appropriate fragments. At that point, the contents of the playback buffer are flushed.

In the following experiments we watch a sample video clip ("Big Buck Bunny") provided by Microsoft at the IIS Web site:

```
http://www.iis.net/media/experiencesmoothstreaming
```

The manifest file declares eight video encoding bitrates between 0.35 Mbps and 2.75 Mbps and one audio encoding bitrate (64 Kbps). We represent an encoding bitrate of r Mbps as P_r, (e.g., $P_{2.75}$). Each video fragment (except the

last) has the same duration: τ=2 s. The audio fragments are approximately of the same duration.

3.1 Behavior under Unrestricted avail-bw

Figure 1 shows the various throughput metrics, considering only the video stream, in a typical experiment without restricting the avail-bw using DummyNet. t=0 corresponds to the time when the Wireshark capture starts. Note that the player starts from the lowest encoding bitrate and it quickly, within the first 5-10 seconds, climbs to the highest encoding bitrate. As the per-fragment throughput measurements indicate, the highest encoding bitrate ($P_{2.75}$) is significantly lower than the avail-bw in the end-to-end path. The player upshifts to the highest encoding profile from the lowest one in four transitions. In other words, it seems that the player avoids large jumps in the requested bitrate (more than two successive bitrates) - the goal is probably to avoid annoying the user with sudden quality transitions, providing a dynamic but smooth watching experience.

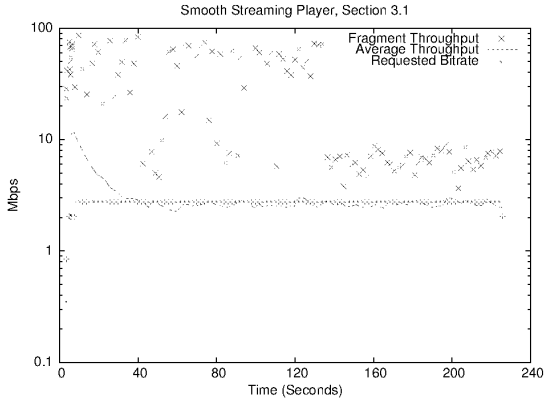


Figure 1: Per-fragment throughput, average TCP throughput and the requested bitrate for video traffic under unrestricted avail-bw conditions. Playback starts at around t=5 s, almost 2 s after the user clicked PLAY.

Another important observation is that during the initial time period, the player asks for video fragments much more frequently than once every τ seconds. Further analysis of the per-fragment interarrivals and download times shows that the player operates in one of two states: `Buffering` and `Steady-State`. In the former, the player requests a new fragment as soon as the previous fragment was downloaded. Note that the player does *not* use HTTP pipelining - it does not request a fragment if the previous fragment has not been fully received. In `Steady-State`, on the other hand, the player requests a new fragment either τ seconds after the previous fragment was requested (if it took less than τ seconds to download that fragment) or as soon as the previous fragment was received (otherwise). In other words, in the `Buffering` state the player aims to maximize its fragment request rate so that it can build up a target playback buffer as soon as possible. In `Steady-State`, the player aims to maintain a constant playback buffer, requesting one fragment every τ seconds (recall that each fragment corresponds to τ seconds of content). We estimated the target video playback buffer size, as described in Section 2, to be about 30 seconds. The time it takes to reach `Steady-State` depends on the avail-bw - as the avail-bw increases, it takes less time to accumulate the 30-second playback buffer. We have consistently observed that the player does not sacrifice quality, requesting low-bitrate encodings, to fill up its playback buffer sooner. Another interesting observation is that the player does not request a video bitrate whose frame resolution (as declared at the manifest file) is larger than the resolution of the display window.

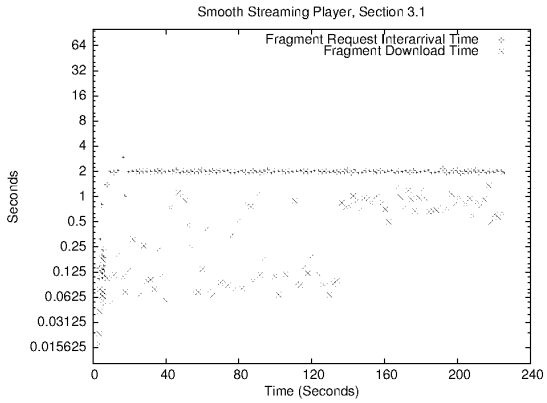


Figure 2: Interarrival and download times of video fragments under unrestricted avail-bw conditions. Each fragment is two seconds long.

3.2 Behavior of the Audio Stream

Audio fragments are of the same duration with video fragments, at least in the movies we experimented with. Even though audio fragments are much smaller in bytes than video fragments, the Smooth Streaming player does not attempt to accumulate a larger audio playback buffer than the corresponding video buffer (around 30 s). Also, when the avail-bw drops, the player does not try to request audio fragments more frequently than video fragments (it would be able to do so). Overall, it appears that the Smooth Streaming player attempts to keep the audio and video stream download processes as much in sync as possible.

3.3 Behavior under Persistent avail-bw Variations

In this section, we summarize a number of experiments in which the avail-bw goes through four significant and persistent transitions, as shown in Figure 3. First, note that, as expected, the per-fragment throughput is never higher than the avail-bw. Instead, the per-fragment throughput tracks quite well the avail-bw variations for most of the time; part of the avail-bw, however, is consumed by audio fragments and, more importantly, TCP throughput can vary significantly after packet loss events.

We next focus on the requested video bitrate as the avail-bw changes. Initially, the avail-bw is 5 Mbps and the player requests the $P_{2.04}$ profile because it is constrained by the resolution of the display window (if we were watching the video in full-screen mode, the player would request the highest $P_{2.75}$ profile). The playback buffer (shown in Figure

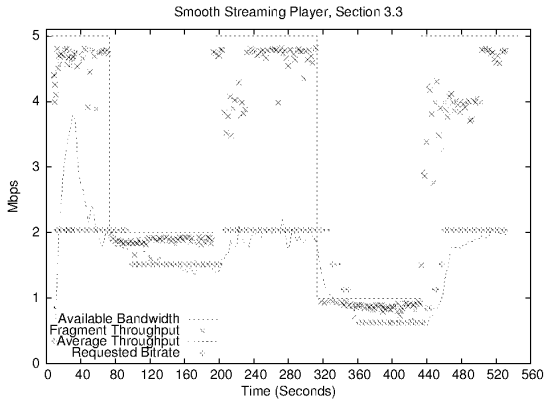


Figure 3: Per-fragment throughput, average TCP throughput and the requested bitrate for the video traffic under persistent avail-bw variations. Playback starts at around t=10 s, almost 3 s after the user clicked PLAY.

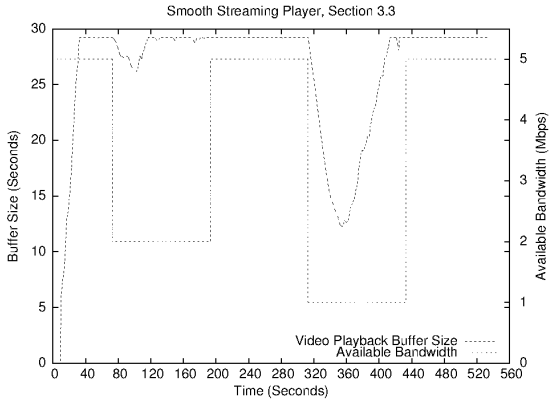


Figure 4: Video playback buffer size in seconds under persistent avail-bw variations.

4) has reached its 30 s target by t=40 s and the player is in `Steady-State`.

At time t=73 s, the avail-bw is dropped to 2 Mbps - that is not sufficient for the $P_{2.04}$ encoding because we also need some capacity for the audio traffic and for various header overheads. The player reacts by switching to the next lower profile ($P_{1.52}$) but after some significant delay (almost 25 seconds). During that time period, the playback buffer has decreased by only 3 seconds (the decrease is not large because the avail-bw is just barely less than the cumulative requested traffic). The large reaction delay indicates that the player does *not* react to avail-bw changes based on the latest per-fragment throughput measurements. Instead, it averages those per-fragment measurements over a longer time period so that it acts based on a smoother estimate of the avail-bw variations. The playback buffer size returns to its 30 s target after the player has switched to the $P_{1.52}$ profile.

The avail-bw increase at t=193 s is quickly followed by an appropriate increase in the requested encoding bitrate. Again, the switching delay indicates that the Smooth Streaming player is conservative, preferring to estimate reliably the avail-bw (using several per-fragment throughput measurements) instead of acting opportunistically based on the latest fragment throughput measurement.

The avail-bw decrease at t=303 s is even larger (from 5 Mbps to 1 Mbps) and the player reacts by adjusting the requested bitrate in four transitions. The requested bitrates are not always successive. After those transitions, the request bitrate converges to an appropriate value $P_{0.63}$, much less than the avail-bw. It is interesting that the player could have settled at the next higher bitrate ($P_{0.84}$) - in that case, the aggregate throughput (including the audio stream) would be 0.94 Mbps. That is too close to the avail-bw (1 Mbps), however. This implies that Smooth Streaming is conservative: it prefers to maintain a safety margin between the avail-bw and its requested bitrate. We think that this is wise, given that the video bitrate can vary significantly around its *nominal encoding value* due to the variable bitrate (VBR) nature of video compression.

Another interesting observation is that the player avoids large transitions in the requested bitrate - such quality transitions can be annoying to the viewer. Also, the upward transitions are faster than the downward transitions - still, however, it can take several tens of seconds until the player has switched to the highest sustainable bitrate.

3.4 Behavior under Short-term avail-bw Variations

In this section, we summarize a number of experiments in which the avail-bw goes through positive or negative "spikes" that last for only few seconds, as shown in Figures 5 and 7. The spikes last for 2 s, 5 s and 10 s, respectively. Such short-term avail-bw variations are common in practice, especially in 802.11 WLAN networks. We think that a good adaptive player should be able to compensate for such spikes using its playback buffer, without causing short-term rate adaptations that can be annoying to the user.

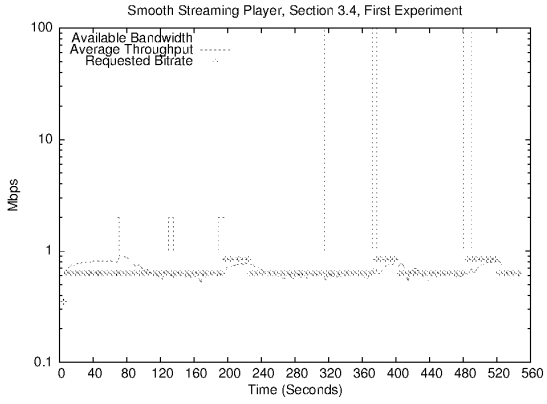


Figure 5: Average TCP throughput and the requested bitrate for the video traffic under positive avail-bw spikes. Playback starts at around t=7 s, almost 4 s after the user clicked PLAY.

Figure 5 shows the case of positive spikes. Here, we repeat the three spikes twice, each time with a different

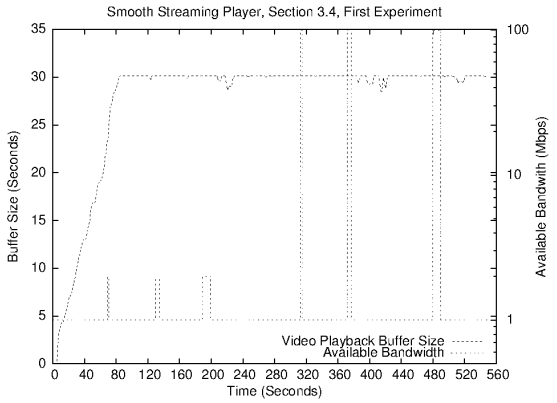


Figure 6: Video playback buffer size in seconds under positive avail-bw spikes.

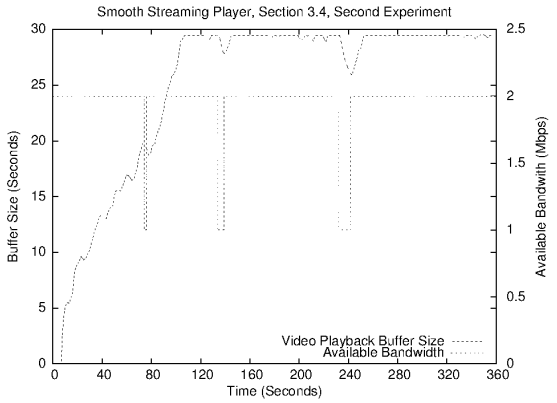


Figure 8: Video playback buffer size in seconds under negative avail-bw spikes.

increase magnitude. The Smooth Streaming player ignores the 2-second spikes and the smaller 5-second spike. On the other hand, it reacts to the 10-second spikes by increasing the requesting video bitrate. Unfortunately, it does so too late (sometimes after the end of the spike) and for too long (almost till 40 s after the end of the spike). During the time periods that the requested bitrate is higher than the avail-bw, the playback buffer size obviously shrinks, making the player more vulnerable to freeze events (See Figure 6). This experiment confirms that the player reacts, not to the latest fragment download throughput, but to a smoothed estimate of those measurements that can be unrelated to the current avail-bw conditions.

Figures 7 and 8 show similar results in the case of negative spikes. Here, the spikes reduce the avail-bw from 2 Mbps to 1 Mbps. The player reacts to all three spikes, even the spike that lasts for only 2 s. Unfortunately, the player reacts too late and for too long: it requests a lower bitrate after the end of each negative spike and it stays at that lower bitrate long for 40-80 s. During those periods, the user would unnecessarily experience a lower video quality.

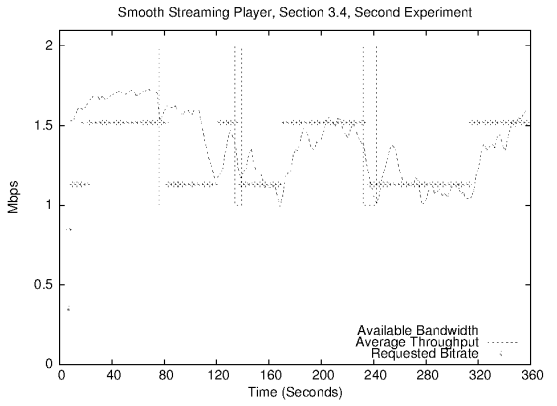


Figure 7: Average TCP throughput and the requested bitrate for the video traffic under negative avail-bw spikes. Playback starts at around t=9 s, almost 3 s after the user clicked PLAY.

4. NETFLIX PLAYER

The Netflix player uses Microsoft's Silverlight for media representation, but a different rate-adaptation logic. The Netflix player also maintains two TCP connections with the server, and it manages these TCP connections similarly with the Smooth Streaming player. As will become clear, however, the Netflix player does not send audio and video fragment requests at the same pace. Also, the format of the manifest file and requests are different. Further, most of the initial communication between the player and server, including the transfer of the manifest file, is done over SSL. We decrypted the manifest file using a Firefox plugin utility called Tamper Data that accesses the corresponding private key in Firefox. Video and audio fragments are delivered in wmv and wma formats, respectively. An example of a Netflix fragment request follows:

```
GET /sa2/946/1876632946.wmv
/range/2212059-2252058?token=1283923056
_d6f6112068075f1fb60cc48eab59ea55&random
=1799513140 HTTP/1.1
```

Netflix requests do not correspond to a certain time duration of audio or video. Instead, each request specifies a range of bytes in a particular encoding profile. Thus, we cannot estimate the playback buffer size as described in Section 2. We can only approximate that buffer size assuming that the actual encoding rate for each fragment is equal to the corresponding nominal bitrate for that fragment (e.g., a range of 8 Mb at the $P_{1.09}$ encoding profile corresponds to 8 seconds worth of video) - obviously this is only an approximation but it gives us a rough estimate of the playback buffer size.

After the user clicks the PLAY button, the player starts by performing some TCP transfers, probably to measure the capacity of the underlying path. Then it starts buffering audio and video fragments, but without starting the playback yet. The playback starts either after a certain number of seconds, or when the buffer size reaches a target point. If that buffer is depleted at some point, the Netflix player prefers to stop the playback, showing a message that

the player is adjusting to a slower connection. The playback resumes when the buffer size reaches a target point.

In the following experiments we watch the movie "Mary and Max". The manifest file provides five video encoding bitrates between 500 Kbps and 3.8 Mbps and two audio encoding bitrates (64 Kbps and 128 Kbps).

4.1 Behavior under Unrestricted avail-bw

Figure 9 shows the various throughput metrics, considering only the video stream, in a typical experiment without using DummyNet to restrict the avail-bw. The interarrival of video fragment requests and the download times for each video fragment are shown in Figure 10. In this experiment, the playback started about 13 s after the user clicked PLAY. The playback delay can be much larger depending on the initial avail-bw (even up to few minutes). During this interval, several security checks are also performed before the player starts buffering and the playback begins [11]. For the first few fragments, the player starts from the lowest encoding bitrate and requests a number of fragments from all the available bitrates. Then, the player stays at that highest bitrate for the duration of the experiment.

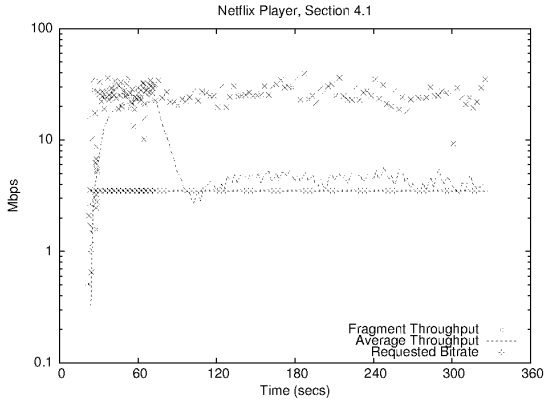


Figure 9: Per-fragment throughput, average TCP throughput and the requested bitrate for video traffic under unrestricted avail-bw conditions. Playback starts at around t=24 s, almost 16 s after the user clicked PLAY.

During the first 55 s of streaming, until t=75 s, the player is clearly in the `Buffering` state: it requests a new video fragment right after the previous fragment has been downloaded. The achieved TCP throughput in this path is about 30 Mbps, allowing the player to quickly accumulate a large playback buffer. We estimated the size of the playback buffer at the end of the `Buffering` state (t=75 s) at about 300 s worth of video - this is an order of magnitude larger than the playback buffer size we observed for Smooth Streaming.

When the player switches to `Steady-State`, video fragments are requested almost every three seconds, with significant variation, however.

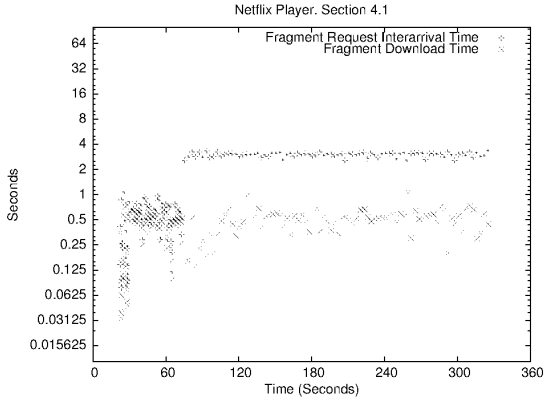


Figure 10: Interarrival and download times of video fragments under unrestricted avail-bw conditions.

4.2 Behavior of the Audio Stream

Audio fragments in the Netflix player are significantly larger than the ones in Smooth Streaming. Specifically, an audio fragment is typically 30 s long. Thus, after the player has reached `Steady-State`, a new audio fragment is requested every 30 s. Further, it appears that this player does not attempt to keep the audio and video stream download processes in sync; it can be that the audio playback buffer size is significantly larger than the video playback buffer size.

4.3 Behavior under Persistent avail-bw Variations

Figure 11 shows the various throughput-related metrics in the case of persistent avail-bw variations. As in the experiment with unrestricted avail-bw, the player first requests few fragments at all possible encodings. Within the first 40 s it converges to the highest sustainable bitrate ($P_{1.50}$) for that avail-bw (2 Mbps). It should be noted that in this experiment the player never leaves the `Buffering` state (based on analysis of the video fragment request interarrivals).

When the avail-bw drops to 1 Mbps, the player reacts within about 20 s, which implies that its avail-bw estimator is based on a smoothed version of the underlying per-fragment throughput, as opposed to the instantaneous and latest such measurement. It is interesting that the selected profile at that phase ($P_{1.00}$) is not sustainable, however, because it is exactly equal to the avail-bw (some avail-bw is consumed by audio traffic and other header overheads). Thus, the playback buffer size slowly decreases, forcing the player between 320 and 400 s to occasionally switch to the next lower bitrate. This observation implies that the Netflix player prefers to utilize a certain high bitrate even when the avail-bw is insufficient, as long as the player has accumulated more than a certain playback buffer size. We make the same observation from 450 to 500 s. During that interval, the player switches to a profile ($P_{2.60}$) that is much higher than the avail-bw (2 Mbps). The player can do this, without causing any problems, because it has accumulated a sufficiently large playback buffer size at that point.

In summary, it appears that the Netflix player is more

aggressive than Smooth Streaming, trying to deliver the highest possible encoding rate even when the latter is more than the avail-bw, as long as the playback buffer size is sufficiently large.

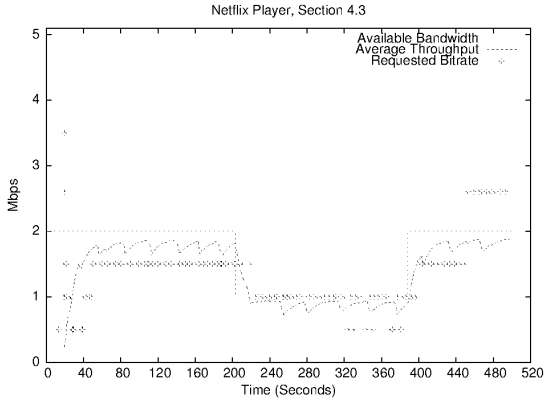


Figure 11: Average TCP throughput and the requested bitrate for the video traffic under persistent avail-bw variations. Playback starts at around t=20 s, almost 13 s after the user clicked PLAY.

4.4 Behavior under Short-term avail-bw Variations

Figure 12 shows how the Netflix player reacts to positive avail-bw spikes, while Figure 13 shows how it reacts to negative avail-bw spikes. We cannot compare directly the results of these experiments to the corresponding experiments with Smooth Streaming, because the video encoding profiles are different between the movies in these two experiments. As in the case of persistent avail-bw variations, we observe that the Netflix player is rather aggressive, reacting to large increases in avail-bw even if they are short-lived. As opposed to Smooth Streaming, which often reacts too late and for too long, Netflix reacts faster, while the spike is still present, even though the reaction can still last much longer after the spike.

On the other hand, in the experiment with negative avail-bw spikes, the Netflix player does not switch to a lower bitrate. It prefers to compensate for the lack of avail-bw using the large playback buffer size that it has previously accumulated.

5. ADOBE OSMF

We have repeated the same set of tests with Adobe's sample OSMF player, using Flash version 10.1.102.64 with player version WIN 10.1.102.64 and OSMF library version 1.0. In the following experiments, we watch a movie trailer ("Freeway") provided by Akamai's HD-video demo Web site for Adobe HTTP Dynamic Streaming:

```
http://wwwns.akamai.com/hdnetwork/demo/flash/zeri/
```

Note that the player used in this Web site was not built specifically to showcase HTTP Dynamic Streaming.

The manifest file declares eight encoding bitrates for this trailer between 0.25 Mbps and 3.00 Mbps. In this

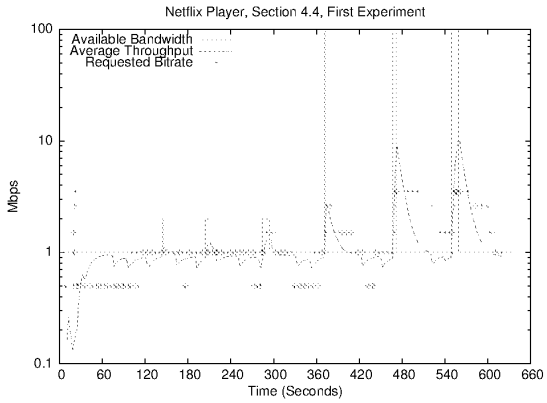


Figure 12: Average TCP throughput and the requested bitrate for the video traffic under positive avail-bw spikes. Playback starts at around t=20 s, almost 16 s after the user clicked PLAY.

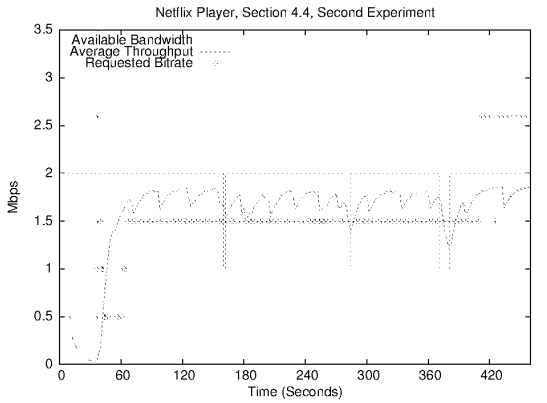


Figure 13: Average TCP throughput and the requested bitrate for the video traffic under negative avail-bw spikes. Playback starts at around t=37 s, almost 31 s after the user clicked PLAY.

player, each server file (F4F format) contains one segment of the movie. A segment can contain multiple fragments. The duration of each fragment is determined by the server. The HTTP requests that the player generates include a fragment number instead of a timestamp or a byte range. An example of an OSMF HTTP request follows:

```
GET /content/inoutedit-mbr
/inoutedit_h264_3000Seg1-Frag5 HTTP/1.1
```

Note that the requested bitrate is shown in the request (3000 Kbps) together with the requested segment and fragment numbers. The player maintains one TCP connection with the server and receives all fragments through this connection. The player might shut down the connection and open a new one if the user jumps to a different point in the stream and the connection is not idle. According to information provided by the player itself, the target playback buffer seems to be less than 10 seconds.

Figure 14 shows that initially the player requests one fragment at the lowest available profile and then quickly climbs to the largest possible one. But, it does not converge to that profile and continues switching between profiles occasionally. When the avail-bw is dropped to 2 Mbps at t=139 s, the player fails to converge to the highest sustainable profile ($P_{1.7}$). Instead, it keeps switching between profiles, often using the lowest and highest ones ($P_{0.25}$ and $P_{3.00}$). The user observes several dropped frames and freeze events, which is an indication of a depleted playback buffer.

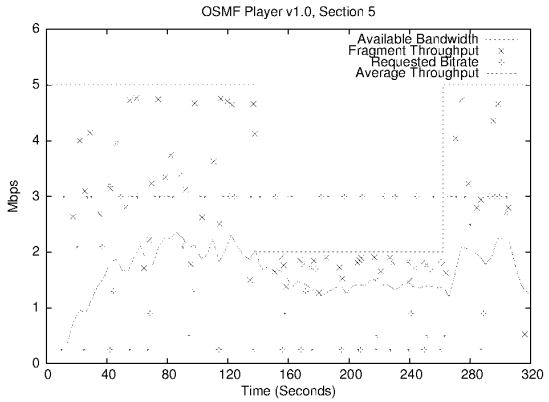


Figure 14: Per-fragment throughput, the requested bitrate for the video traffic and average TCP throughput under persistent avail-bw variations. Playback starts at around t=13 s, almost 3 s after the user clicked PLAY.

We have also conducted the same set of experiments with the latest version of the OSMF player obtained from the following Web site.

`http://sourceforge.net/adobe/osmf/home/`

Figure 15 shows the various throughput related metrics in an experiment with the OSMF player version 1.5 under persistent avail-bw variations. We see a very similar issue here. The player makes similar problematic rate switchings and gets into oscillation.

To summarize, we have observed that the OSMF player fails to converge to a sustainable bitrate especially when the avail-bw is smaller than or very close to the highest available bitrate of the media. Instead, it usually oscillates between the lowest and highest bitrates. The default rate-adaptation algorithm seems to be tuned for short variations in the avail-bw. We do not describe here the rest of the experiments we performed with it, because they simply confirm that the default rate-adaptation algorithm deployed in the OSMF player does not function properly under our test scenarios.

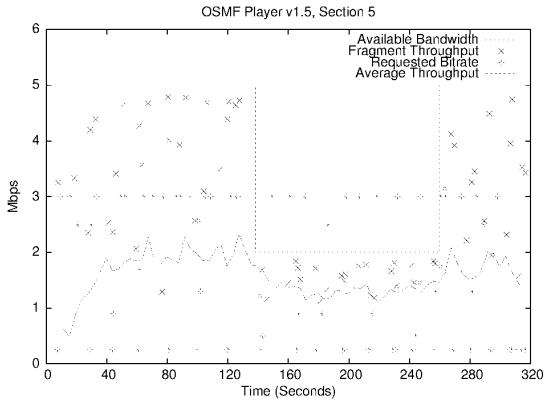


Figure 15: Per-fragment throughput, the requested bitrate for the video traffic and average TCP throughput under persistent avail-bw variations. Playback starts at around t=11 s, almost 4 s after the user clicked PLAY.

6. TWO COMPETING PLAYERS

Suppose that two adaptive HTTP streaming players share the same bottleneck. This can happen, for instance, when people in the same house watch two different movies - in that case the shared bottleneck is probably the residential broadband access link. Another example of such competition is when a large number of users watch the same live event, say a football game. In that case the shared bottleneck may be an edge network link. There are many questions in this context. Can the players share the avail-bw in a stable manner, without experiencing oscillatory bitrate transitions? Can they share the avail-bw in a fair manner? How does the number of competing streams affect stability and fairness? How do different adaptive players compete with each other? And how does a player compete with TCP bulk transfers (including progressive video downloads)? In this section, we only "touch the surface" of these issues, considering a simple scenario in which two identical players (Smooth Streaming) compete at a bottleneck in which the avail-bw varies between 1-4 Mbps. The idea is that, if we observe significant problems even in this simple scenario, we should also expect similar issues in more complex scenarios.

In the following, we present results from two experiments. It should be noted that such experiments are fundamentally non-reproducible: there is always some stochasticity in the way players share the bottleneck's avail-bw. However, our observations, at a qualitative level, are consistent across several similar experiments.

Figure 16 shows the avail-bw variations in the first experiment, together with the requested bitrates from the two players. The second player starts about one minute after the first one. Until that point, the first player was using the highest profile ($P_{2.75}$). After the second player starts, the two players could have shared the 4 Mbps bottleneck by switching to $P_{1.52}$, however, they do not. Instead, the second player oscillates between lower profiles. When the avail-bw drops to 3 Mbps or 2 Mbps, the oscillations continue for both players. The only stable period during this experiment is when the avail-bw is limited to 1 Mbps: in that case

both players switch to the lowest profile $P_{0.35}$ simply because there is no other bitrate that is sustainable for both players. Interestingly, when the avail-bw increases to 4 Mbps the two players start oscillating in a synchronized manner: when they both switch to $P_{2.04}$ the shared bottleneck is congested. It seems that both players observe congestion at the same time, and they react in the same manner lowering their requested bitrate at about the same time.

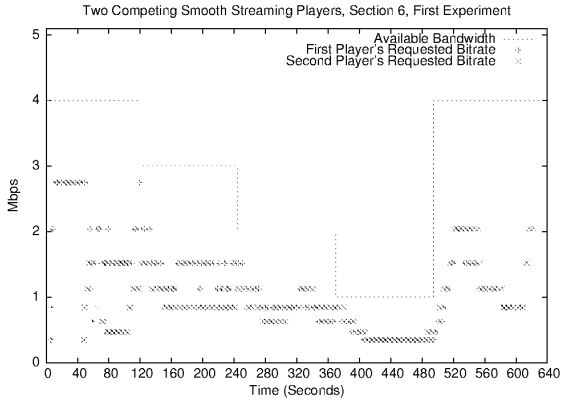


Figure 16: Two Smooth Streaming players compete for avail-bw. The players start the playback at around t=7 s and t=57 s, respectively.

The previous experiment reveals some interesting points about Smooth Streaming. First, it seems that the avail-bw estimation method in that player considers only time periods in which fragments are actually downloaded - there is probably no estimation when the player's connections are idle. So, if two players X and Y share the same bottleneck, and Y is idle while X downloads some fragments, X can overestimate the avail-bw. Second, it appears that the Smooth player does not use randomization in the rate-adaptation logic. Previous studies have shown that a small degree of randomization was often sufficient to avoid synchronization and oscillations [6].

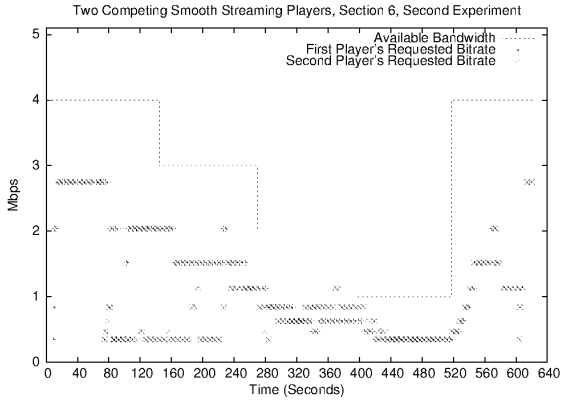


Figure 17: Two Smooth Streaming players compete for avail-bw. The players start the playback at around t=12 s and t=77 s, respectively.

Figure 17 shows the results for another run. Here, the second player stays at the lowest possible bitrate for about 150 s after it starts streaming, while the first player uses the remaining avail-bw with the highest sustainable bitrate. This is clearly a very unfair way to share the bottleneck link. It should be noted that this unfairness is *unrelated* to TCP's well-known unfairness towards connections with large round-trip times (RTT). In this experiment, both connections have the same RTT. The unfairness here is not generated by TCP's congestion control, but by the offered load that each application (video player) requests. The second player estimates the avail-bw to be much lower, and it does not even try to increase its requested bitrate. If it had done so, it would likely be able to obtain a higher bitrate forcing the first player to a lower bitrate. It appears, however, that the Smooth player's rate-adaptation algorithm does not include such bandwidth-sharing objectives.

7. SMOOTH LIVE STREAMING

We are also interested in the similarities and differences between live and on-demand adaptive video streaming. What is the playback delay in the case of live streaming? Does the player react differently to avail-bw variations when it streams live content? And how does the player react when the playback buffer becomes empty? Does it skip fragments so that it maintains a small playback delay, or does it increase the playback delay aiming to show all fragments? We explored these questions with the Smooth Streaming player. In the following experiments, we used the live video feed from the Home Shopping Network (HSN) web site: `http://www.hsn.com/hsn-tv_at-4915_xa.aspx?nolnav=1`.

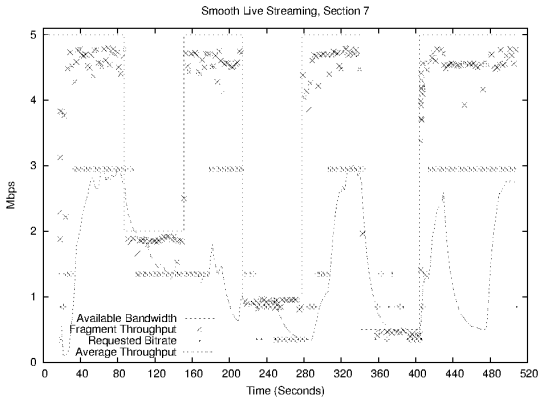


Figure 18: Per-fragment throughput, the requested bitrate for the video traffic and average TCP throughput for live video streaming. Playback starts at around t=20 s.

Figure 18 shows the various throughput metrics and the requested video bitrate, while Figure 19 shows the estimated video playback buffer size (in seconds). A first important difference with on-demand streaming is that the initial playback buffer size is about 8 s; significantly lower than the typical playback buffer sizes we observed in on-demand Smooth Streaming sessions. By the way, even though this playback delay may sound too large for live content,

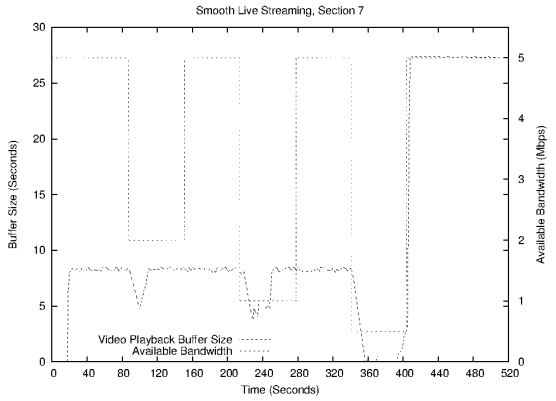


Figure 19: Video playback buffer size in seconds for live video streaming.

note that even cable TV networks usually enforce a similar delay (referred to as "profanity delay") to avoid broadcasting inappropriate scenes.

A second important point, which is not shown in the previous graphs but it can be observed by the timestamps of the HTTP requests, is that the player initially requests fragments that correspond to about 8 s in the past. This way, it can start displaying the video without having to wait for a long initial playback delay; of course, what the user watches then happened at least 8 s ago. As in the case of on-demand streaming, the initial fragment request rate (while the player is in the `Buffering` state) is higher, requesting a new fragment as soon as the last fragment is received.

Other than the previous two points, it appears that the Smooth Streaming player does not react to avail-bw variations any different with live content than with on-demand content. Note that the persistent avail-bw decreases and increases are followed by similar bitrate adjustments as in Section 3.

Another interesting point is what happens when the playback buffer becomes empty. This happened in this experiment at around t=360 s, when the avail-bw was decreased to 500 Kbps. During that event the playback buffer remained practically empty for about 40 s. The player still receives some fragments during that period but they would not increase the playback buffer size by more than a fragment. The player was late to switch to a sufficiently lower bitrate, and so several fragments were requested at bitrates ($P_{1.40}$ and $P_{0.80}$) that were higher than the avail-bw. The end-result was that those fragments took too long to download, the buffer became depleted, and the playback stalled for about 27 s.

Arguably, it is reasonable to expect for live streaming that the player could skip some fragments that take too long to download, jumping to a later point in the video stream. The Smooth Streaming implementation for this particular Web site does not do so, however. It appears that the player aims to show every single fragment. Consequently, the playback delay can increase, gradually staying more and more behind the live broadcast. Indeed, in this case after the avail-bw increased to 5 Mbps, the playback buffer size increased to almost 27 s, which is comparable to the typical playback delay of on-demand content using Smooth Streaming.

8. CONCLUSIONS

We conducted an experimental evaluation of three commercial adaptive HTTP streaming players, focusing on how they react to persistent and short-term avail-bw variations. Here, we summarize our findings for each player and conclude the paper.

The Smooth Streaming player is quite effective under unrestricted avail-bw as well as under persistent avail-bw variations. It quickly converges to the highest sustainable bitrate, while it accumulates at the same time a large playback buffer requesting new fragments (sequentially) at the highest possible bitrate. This player is rather conservative in its bitrate switching decisions. First, it estimates the avail-bw by smoothing the per-fragment throughput measurements, introducing significant delays in the rate-adaptation logic. Second, it avoids large bitrate changes that could be annoying to the viewer. On the negative side, the Smooth Streaming player reacts to short-term avail-bw spikes too late and for too long, causing either sudden drops in the playback buffer or unnecessary bitrate reductions. Further, our experiments with two competing Smooth Streaming players indicate that the rate-adaptation logic is not able to avoid oscillations, and it does not aim to reduce unfairness in bandwidth sharing. The Live Smooth Streaming player behaves similarly, except that the playback buffer is initially shorter and the player starts requesting fragments from the recent past.

The Netflix player is similar to Smooth Streaming (they both use Silverlight for the media representation). However, we observed that the former showed some important differences in its rate-adaptation behavior, becoming more aggressive than the latter and aiming to provide the highest possible video quality, even at the expense of additional bitrate changes. Specifically, the Netflix player accumulates a very large buffer (up to few minutes), it downloads large chunks of audio in advance of the video stream, and it occasionally switches to higher bitrates than the avail-bw as long as the playback buffer is almost full. It shares, however, the previous shortcomings of Smooth Streaming.

The OSMF player often fails to converge to an appropriate bitrate even after the avail-bw has stabilized. This player has been made available so that developers will customize the code including the rate-adaptation algorithm for HTTP Dynamic Streaming for their use case. We do not summarize any other experiment here.

Overall, it is clear that the existing adaptive HTTP streaming players are still at their infancy. The technology is new and it is still not clear how to design an effective rate-adaptation logic for a complex and demanding application (video streaming) that has to function on top of a complex transport protocol (TCP). The interactions between these two feedback loops (rate-adaptation logic at the application layer and TCP congestion control at the transport layer) are not yet understood well. In future work, we plan to focus on these issues and design improved rate-adaptation mechanisms.

9. REFERENCES

[1] Adobe. HTTP Dynamic Streaming on the Adobe Flash Platform. *Adobe Systems Incorporated*, 2010. http://www.adobe.com/products/httpdynamicstreaming/pdfs/httpdynamicstreaming_wp_ue.pdf.

[2] A. C. Begen, T. Akgul, and M. Baugher. Watching video over the Web, part I: streaming protocols. *To appear in IEEE Internet Comput.*, 2011.

[3] L. De Cicco and S. Mascolo. An Experimental Investigation of the Akamai Adaptive Video Streaming. In *Proc. of USAB WIMA*, 2010.

[4] S. Deshpande. Adaptive timeline aware client controlled HTTP streaming. In *Proc. of SPIE*, 2009.

[5] W. Feng, M. Liu, B. Krishnaswami, and A. Prabhudev. A priority-based technique for the best-effort delivery of stored video. In *Proc. of MMCN*, 1999.

[6] R. Gao, C. Dovrolis, and E. Zegura. Avoiding oscillations due to intelligent route control systems. In *Proc. of IEEE INFOCOM*, 2006.

[7] A. Goel, C. Krasic, and J. Walpole. Low-latency adaptive streaming over TCP. *ACM TOMCCAP*, 4(3):1–20, 2008.

[8] P.-H. Hsiao, H. T. Kung, and K.-S. Tan. Video over TCP with receiver-based delay control. In *Proc. of ACM NOSSDAV*, 2001.

[9] R. Kuschnig, I. Kofler, and H. Hellwagner. An evaluation of TCP-based rate-control algorithms for adaptive Internet streaming of H.264/SVC. In *Proc. of ACM MMSys*, 2010.

[10] R. Kuschnig, I. Kofler, and H. Hellwagner. Improving Internet video streamilng performance by parallel TCP-based request-response streams. In *Proc. of IEEE CCNC*, 2010.

[11] Pomelo LLC. Analysis of Netflix's security framework for 'Watch Instantly' service. *Pomelo, LLC Tech Memo*, 2009. http://pomelollc.files.wordpress.com/2009/04/pomelo-tech-report-netflix.pdf.

[12] A. Orebaugh, G. Ramirez, J. Burke, and J. Beale. *Wireshark and Ethereal network protocol analyzer toolkit*. Syngress Media Inc, 2007.

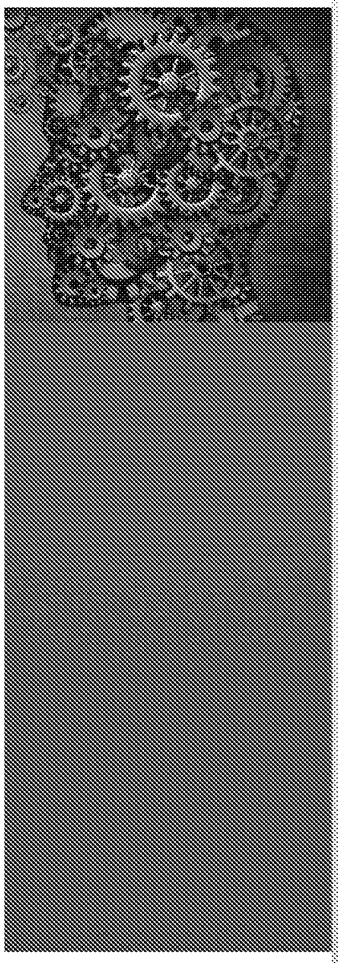
[13] M. Prangl, I. Kofler, and H. Hellwagner. Towards QoS improvements of TCP-based media delivery. In *Proc. of ICNS*, 2008.

[14] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *SIGCOMM CCR*, 27(1):31–41, 1997.

[15] S. Tullimas, T. Nguyen, R. Edgecomb, and S.-C. Cheung. Multimedia streaming using multiple TCP connections. *ACM TOMCCAP*, 4(2):1–20, 2008.

[16] B. Wang, J. Kurose, P. Shenoy, and D. Towsley. Multimedia streaming via TCP: An analytic performance study. *ACM TOMCCAP*, 4(2):1–22, 2008.

[17] A. Zambelli. IIS smooth streaming technical overview. *Microsoft Corporation*, 2009. http://download.microsoft.com/download/4/2/4/4247C3AA-7105-4764-A8F9-321CB6C765EB/IIS_Smooth_Streaming_Technical_Overview.pdf.



Method for the encoding of a compressed video sequence derived from the same video sequence compressed at a different bit rate without loss of data

An IP.com Prior Art Database Technical Disclosure

Authors et. al.: Disclosed Anonymously
IP.com Number: IPCOM000008165D
IP.com Electronic Publication: May 22, 2002

IP.com is the world's leader in defensive publications. The largest and most innovative companies publish their technical disclosures into the IP.com Prior Art Database. Disclosures can be published in any language, and they are searchable in those languages online. Unique identifiers indicate documents containing chemical structures. Original disclosures that are published online also appear in The IP.com Journal. The IP.com Prior Art Database is freely available to search by patent examiners throughout the world.

Terms: Client may copy any content obtained through the site for Client's individual, non-commercial internal use only. Client agrees not to otherwise copy, change, upload, transmit, sell, publish, commercially exploit, modify, create derivative works or distribute any content available through the site.

Note: This is a PDF rendering of the actual disclosure. To access the disclosure package containing an exact copy of the publication in its original format as well as any attached files, please download the full document from IP.com at http://ip.com/IPCOM/000008165.

Method for the encoding of a compressed video sequence derived from the same video sequence compressed at a different bit rate without loss of data

Disclosed is a method for the encoding of a compressed video sequence given the same video sequence compressed at a different bit rate without loss of data (lossless). Benefits include improved functionality and improved performance.

Background

Scalable codecs and transcoding are two known methods to provide content at different bit rates or quality.

Scalable codecs generate bit streams representing individual layers such that enhancement layers can be combined with the base layer to produce video of higher (spatial, temporal, or SNR) qualities. Typically, low-resolution data is used for the compression of higher-resolution data. As a result, the higher resolution stream is coupled with the data of the lower-resolution stream (the reverse is also possible). The result is a lower coding efficiency when compared to single-layer encoding. A single layer stream cannot be extracted from the base and enhancement layers without re-encoding the video (requantization with reduced coding efficiency).

Transcoding takes a single layer video stream and generates from it another single layer stream but at a lower bit rate to meet storage or bandwidth constraints. This implies that quantized data is requantized and compared to the direct encoding of the original data at a lower rate. A loss of 1 dB typically occurs. Transcoders can be as simple as the cascade of a decoder and encoder. Alternatively, the decoding and encoding process can be integrated by reusing information, like motion vectors. The processing may be performed in the frequency domain.

General description

The disclosed method enables multiple single-layer bit streams of a video sequence to be efficiently stored or transmitted using lossless compression of a reference stream. One of the single layer streams is designated as the reference stream. The other single layer streams are losslessly encoded with respect to that reference stream to form a differentially encoded stream. With this approach, given the reference stream and a differentially encoded stream, the corresponding single layer stream that was obtained by encoding the uncompressed video data can be reconstructed without loss of data.

The disclosed method applies to applications that require a video sequence at different rates. This situation occurs in streaming servers for clients with various bandwidth capabilities. A content provider may also want to provide its content at different rates as a function of the service a client has paid for or what the content will be used for, such as high quality for editing, lower quality for preview. The content owner could be a home user who wants high-quality streams for her digital video clips for archiving and lower-quality streams for posting on the web

or for viewing on his PDA. Content with different bit rates can also be used in peer-to-peer services to match the bandwidth between peers or the computing capacity of the peer requesting the content (content with lower bit rate requires less computation for decoding).

Advantages

The disclosed method provides several advantages, including

- Improved functionality by enabling multiple single-layer bit streams at various bit rates without loss of data
- Improved performance by the use of single-layer encoding rather than scalable codecs or transcoding
- Improved use of storage by overlaying the streams rather than storing separate single-layer streams

Detailed description

The disclosed method is the encoding of a compressed video sequence derived from a compressed representation of the same video sequence but at a lower or higher quality (a lower or higher bit rate). This approach can be applied to the lossless encoding of a compressed video sequence given a compressed representation of the same video data but at a different temporal, special, or signal-to-noise ratio (SNR) resolution.

For example, a video sequence is encoded at multiple rates or quality levels. The encoding can be accomplished in a number of ways: using one encoder at different settings, using many encoders simultaneously, using a system that can simultaneously encode a sequence at many rates/quality levels, or any combination of these methods. The different encoded streams can be generated at different points in time. The encoded streams of the same sequence can then be differentially encoded. For example, *encoded_stream_1* and *encoded_stream_2* are the inputs to differential encoder, and the output, *D_stream_2,1* is the data from which *encoded_stream_2* can be reconstructed from *encoded_stream_1* (see Figure 1). The differential encoding can be applied to any two of the encoded streams, and the reference stream can be any of those two streams.

The encoder may be integrated with the differential encoder (see Figure 2). The output can then be limited to one encoded stream, which is the reference stream, and a set of differentially encoded streams. The encoders are integrated for computational efficiency or a more efficient hardware implementation.

The differential decoder has two inputs (see Figure 3), a reference stream, and a differential stream with respect to that reference stream. The output is the target stream, that is, the encoded stream that was differentially encoded. From differential stream, *D_stream_m,k,* and the reference stream, *encoded_stream_k,* the differential decoder builds *encoded_stream_m.*

In another implementation (see Figure 4), the output of the decoder is not *encoded_stream_m* but the video sequence that would have been reconstructed from *encoded_stream_m.* For an

application where the video sequence is required bit the encoded stream is not required, a more computationally or hardware efficient decoder can be built by integrating both decoders.

Spatial domain

In a simple, conceptual implementation, the differential encoder generates the reconstructed frames of stream *k* and *m*, denoted as f_k and f_m (see Figure 5). Their difference is computed and encoded as the differential stream. The encoding can be accomplished using any lossless encoding technique. The differential bit stream may include information other than the actual frame difference. For example, information like motion vectors and quantization values may be required at the decoder to generate *encoded_stream_m* from f_m. The differential decoder reconstructs the target stream, *encoded_stream_m,* from the reference stream, *encoded_stream_k,* and the differential stream, *D_stream_m,k* (see Figure 6).

The video encoder and the differential encoder are separate processes, but both encoding processes can be integrated. The integration level can be function of the applications for which they are used and the actual outputs that are needed. For example, the encoding of the sequences and the differential encoding can be integrated so that no decoding is performed in the differential encoder because the data the differential encoder uses is generated by the single layer encoding process.

Transform domain

Standard video codecs process the data in a transformed domain because they result in higher compression ratios. For example, in MPEG standards, the data (other than motion vector and stream organization data) is encoded in the DCT domain. Generating the differential data in the same DCT domain is an alternative that may lead to a more efficient codec. The following explanation describes how the differential encoder and decoder can be implemented with quantized coefficient data, in the DCT domain, or any other domain, including the spatial domain.

The value L_k designates the values of the quantized coefficient levels in the reference stream. L_m designates the level values of the same quantized coefficients but in the target stream (that is, the stream to be encoded differentially). First, a prediction of L_m is computed and is noted as *Lm.* Next, their difference, noted as $D_{m,k}$, is encoded without loss in the differential bit stream. At the decoder (see Figure 8), given L_k and $D_{m,k}$, L_m can be computed.

Example of predictors

A very simple predictor (see Figure 9) uses the quantization parameters of the reference stream. The predictor first generates the quantized coefficient values from the quantized coefficient levels. The values are then requantized, this time with the quantizer parameters used for the target stream. The output of the second quantizer is the predicted level values of the quantized coefficients of the target stream.

Another predictor that is more complex (see Figure 10) performs additional processing on the dequantized coefficient values, V_k, before they are requantized with the target stream parameters to improve the accuracy of the predicted quantized levels, *Lm*.

Two examples (see Figure11 and Figure 12) illustrate the processing that can be done on the dequantized coefficients values V_k. In both cases, this processing is targeted to coefficient values of data representing a displaced frame difference (motion compensated prediction error). In these figures, F_k and F_m respectively denote the coefficients of the predicted frame for the reference stream and the target stream.

The dequantized coefficient values *Vk* (see Figure 11) are added to the difference between *Fk*, the coefficients of the predicted frame from the reference sequence, and F_m, the coefficients of the predicted frame from the target sequence, to form V_m. This operation is done to take into account that the frame used for prediction is different in both streams. The predicted coefficients V_m are then quantized to obtain the predicted quantized coefficient levels L_m, as described previously.

In another example (see Figure 12), the coefficient values of the reconstructed frame of the reference stream, which we denote as F_k, are computed. Algorithms, like post-processing algorithms (such as deblocking and deringing) can then be applied to F_k. These algorithms attenuate the distortion and artifacts caused by the compression. They can contribute to improve the prediction of the quantized coefficient values of the target stream when the target stream is encoded at higher quality than the reference stream. The coefficient values of the motion compensated frame of the target stream, F_m, are subtracted from *F'k* to generate the predicted coefficient values, V_m, followed by the quantization. A similar process can be done for intra-coded frames. In this example, the addition of F_k and the subtraction of F_m are removed.

These various predictors can be combined and selectively applied to frames, blocks of data in a frame, or even single coefficients.

A complete implementation

Several implementations of the disclosed method can be tested. The results presented below use the predictor of Figure 11 for non-zero coefficients in non-intra predicted blocks of the reference stream. The simpler predictor of Figure 9 is used for nonzero coefficients in intra-blocks. Zero coefficients are simply predicted as zero.

The syntax elements needed to correctly decode the differential stream and the entropy coding of the differential stream are not contained in the method. Any suitable syntax description and entropy coding can be used with the disclosed method.

Benefits

The disclosed method can be used to efficiently store a video sequence encoded at different rates. For example, a 352 x 240, 30 fps video sequence may be encoded at 5 different rates from

750 Kbits/sec to 2 Mbits/sec. The disclosed method saves 30% of storage space when compared to independently storing the 5 streams (see Figure 13).

This savings in storage comes at the expense of extra computation to reconstruct a stream that was differentially encoded.

The disclosed method can also be used as an alternative to transcoding. The sequences can be differentially encoded at the rates that are required. This process uses more storage than transcoding does, but eventually a stream is generated at the required bit rate with higher quality (see Figure 14).

The disclosed method can also be used in a streaming scenario in which both the reference stream and a differential bit stream are simultaneously transmitted. This solution enables a client with lower bandwidth or computational power to only receive the lower quality signal, while the client with higher bandwidth and computational power can receive the higher quality signal.

When the reference and differential bit stream are recombined, the single layer stream has the coding efficiency of the single layer encoder used to generate it. The loss of transcoding does not occur and neither does the loss incurred by scalable codecs, which couple the encoding of the base and enhancement layers.

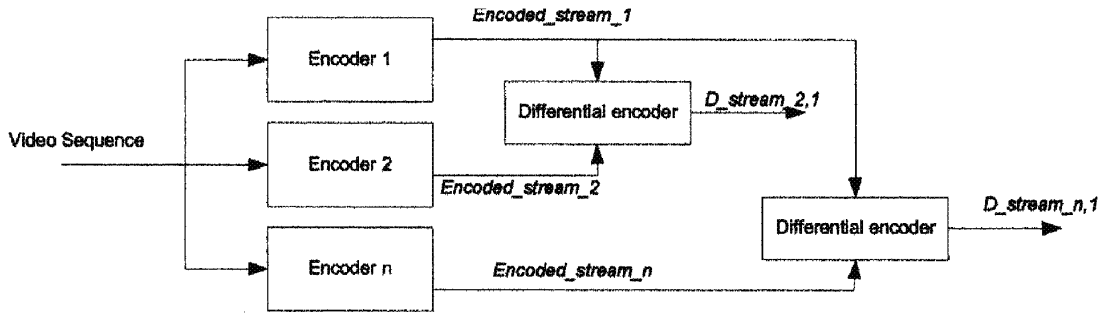


Fig. 1

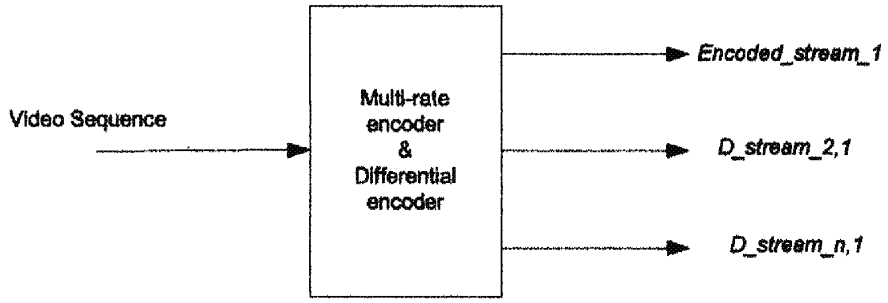


Fig. 2

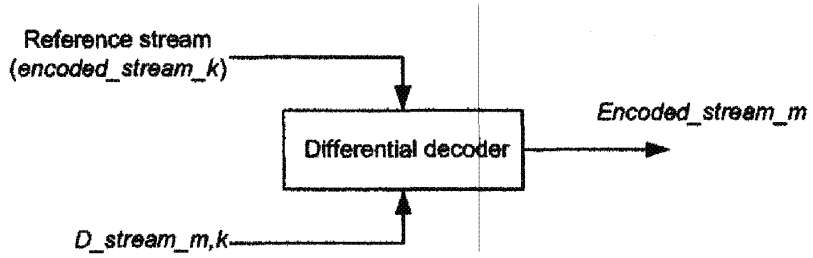


Fig. 3

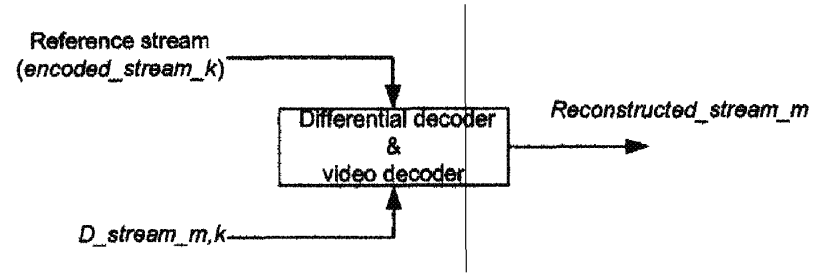


Fig. 4

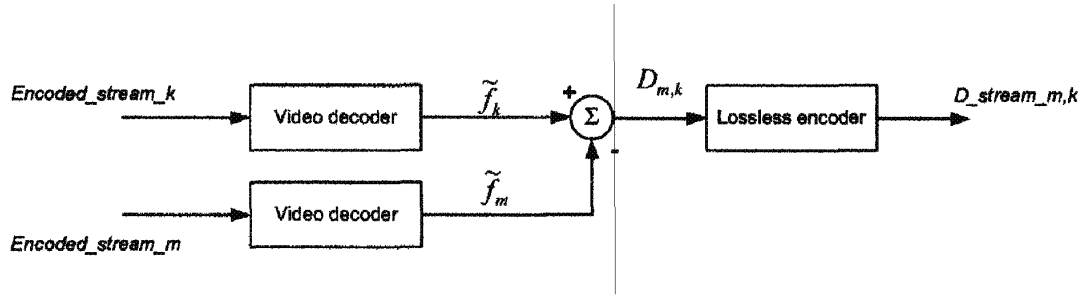


Fig. 5

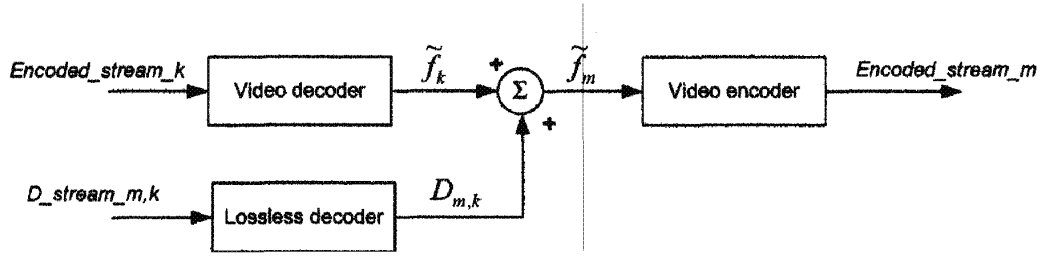


Fig. 6

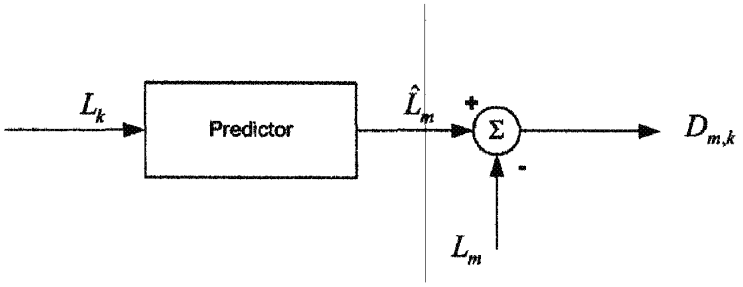


Fig. 7

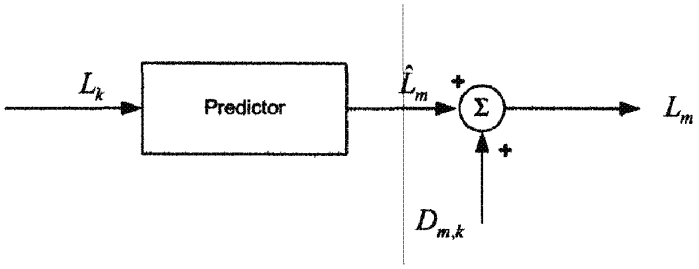


Fig. 8

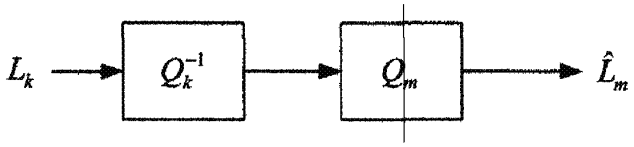


Fig. 9

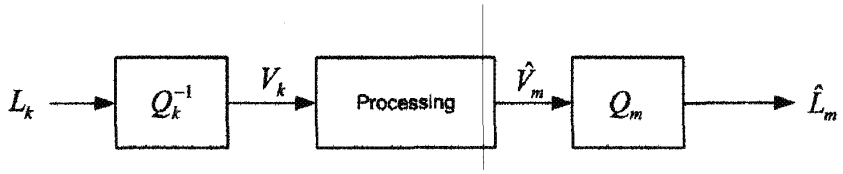


Fig. 10

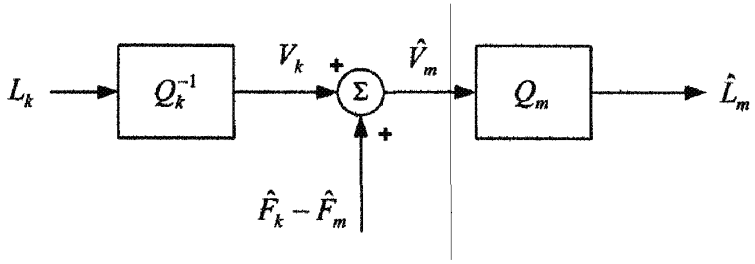


Fig. 11

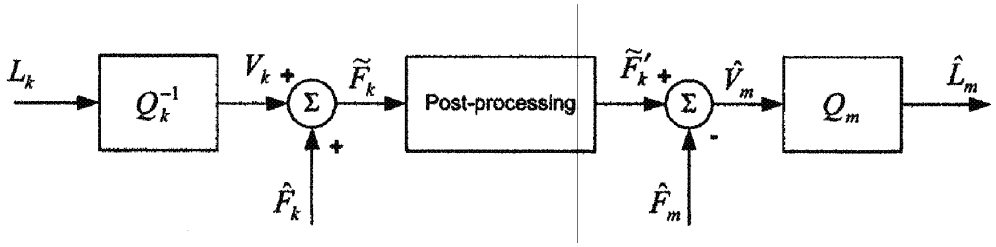


Fig. 12

Quant. scale value	**Bit rate of differential and reference streams (bits/sec)** Ref Q=9	Ref Q=11	Ref Q=13	**Bit rate of independently coded streams**
7	1,035,357	1,200 31	1,330,536	1,957,853
9	1,492,394	785,261	926,537	1,492,394
11	670,705	1,119,81	625,793	1,193,814
13	626,852	53615	992,703	992,703
17	435,082	586,10	481,483	741 ,834
Total bit rate	**4,260,390**	**4,301,652**	**4,363,052**	**6,378,599**
Savings	33.2%	32.60),	31.6%	

Fig. 13

	1. One reference stream + + differential streams	2. Multiple Single-layer streams	3.Transcoding
Storage space	30% less than 2	Highest	Lowest
Quality	Lossless	Lossless	Lossy (~1 dB)

Fig. 14

Disclosed anonymously

Blu-ray Disc

Make a donation to Wikipedia and give the gift of knowledge!

From Wikipedia, the free encyclopedia

Blu-ray Disc (also known as **Blu-ray** or **BD**) is an optical disc storage medium. Its main uses are high-definition video and data storage. The disc has the same physical dimensions as standard DVDs and CDs.

The name *Blu-ray Disc* is derived from the blue laser (violet-colored) used to read and write this type of disc. Because of the beam's shorter wavelength (405 nanometres), substantially more data can be stored on a Blu-ray Disc than on the DVD format, which uses a red (650 nm) laser. A two-layer Blu-ray Disc can store 50 gigabytes, almost six times the capacity of a two-layer DVD, or ten times that of a single-layer DVD.

During the format war over high-definition optical discs, Blu-ray Disc competed with the HD DVD format. On February 19, 2008, Toshiba—the main company supporting HD DVD—announced that it would no longer develop, manufacture, and market HD DVD players and recorders,[2] leading almost all other HD DVD companies to follow suit, effectively ending the format war.

Blu-ray Disc was developed by the Blu-ray Disc Association, a group representing makers of consumer electronics, computer hardware, and motion pictures. As of October 29, 2008, more than 1010 Blu-ray Disc titles have been released in the United States and more than 550 Blu-ray Disc titles have been released in Japan.[3][4][5] There are expected to be over 1300 Blu-ray Disc titles released in the United States by the end of 2008.[6]

Blu-ray Disc



Media type	High-density optical disc
Encoding	MPEG-2, H.264/MPEG-4 AVC, and VC-1
Capacity	25 GB (single layer) 50 GB (dual layer)
Read mechanism	405 nm laser: 1× at 36 Mbit/s 2× at 72 Mbit/s 4× at 144 Mbit/s 6× at 216 Mbit/s[1] 8× at 288 Mbit/s 12× at 432 Mbit/s
Developed by	Blu-ray Disc Association
Usage	Data storage, High-definition video High-definition audio PlayStation 3 games

Contents

- 1 History
  - 1.1 Origins
  - 1.2 Blu-ray Disc format finalized
  - 1.3 Launch and sales developments
  - 1.4 Competition from HD DVD
  - 1.5 End of the format war
  - 1.6 Restart of Format War
- 2 Technical specifications
  - 2.1 Laser and optics
  - 2.2 Hard-coating technology
  - 2.3 Recording speed
- 3 Software standards
  - 3.1 Codecs
  - 3.2 Java software support
  - 3.3 Region codes
  - 3.4 Digital rights management
- 4 Player profiles
- 5 Backward compatibility
- 6 Ongoing development
- 7 Variants
  - 7.1 Mini Blu-ray Disc
  - 7.2 BD9/BD5 Blu-ray Disc
  - 7.3 AVCREC
  - 7.4 Blu-ray Disc recordable
- 8 See also
  - 8.1 Alternative disc technologies
- 9 References
- 10 External links

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1914
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

History

In 1998, commercial HDTV sets began to appear in the consumer market; however, there was no commonly accepted, inexpensive way to record or play HD content. In fact, there was no medium with the storage required to accommodate HD codecs, except JVC's Digital VHS and Sony's HDCAM.[7] Nevertheless, it was well known that using lasers with shorter wavelengths would enable optical storage with higher density. When Shuji Nakamura invented practical blue laser diodes, it was a sensation, although a lengthy patent lawsuit delayed commercial introduction.[8]

Origins

Philips and Sony started two projects applying the new diodes: UDO (Ultra Density Optical) and DVR Blue (together with Pioneer), a format of rewritable discs which would eventually become Blu-ray Disc (more specifically, BD-RE).[9] The core technologies of the formats are essentially similar.

The first DVR Blue prototypes were unveiled at the CEATEC exhibition in October 2000.[10] Because the Blu-ray Disc standard places the data recording layer close to the surface of the disc, early discs were susceptible to contamination and scratches and had to be enclosed in plastic cartridges for protection. In February 2002, the project was officially announced as Blu-ray,[11] and the Blu-ray Disc Association was founded by the nine initial members.

The first consumer devices were in stores on April 10, 2003. This device was the Sony BDZ-S77; a BD-RE recorder that was made available only in Japan. The recommended price was US$3800;[12] however, there was no standard for pre-recorded video and no movies were released for this player. The Blu-ray Disc standard was still years away as a newer, more secure DRM system was needed before Hollywood studios would accept it, not wanting to repeat the failure of the Content Scramble System used on DVDs.

Blu-ray Disc format finalized

The Blu-ray Disc physical specifications were finished in 2004.[13] In January 2005, TDK announced that they had developed a hard coating polymer for Blu-ray Discs.[14] The cartridges, no longer necessary, were scrapped. The BD-ROM specifications were finalized in early 2006.[15] AACS LA, a consortium founded in 2004,[16] had been developing the DRM platform that could be used to securely distribute movies to consumers. However, the final AACS standard was delayed,[17] and then delayed again when an important member of the Blu-ray Disc group voiced concerns.[18] At the request of the initial hardware manufacturers, including Toshiba, Pioneer and Samsung, an interim standard was published which did not include some features, like managed copy.[19]

Launch and sales developments

The first BD-ROM players were shipped in the middle of June 2006, though HD DVD players beat them in the race to the market by a few months.[20][21]

Optical disc authoring

- Optical disc
- Optical disc drive
- Optical disc authoring
- Authoring software
- Recording technologies
  - Recording modes
  - Packet writing

Optical media types

- Laserdisc (LD), Video Single Disc (VSD)
- Compact Disc (CD): Red Book, CD-ROM, CD-R, CD-RW, 5.1 Music Disc, SACD, PhotoCD, CD Video (CDV), Video CD (VCD), SVCD, CD+G, CD-Text, CD-ROM XA, CD-i
- GD-ROM
- MiniDisc (MD) (Hi-MD)
- DVD: DVD-R, DVD+R, DVD-R DL, DVD+R DL, DVD-RW, DVD+RW, DVD-RW DL, DVD+RW DL, DVD-RAM, DVD-D
- Ultra Density Optical (UDO)
- Universal Media Disc (UMD)
- HD DVD: HD DVD-R, HD DVD-RW
- **Blu-ray Disc** (BD): BD-R, BD-RE
- High-Definition Versatile Disc (HVD)
- High definition Versatile Multilayer Disc (HD VMD)

Standards

- Rainbow Books
- File systems
  - ISO 9660
    - Joliet
    - Rock Ridge
    - El Torito
    - Apple ISO 9660 Extensions
  - Universal Disk Format (UDF)
    - Mount Rainier

Further reading

- History of optical storage media
- High definition optical disc format war

The first Blu-ray Disc titles were released on June 20, 2006. The earliest releases used MPEG-2 video compression, the same method used on DVDs. The first releases using the newer VC-1 and AVC codecs were introduced in September 2006.[22] The first movies using dual layer discs (50 GB) were introduced in October 2006.[23] The first audio-only release was made in March 2008.[24]

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1915
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The first mass-market Blu-ray Disc rewritable drive for the PC was the BWU-100A, released by Sony on July 18, 2006. It recorded both single and dual layer BD-R as well as BD-RE discs and had a suggested retail price of US$699.



A blank rewritable Blu-ray Disc (BD-RE)

Competition from HD DVD

The DVD Forum (which was chaired by Toshiba) was deeply split over whether to develop the more expensive blue laser technology or not. In March 2002, the forum voted to approve a proposal endorsed by Warner Bros. and other motion picture studios that involved compressing HD content onto dual-layer DVD-9 discs.[25][26] In spite of this decision, however, the DVD Forum's Steering Committee announced in April that it was pursuing its own blue-laser high-definition solution. In August, Toshiba and NEC announced their competing standard Advanced Optical Disc.[27] It was finally adopted by the DVD Forum and renamed HD DVD the next year,[28] after being voted down twice by Blu-ray Disc Association members, prompting the U.S. Department of Justice to make preliminary investigations into the situation.[29][30]

HD DVD had a head start in the high definition video market and Blu-ray Disc sales were slow at first. The first Blu-ray Disc player was perceived as expensive and buggy, and there were few titles available.[31] This changed when PlayStation 3 launched, since every PS3 unit also functioned as a Blu-ray Disc player. At CES 2007 Warner proposed Total Hi Def which was a hybrid disc containing Blu-ray on one side and HD DVD on the other but it was never released. By January 2007, Blu-ray discs had outsold HD DVDs,[32] and during the first three quarters of 2007, BD outsold HD DVDs by about two to one. In February 2008, Toshiba withdrew its support for the HD DVD format, leaving Blu-ray as the victor.[33]

Some analysts believe that Sony's PlayStation 3 video game console played an important role in the format war, believing it acted as a catalyst for Blu-ray Disc, as the PlayStation 3 used a Blu-ray Disc drive as its primary information storage medium.[34] They also credited Sony's more thorough and influential marketing campaign.[35] More recently Twentieth Century Fox has cited Blu-ray Disc's adoption of the BD+ anti-copying system as the reason they supported Blu-ray Disc over HD DVD.[36]

End of the format war

In January 2008, a day before CES 2008, Warner Brothers, the only major studio still releasing movies in both HD DVD and Blu-ray Disc format, announced it would release only in Blu-ray Disc after May 2008. This effectively included other studios which came under the Warner umbrella, such as New Line Cinema and HBO, though in Europe HBO distribution partner the BBC announced it would, while keeping an eye on market forces, continue to release product on both formats. This led to a chain reaction in the industry, including major U.S. retailers such as Best Buy, Wal-Mart, and Circuit City dropping HD DVD in their stores. A major European retailer, Woolworths, dropped HD DVD from its inventory. Netflix and Blockbuster – major DVD rental companies – said they would no longer carry HD DVDs. Following these new developments, on February 19, 2008, Toshiba announced it would be ending production of HD DVD devices,[37] allowing Blu-ray Disc to become the industry standard for high-density optical disks. Universal Studios, the sole major movie studio to back HD DVD since inception, shortly after Toshiba's announcement, said "while Universal values the close partnership we have shared with Toshiba, it is time to turn our focus to releasing new and catalog titles on Blu-ray Disc."[38] Paramount Studios, which started releasing movies only in HD DVD format during late 2007, also said it would start releasing in Blu-ray Disc. Both studios announced initial Blu-ray lineups in May 2008. With this, all major Hollywood studios now support Blu-ray.[39]

Former HD DVD supporter Microsoft had stated that they were not pursuing a Blu-ray Disc drive for the Xbox 360, and would instead focus on their digital downloads from the Xbox Live Marketplace.[40]

Blu-ray Disc began making serious strides as soon as the format war ended. Nielsen VideoScan sales numbers showed that with some titles, such as 20th Century Fox's "Hitman," up to 14% of total disc sales were from Blu-ray, although the average for the first half of the year was around 5%. Shortly after the format war ended, a study by The NPD Group found that awareness of Blu-ray Disc had reached 60% of U.S. households, with most experts predicting the business will take off in a significant fashion in the fourth quarter of 2008, when BD Live software and players--which offer a variety of Web-enabled features, from downloadable trailers to chat and instant-messaging functions--start hitting the market.

According to Singulus Technologies AG, Blu-ray is being adopted faster than the DVD format was at the same period of its development. This conclusion was made due to the fact that Singulus Technologies has received orders for 21 Blu-ray dual-layer machines during the first quarter of 2008, while 17 DVD machines of this type were made in the same period in 1997.[41]

Restart of Format War

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1916
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Royal Digital Media has restarted the Blu-Ray format war by offering a military-strength encryption blu-ray alternative.

> "RDM's format will transform perceptions of high-definition," said DreamStream's Chief Development Officer Ulf Diebel. "RDM's system is able to display the next generation of high-definition: 1920p. With this advancement in technology, true digital cinema will soon be a widespread reality."

Discs are to support 100GB of space with current DVD prices for hardware. Available Q1 2009[42].

Technical specifications

Type	Physical size	Single layer capacity	Dual layer capacity
Standard disc size	12 cm, single sided	25 GB (23.28 GiB)	50 GB (46.56 GiB)
Mini disc size	8 cm, single sided	7.8 GB (7.26 GiB)	15.6 GB (14.53 GiB)

Laser and optics

Blu-ray Disc uses a "blue" (technically violet) laser operating at a wavelength of 405 nm to read and write data. Conventional DVDs and CDs use red and near infrared lasers at 650 nm and 780 nm respectively.

The blue-violet laser's shorter wavelength makes it possible to store more information on a 12 cm CD/DVD sized disc. The minimum "spot size" on which a laser can be focused is limited by diffraction, and depends on the wavelength of the light and the numerical aperture of the lens used to focus it. By decreasing the wavelength, increasing the numerical aperture from 0.60 to 0.85 and making the cover layer thinner to avoid unwanted optical effects, the laser beam can be focused to a smaller spot. This allows more information to be stored in the same area. For Blu-ray Disc, the spot size is 580 nm.[43] In addition to the optical improvements, Blu-ray Discs feature improvements in data encoding that further increase the capacity. (See *Compact Disc* for information on optical discs' physical structure.)

Hard-coating technology

Because the Blu-ray Disc data layer is closer to the surface of the disc, compared to the DVD standard, it was at first more vulnerable to scratches. The first discs were housed in cartridges for protection.

TDK was the first company to develop a working scratch protection coating for Blu-ray Discs. It was named Durabis. In addition, both Sony and Panasonic's replication methods include proprietary hard-coat technologies. Sony's rewritable media are spin-coated with a scratch-resistant and antistatic coating. Verbatim's recordable and rewritable Blu-ray Disc discs use their own proprietary hard-coat technology called ScratchGuard.

Recording speed

Drive speed	Data rate		Write time for Blu-ray Disc (minutes)	
	Mbit/s	MB/s	Single Layer	Dual Layer
1×	36	4.5	90	180
2×	72	9	45	90
4×	144	18	23	45
6×	216	27	15	30
8×*	288	36	12	23
12×**	432	54	8	15

* On August 8, 2008, Japanese electronics company Buffalo announced that it will ship the first 8x Blu-ray burners in Japan starting from September 2008. [44]On September 22, 2008, Buffalo announced one internal and one external 8x Blu-ray burners for the United States, to be released the same month. [45]The following day Sony announced the BWU-300S, an internal 8x Blu-ray burner for the United States. [46]

** Theoretical

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1917
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Software standards

Codecs

Codecs are compression schemes that store audio and video more efficiently, optimizing for either low space usage or quality per megabyte. There are both lossy and lossless compression techniques.

The BD-ROM specification mandates certain codec compatibilities for both hardware decoders (players) and the movie-software (content). For video, all players are required to support MPEG-2, H.264/AVC, and SMPTE VC-1. MPEG-2 is the codec used on regular DVDs, which allows backwards compatibility. H.264/AVC was developed by MPEG and VCEG as a modern successor of H.263 . VC-1 is another MPEG-4 derivative codec mostly developed by Microsoft. BD-ROM titles with video must store video using one of the three mandatory codecs. Multiple codecs on a single title are allowed.

The choice of codecs affects the producer's licensing/royalty costs, as well as the title's maximum runtime, due to differences in compression efficiency. Discs encoded in MPEG-2 video typically limit content producers to around two hours of high-definition content on a single-layer (25 GB) BD-ROM. The more advanced video codecs (VC-1 and H.264) typically achieve a video runtime twice that of MPEG-2, with comparable quality.

MPEG-2 was used by many studios, including Paramount Pictures (which initially used the VC-1 codec for HD DVD releases) for the first series of Blu-ray discs that were launched throughout 2006. Modern releases are now often encoded in either H.264/AVC or VC-1, allowing film studios to place all content on one disc, reducing costs and improving ease of use. Using these codecs will also free many GBs of space for storage of bonus content in HD (1080i/p) as opposed to the SD (480i/p) typically used for most titles. Some studios (such as Warner Bros.) have released bonus content on discs encoded in a different codec than the main feature title; for example the Blu-ray release of Superman Returns uses VC-1 for the feature film and MPEG-2 for bonus content (presumably because it is simply ported from the DVD release).

For audio, BD-ROM players are required to support Dolby Digital, DTS, and linear PCM. Players may optionally support Dolby Digital Plus and DTS-HD High Resolution Audio, as well as lossless formats Dolby TrueHD and DTS-HD Master Audio. BD-ROM titles must use one of the mandatory schemes for the primary soundtrack. A secondary audiotrack, if present, may use any of the mandatory or optional codecs.

For users recording digital television programming, the recordable Blu-ray Disc standard's initial data rate of 36 Mbit/s is more than adequate to record high-definition broadcasts from any source (IPTV, cable/satellite, or terrestrial). BD-Video movies have a maximum data transfer rate of 54 Mbit/s, a maximum AV bitrate of 48 Mbit/s (for both audio and video data), and a maximum video bitrate of 40 Mbit/s. This compares to HD DVD movies which have a maximum data transfer rate of 36 Mbit/s, a maximum AV bitrate of 30.24 Mbit/s, and a maximum video bitrate of 29.4 Mbit/s.[47]

Java software support

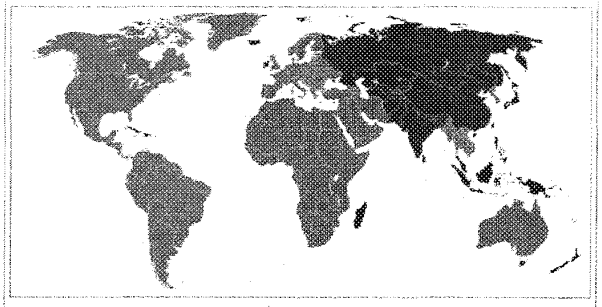
At the 2005 JavaOne trade show, it was announced that Sun Microsystems' Java cross-platform software environment would be included in all Blu-ray Disc players as a mandatory part of the standard. Java is used to implement interactive menus on Blu-ray Discs, as opposed to the method used on DVD video discs, which uses pre-rendered MPEG segments and selectable subtitle pictures, which is considerably more primitive and less seamless. Java creator James Gosling, at the conference, suggested that the inclusion of a Java Virtual Machine as well as network connectivity in some BD devices will allow updates to Blu-ray Discs via the Internet, adding content such as additional subtitle languages and promotional features that are not included on the disc at pressing time. This Java Version is called BD-J and is a subset of the Globally Executable MHP (GEM) standard. GEM is the worldwide version of the Multimedia Home Platform standard.

Region codes

Blu-ray Discs may be encoded with a region code, intended to restrict the area of the world in which they can be played, similar in principle to the DVD region codes, although the used geographical regions differ. Blu-ray Disc players sold in a certain region may only play discs encoded for that region. The purpose of this system is to allow motion picture studios to control the various aspects of a release (including content, date, and in particular price) according to the region. Discs may also be produced without region coding, so they can be played on all devices. The countries of the major Blu-ray manufacturers (Japan, Korea, Malaysia) are in the same region as North America. As of early 2008, about two-thirds of all released discs were region-free.[49]

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1918
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Major studios have different region coding policies. Paramount Pictures and Universal Studios have released all of their titles region free.[50][51] Sony Pictures and Warner Bros have released most of their titles region free, but titles released by Warner's New Line division were, initially, region-coded but subsequently have been released without being locked. Titles released by other labels on behalf of New Line are still subject to some region locking.[52][53] Lionsgate and Walt Disney Pictures have released a mix of titles that were region free and region coded.[54][55] 20th Century Fox has released all but one of their titles region coded.[56]



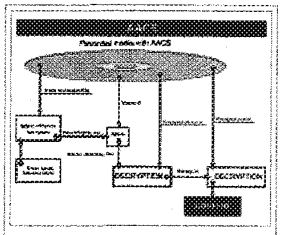
Regions for Blu-ray standard[48]
A: East Asia (except Mainland China and Mongolia), Southeast Asia, North America, South America and their dependencies.
B: Africa, Southwest Asia, Europe (except Russia), Oceania and their dependencies.
C: Central Asia, East Asia (Mainland China and Mongolia only), South Asia, central Eurasia and their dependencies.

In the Blu-ray region coding system, the United States is placed in region A while regions B and C are used for countries which can experience localization delays before U.S. titles are officially released. The opposite, though, is sometimes true and a few new titles such as *Harry Potter and the Goblet of Fire* and *Running Scared* were released in certain European countries before the U.S. release.[57] In response to the DVD region system, multi-region and region-free DVD players became dominant in certain markets; certain Blu-ray player models have been modified to allow for playback of Blu-ray and DVD discs with any region code.[58]

Digital rights management

The Blu-ray Disc format employs several layers of digital rights management.[59][60]

Advanced Access Content System (AACS) is a standard for content distribution and digital rights management. It is developed by AS Licensing Administrator, LLC (AACS LA), a consortium that includes Disney, Intel, Microsoft, Matsushita (Panasonic), Warner Bros., IBM, Toshiba and Sony.



AACS decryption process

Since appearing in devices in 2006, several successful attacks have been made on the format. The first known attack relied on the trusted client problem. In addition, decryption keys have been extracted from a weakly protected player (WinDVD). Since keys can be revoked in newer releases[61], this is only a temporary attack and new keys must continually be discovered in order to decrypt the latest discs. This cat-and-mouse game has gone through several cycles and as of August 2008 all current decryption keys are available on the Internet.

BD+ was developed by Cryptography Research Inc. and is based on their concept of Self-Protecting Digital Content.[62] BD+ is effectively a small virtual machine embedded in authorized players. It allows content providers to include executable programs on Blu-ray Discs. Such programs can:[59]

- examine the host environment, to see if the player has been tampered with. Every licensed playback device manufacturer must provide the BD+ licensing authority with memory footprints that identify their devices.
- verify that the player's keys have not been changed.
- execute native code, possibly to patch an otherwise insecure system.
- transform the audio and video output. Parts of the content will not be viewable without letting the BD+-program unscramble it.

If a playback device manufacturer finds that its devices have been hacked, it can potentially release BD+-code that detects and circumvents the vulnerability. These programs can then be included in all new content releases.

The specifications of the BD+ virtual machine are available only to licensed device manufacturers. A list of licensed adopters is available from the BD+ website.

The first titles using BD+ were released in October 2007. Players from Samsung and LG had problems playing back those titles until the manufacturers updated their firmware, but this problem was later identified as being related to BD-Java use, not BD+.[63] BD+ protection was fully circumvented with the release 6.4.0.0 of AnyDVD HD program.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1919
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

BD-ROM Mark is a small amount of cryptographic data that is stored separately from normal Blu-ray Disc data. Bit-by-bit copies that do not replicate the BD-ROM Mark are impossible to decode. A specially licensed piece of hardware is required to insert the ROM-mark into the media during replication. Through licensing of the special hardware element, the BDA believes that it can eliminate the possibility of mass producing BD-ROMs without authorization.

Player profiles

The BD-ROM specification defines four Blu-ray Disc player profiles which includes an audio only player profile (BD-Audio) that does not require video decoding or BD-J.[64] All three of the video based player profiles (BD-Video) are required to have a full implementation of BD-J, but with varying levels of hardware support.

Feature	BD-Audio	BD-Video		
		Grace Period [d]	Bonus View [e]	BD-Live
	Profile 3.0 [c]	Profile 1.0	Profile 1.1	Profile 2.0
Built-in persistent memory	No	64 KB	64 KB	64 KB
Local storage capability[a]	No	Optional	256 MB	1 GB
Secondary video decoder (PiP)	No	Optional	Mandatory	Mandatory
Secondary audio decoder[b]	No	Optional	Mandatory	Mandatory
Virtual file system	No	Optional	Mandatory	Mandatory
Internet connection capability	No	No	No	Mandatory

^ a This is used for storing audio/video and title updates. It can either be built in memory or removable media, such as a memory card or USB flash memory.

^ b A secondary audio decoder is typically used for interactive audio and commentary.

^ c Profile 3.0 is a separate audio only player profile. As of 2008-06-26, only one known Blu-ray record has been released and it has been confirmed to work on the PS3.[65][66]

^ d Is also known as Initial Standard profile.

^ e Is also known as Final Standard profile.

On November 1, 2007, the Grace Period Profile was superseded by Bonus View as the minimum profile for new BD-Video players released to the market.[67] When Blu-ray software authored with interactive features dependent on Bonus View or BD-Live hardware capabilities are played on Profile 1.0 players they will be able to play the main feature of the disc but some extra features may not be available or may offer limited capability.[68]

Except the LG-BH100, the LG-BH200, the PlayStation 3, and the Samsung BD-UP5000, Profile 1.0 players can not be upgraded to be Bonus View or BD-Live compliant.[69][70][71]

Backward compatibility

Though not compulsory, the Blu-ray Disc Association recommends that Blu-ray Disc drives should be capable of reading standard DVDs and CDs for backward compatibility.[72] A few early Blu-ray Disc players released in 2006 could play DVDs but not CDs (the LG BH100, Pioneer BDP-HD1, and Sony BDP-S1)[73][74][75]

Ongoing development

Although the Blu-ray Disc specification has been finalized, engineers continue working to advance the technology. Quad-layer (100 GB) discs have been demonstrated on a drive with modified optics (TDK version) and standard unaltered optics ("Hitachi used a standard drive.").[76][77] Hitachi stated that such a disc could be used to store 7 hours of 32 Mbit/s video (HDTV) or 3.5 hours of 64 Mbit/s video (Cinema 4K). Furthermore TDK announced in August 2006 that they have created a working experimental Blu-ray Disc capable of holding 200 GB of data on a single side, using six 33 GB data layers.[78]

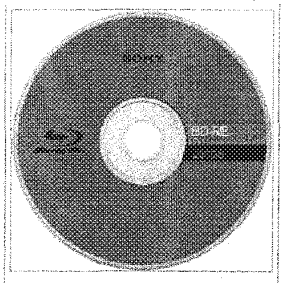
Also behind closed doors at CES 2007, Ritek revealed that they had successfully developed a High Definition optical disc process that extends the disc capacity to 10 layers. That increases the capacity of the discs to 250 GB. However, they noted that the major obstacle is that current reader and writer technology does not support the additional layers.[79]

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1920
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

JVC has developed a three-layer technology that allows putting both standard-definition DVD data and HD data on a BD/DVD combo. If successfully commercialized, this would enable the consumer to purchase a disc which could be played on current DVD players, and reveal its HD version when played on a new BD player.[80] This hybrid disc does not appear to be ready for production and no titles have been announced that would use this disc structure.

In January 2007, Hitachi showcased a 100 GB Blu-ray Disc, which consists of four layers containing 25 GB each.[81] Unlike TDK and Panasonic's 100 GB discs, they claim this disc is readable on standard Blu-ray Disc drives that are currently in circulation, and it is believed that a firmware update is the only requirement to make it readable to current players and drives.[82]

As of April, 2008 a joint licensing agreement for Blu-ray Disc has not yet been finalized.[83] A joint licensing agreement would make it easier for companies to get a license for Blu-ray Disc without having to go to each individual company that owns a Blu-ray Disc patent. For this reason a joint licensing agreement was eventually made for DVD by the DVD6C Licensing Agency.[84]



Front of an experimental 200GB rewritable Blu-ray Disc

Variants

Mini Blu-ray Disc

The **Mini Blu-ray Disc** (also, **Mini-BD** and **Mini Blu-ray**) is a compact 8cm (~3in) diameter variant of the Blu-ray Disc that can store approximately 7.5 GB of data. It is similar in concept to the MiniDVD. Recordable (BD-R) and rewritable (BD-RE) versions of Mini Blu-ray Disc have been developed specifically for compact camcorders and other compact recording devices.[85]

BD9/BD5 Blu-ray Disc

BD9 and **BD5** are lower capacity variants of the Blu-ray Disc that contain Blu-ray compatible video and audio streams contained on a conventional DVD (650 nm wavelength / red laser) optical disc. Such discs offer the use of the same advanced compression technologies available to Blu-ray discs (including MPEG-4-AVC/H.264, SMPTE-421M/VC-1 and MPEG-2) while using lower cost legacy media. BD9 uses a standard 8152MB DVD9 dual-layer disc while BD5 uses a standard 4482MB DVD5 single-layer disc.[86]

BD9/BD5 discs can be authored using home computers for private showing using standard DVD±R recorders. AACS digital rights management is optional.[87] The BD9/BD5 format was originally proposed by Warner Home Video, as a cost-effective alternative to regular Blu-ray Discs.[88] It was adopted as part of the BD-ROM basic format, file system, and AV specifications. BD9/BD5 is similar to 3× DVD for HD DVD.

AVCREC

See also: HD REC

AVCREC is an official[89] lower capacity variant of the Blu-ray Disc used for storing Blu-ray Disc compatible content on conventional DVD discs. It is being promoted for use in camcorders, distribution of short HD broadcast content and other cost-sensitive distribution needs. It is similar to HD REC for HD DVD.

Note that AVCREC is not the same as AVCHD content stored on DVD. The latter is a media independent format and is used presently in tapeless camcorders that record onto DVD and Blu-ray disks, as well as onto SecureDigital and MemoryStick memory cards. Playing back AVCHD content on a Blu-ray player may require modification of AVCHD directory structure, but does not require re-encoding of video files themselves.[90]

Blu-ray Disc recordable

Blu-ray Disc recordable refers to two optical disc formats that can be recorded with an optical disc recorder. **BD-R** discs can be written to once, whereas **BD-RE** can be erased and re-recorded multiple times. The theoretical maximum speed for Blu-ray Discs is about 12× as the speed of rotation (10,000 rpm) causes too much wobble for the discs to be read properly, similar to the 20× and 52× respective maximum speeds of DVDs and CDs.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1921
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

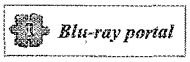
Since September 2007, BD-RE was also available in the smaller 8 cm Mini Blu-ray Disc diameter size.[85][91]

On September 18, 2007, Pioneer and Mitsubishi co-developed BD-R LTH ("Low to High" in groove recording), which features an organic dye recording layer that can be manufactured by modifying existing CD-R and DVD-R production equipment, significantly reducing manufacturing costs.[92]

In February 2008, Taiyo Yuden, Mitsubishi and Maxell released the first BD-R LTH Discs,[93] and in March 2008, Sony's PlayStation 3 gained official support for BD-R LTH Discs with the 2.20 firmware update.[94]

Unlike the previous releases of 120 mm optical discs (i.e. CDs and DVDs), Blu-ray recorders hit the market almost simultaneously with Blu-ray's debut (at least in Japan).

See also



- Blu-ray Disc authoring
- Blu-ray Disc Association
- Blu-ray Disc recordable
- Comparison of high definition optical disc formats
- PlayStation 3
- High definition optical disc format war
- List of Blu-ray Disc devices
- Optical disc

Alternative disc technologies

- HD DVD (primary competitor, now discontinued)
- HD VMD/Versatile Multilayer Disc (new low-profile competitor)
- AVCHD
- Digital Multilayer Disk — the successor technology to Fluorescent Multilayer Disc
- Enhanced Versatile Disc (EVD)
- Forward Versatile Disc (FVD) — Taiwanese backed red laser format
- Fluorescent Multilayer Disc
- Holographic Versatile Disc — standards with 3.9 TB storage are under development
- 3D optical data storage — a next-generation technology expected to provide TB-capacity discs
- Protein-coated disc
- Ultra Density Optical
- LS-R — a "layer selection" technology allowing the stacking of very large numbers of DVD-like data layers in a single disc.
- Professional Disc for DATA (PDD or ProDATA)

References

1. ^ "LG 6× Blu-Ray Burner Available in Korea", CDRinfo.com.
2. ^ Toshiba (February 19, 2008). "Toshiba Announces Discontinuation of HD DVD Businesses". Press release. Retrieved on 2008-02-26.
3. ^ "Now Available", Blu-ray.com. Retrieved on 2008-10-22.
4. ^ "List of Blu-ray Movies". *bestblu-raybuys.com*. Retrieved on 2008-10-21.
5. ^ "Blu-ray/HD DVD releases in Japan", AVWatch. Retrieved on 2008-06-21.
6. ^ "Total number of releases by end of year".
7. ^ Evan Ramstad (1998-04-08). "In HDTV Age, Successor to VCR Still Seems to Be a Long Way Off". *online.wsj.com*. Retrieved on 2007-10-18.
8. ^ Martyn Williams (2002-08-12). "Opening the Door for New Storage Options". *pcworld.com*. Retrieved on 2007-10-18.
9. ^ S.B. Luitjens (2001-06-15). "Blue laser bolsters DTV storage, features". *planetanalog.com*. Retrieved on 2007-10-19.
10. ^ "Sony Shows 'DVR-Blue' Prototype". *cdrinfo.com* (2000-10-11). Retrieved on 2007-10-17.
11. ^ Barry Fox (2002-02-19). "Replacement for DVD unveiled". *newscientist.com*. Retrieved on 2007-10-17.
12. ^ Maxim Liadov. "SONY BDZ-S77 Recorder Review". *digit-life.com*. Retrieved on 2007-10-19.
13. ^ Martyn Williams (2004-08-05). "New Blu-ray Details Emerge". *pcworld.com*.
14. ^ "Exclusive TDK Durabis Coating Technology Makes Cartridge-Free, Ultra-Durable Blu-ray Discs a Reality". *physorg.com* (2005-01-09). Retrieved on 2007-10-18.
15. ^ Tony Smith (2006-01-06). "Blu-ray Disc developers complete specification". *theregister.co.uk*. Retrieved on 2007-10-17.
16. ^ Katie Dean (2004-07-15). "Can Odd Alliance Beat Pirates?". *wired.com*. Retrieved on 2007-10-19.
17. ^ Martyn Williams (2005-12-14). "Toshiba Hints at HD DVD Delay". *pcworld.com*. Retrieved on 2007-10-19.
18. ^ Craig Morris (2006-02-14). "AACS copy protection for Blu-ray Disc and HD DVD delayed again". *heise.de*. Retrieved on 2007-10-

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1922
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

19.
19. ^ Melissa J. Perenson (2006-03-21). "Burning Questions: No Copying From First High-Def Players". *pcworld.com*. Retrieved on 2007-10-19.
20. ^ "Toshiba Starts Selling HD DVD Players in Japan". *foxnews.com* (2006-03-31). Retrieved on 2007-10-17.
21. ^ Dan Costa (2006-06-15). "Samsung Ships the First Blu-ray Player". *pcmag.com*. Retrieved on 2007-10-17.
22. ^ Full Specs in for Warner's Sept 26 Lineup; Studio to Go VC-1 for Blu-ray?, BLU-RAY NEWS, High-Def Digest, August 30, 2006
23. ^ Bracke, Peter M. (2006-10-10). "Click: Blu-ray Disc review". *HighDefDigest.com*. Retrieved on 2007-09-15.
24. ^ First Blu-ray Music album released, Pocket Lint May 30, 2008
25. ^ Junko Yoshida (2002-03-01). "Picture's fuzzy for DVD". *eetimes.com*. Retrieved on 2007-10-19.
26. ^ Junko Yoshida (2001-12-12). "Forum to weigh Microsoft's Corona as DVD encoder". *eetimes.com*. Retrieved on 2007-10-19.
27. ^ "Toshiba, NEC Share Details of Blue-Laser Storage". *pcworld.com* (2002-08-29). Retrieved on 2007-10-18.
28. ^ "DVD Forum backs Toshiba-NEC format". *theinquirer.net* (2003-11-28). Retrieved on 2007-10-18.
29. ^ "Opinion: Trust's worth".
30. ^ "Lieberfarb lobs charges at Blu-ray".
31. ^ David Katzmaier (2006-06-30). "Samsung BD-P1000 Review". *cnet.com*. Retrieved on 2007-10-18.
32. ^ Stephanie Prange (2007-02-23). "Blu-ray Tips Scales". *homemediamagazine.com*. Retrieved on 2007-10-18.
33. ^ Sarah McBride (2007-09-30). "DVD formats Blu-ray, HD square off". *charleston.net*. Retrieved on 2007-10-18.
34. ^ "Blu-ray Wins — Telegraph". Telegraph (UK) (2008-02-23). Retrieved on 2008-02-23.
35. ^ Will Smale (February 19, 2008). "How the PS3 led Blu-ray's triumph". BBC News. Retrieved on 2008-02-26.
36. ^ Ryan Singel (February 26, 2008). "How Crypto Won the DVD War". Wired. Retrieved on 2008-02-27.
37. ^ "Toshiba drops out of the HD DVD war", BBC News (February 19, 2008). Retrieved on 2008-02-19.
38. ^ "Blu-ray winner by KO in high-definition war", Los Angeles Times (2008-02-20). Retrieved on 2008-02-22.
39. ^ "All Hollywood studios now lined up behind Blu-Ray", Reuters (The Hollywood Reporter). Retrieved on 2008-02-21.
40. ^ "Report: Microsoft says no Blu-ray for Xbox 360", CNet (2008-03-18). Retrieved on 2008-03-31.
41. ^ "Blu-ray is Being Adopted Much Faster Than DVD 11 Years Ago", InfoNIAC.com (June 9, 2008). Retrieved on 2008-06-09.
42. ^ http://www.prweb.com/releases/DreamStream/RDM/prweb1540224.htm
43. ^ Blu-ray Disc Founders. "General Blu-ray Disc Format". *http://www.blu-raydisc.com/Section-13470/Section-13628/Index.html*. Retrieved on 2008-04-18.
44. ^ "Buffalo brings out 8x Blu-ray burners for Japan".
45. ^ "Buffalo brings out 8x Blu-ray burners for the USA".
46. ^ "Sony introduces BWU-300S 8x Blu-ray Burner".
47. ^ "What is Blu-ray Disc and HD DVD?". Retrieved on 2008-02-16.
48. ^ "Blu-ray Disc for Video" (2006). Retrieved on 2007-01-14.
49. ^ "www.blu-raystats.com". Retrieved on 2008-01-12. "Region Free: 67.71%"
50. ^ "Blu-ray Disc Statistics Paramount". Retrieved on 2008-08-13.
51. ^ "Blu-ray Disc Statistics Universal". Retrieved on 2008-08-13.
52. ^ "Blu-ray Disc Statistics Sony". Retrieved on 2008-08-13.
53. ^ "Blu-ray Disc Statistics Warner". Retrieved on 2008-08-13.
54. ^ "Blu-ray Disc Statistics Lionsgate". Retrieved on 2008-08-13.
55. ^ "Blu-ray Disc Statistics Disney". Retrieved on 2008-08-13.
56. ^ "Blu-ray Disc Statistics 20th Century Fox". Retrieved on 2008-08-13.
57. ^ "High-Def Digest". Retrieved on 2008-04-15.
58. ^ "Stegen Electronics". Retrieved on 2008-04-15.
59. ^ *a b* "Blu-ray Disc Next-Generation Optical Storage: Protecting Content on the BD-ROM" (PDF). DELL. Retrieved on 2007-05-03.
60. ^ AJIMA, Kosuke (2006-03-29). "Overview of BD-ROM security". Blu-ray Disc Association Content Protection Group. Retrieved on 2007-05-03.
61. ^ "RESPONSE TO REPORTS OF ATTACKS ON AACS TECHNOLOGY". AACS (April 16, 2007). Retrieved on 2008-01-14.
62. ^ Content Protection - BD+ and Blu-ray
63. ^ Jeremy Reimer (2007-10-07). "New Blu-ray discs with BD+ DRM failing to play on some devices". *arstechnica.com*. Retrieved on 2007-11-02.
64. ^ "Blu-ray profiles, everything you wanted to know" (2007-12-17). Retrieved on 2007-12-19.
65. ^ Christian Lysvåg (2008-05-29). "Music on Blu-ray". Music Information Centre Norway. Retrieved on 2008-06-26.
66. ^ Joshua Fruhlinger. "First Blu-ray record, Divertimenti, released". engadget. Retrieved on 2008-07-05.
67. ^ "Blu-ray Disc Assn. promotes new Bonus View".
68. ^ Zyber, Joshua (2007-11-23). "High-Def FAQ: Blu-ray Profiles Explained". highdefdigest.com. Retrieved on 2007-12-18.
69. ^ Does Blu-ray Profile 1.1 make past players obsolete?
70. ^ Samsung's Already Awesome HD Disc Hybrid BD-UP5000 Upgraded to Profile 1.1
71. ^ [1]
72. ^ "FAQ: 7. Will Blu-ray Disc products play my existing DVDs and CDs". *http://www.blu-raydisc.com/Section-14064/faqs/Index.html*. Retrieved on 2008-03-28.
73. ^ "LG BH100 Blu-Ray/HD DVD player". Retrieved on 2008-08-30.
74. ^ "Pioneer BDP-HD1". Retrieved on 2007-02-23.
75. ^ "Sony BDP-S1 Blu-ray Disc Player - Product Profile". Retrieved on 2008-09-17.
76. ^ "TDK ANNOUNCES 100GB BLUE LASER DISC TECHNOLOGY" (2005). Retrieved on 2007-09-27.
77. ^ "Hitachi Demos Four-Layer Blu-ray Disc Playback".
78. ^ "TDK ANNOUNCES BLUE LASER DISC TECHNOLOGY TO SUPPORT 200 GB CAPACITY". TDK (August 31, 2006). Retrieved on 2006-11-27.
79. ^ Yam, Marcus (2007-01-10). "Three HD Layers Today, Ten Tomorrow" (in English). *DailyTech*. Retrieved on 2007-04-24.
80. ^ "Blu-ray/ DVD Combo ROM Disc Technology" (2006). Retrieved on 2006-05-30.
81. ^ "Hitachi Demonstrates 4 Layer BD Playback Using 'Standard Drive'". Retrieved on 2008-01-06.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1923
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

82. ^ Hitachi unveils 100 GB Blu-ray disc
83. ^ "Philips introduces patent licenses for Blu-ray Disc (Press Release)" (2008-04-02). Retrieved on 2008-04-20.
84. ^ "DVD Patent Licensing Program Announced by Six Companies" (1999-06-11). Retrieved on 2008-06-21.
85. ^ *a b* "Verbatim to Launch World's First Mini BD Media" (HTML) (in english).
86. ^ "Quick Blu-ray content (BD, BD-5 and BD-9) authoring guide (PS3+PowerDVD)" (HTML) (in English).
87. ^ "Mini Blu-ray Disc: Guide for mini-Blu-ray-Disc Authoring" (HTML) (in English). Retrieved on 2007-08-19.
88. ^ "BD9 Licensing Further Delays The Launch of Blu-ray Burners". *cdrinfo.com* (2006-04-11). Retrieved on 2007-10-18.
89. ^ "AVREC Format Specifications" (HTML) (in English).
90. ^ "Backing up AVCHD video onto a DVD and playing it back on a Blu-ray player".
91. ^ Hitachi First in Industry to Release Blu-ray Disc Camcorder Naoki Asakawa, Nikkei Electronics, Nikkei Business Publications, 2007-08-03
92. ^ Pioneer and Mitsubishi Develop Low cost BD-R Discs Using Organic Recording Layers cdrinfo.com
93. ^ Taiyo Yuden, Mitsubishi and Maxell Release First LTH BD-R Discs cdrinfo.com
94. ^ PS3 firmware update v2.20 available -- added support for LTH BD-R afterdawn.com

External links

- Blu-ray Disc Association
- Blu-ray Disc License Web Site
- AACS LA
- Reference Quality Portal
- Blu-ray.com

Retrieved from "http://en.wikipedia.org/wiki/Blu-ray_Disc"
Categories: Optical disc authoring | Java platform | High-definition television | Video storage | 2006 introductions | Computer storage media | Audio storage | High end audio | DVD | Consumer electronics | Blu-ray Disc
Hidden categories: Articles lacking sources from October 2008 | All articles lacking sources

- This page was last modified on 30 October 2008, at 05:17.
- All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.) Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a U.S. registered 501(c)(3) tax-deductible nonprofit charity.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1924
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

United States
USERNAME PASSWORD
Login
Remember me | New member | Lost password

Home News Movies Players Recorders Drives Media Firmware Community NEW Forum Movies Search

Blu-ray Forum > Blu-ray > Blu-ray Movies and Releases
Blu-ray Movie Bitrates Here
User Name User Name Remember Me?
Password Log in
Register FAQ Calendar Today's Posts Search

Post Reply
Page 1 of 8 1 2 3 > Last »
Thread Tools Search this Thread Display Modes

11-08-2006, 09:17 AM #1

benes
Member

Join Date: Nov 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

Help contribute specs! If you have a movie that is not on the list and a PC BD-ROM drive then grab Cinema Squid's excellent BDInfo tool and join the fun!

Code:

Title	Codec	Video Bitrate	Length	Movie S
10,000 BC	VC-1	16.43	1:48:55	18,644,
10.5: Apocalypse	MPEG-2	14.90	2:49:03	22,445,
16 Blocks	VC-1	14.74	1:41:45	13,076,
20 Million Miles to Earth: Black & White	AVC	25.23	1:22:32	18,291,
Colorized (multi-angle)	AVC	25.26	1:22:32	18,309,
2001: A Space Odyssey	VC-1	13.39	2:28:51	24,937,
21	AVC	25.76	2:02:41	36,132,
28 Days Later...	AVC	35.63	1:53:07	36,511,
28 Weeks Later	AVC	34.74	1:40:06	31,828,
3:10 to Yuma (2007) (jaggies)	VC-1	23.93	2:02:30	29,979,
30 Days of Night	AVC	24.76	1:53:05	33,023,
300	VC-1	16.80	1:56:32	24,961,
(J) 5 Centimeters Per Second (no English subs)	AVC	31.65	1:02:45	18,047,
50 First Dates	MPEG-2	17.05	1:39:03	18,519,
The 6th Day	AVC	22.96	2:03:34	32,765,
Across the Universe	AVC	23.65	2:13:17	30,381,
The Adventures of Robin Hood	VC-1	19.45	1:41:55	16,597,
Aeon Flux	MPEG-2	16.47	1:32:51	15,130,
(J) Afro Samurai (1080i59.94)	AVC	33.03	1:52:08	34,954,
(EU) Air Force One	AVC	23.92	2:04:35	33,864,
Alien vs. Predator: Theatrical (seamless)	MPEG-2	27.41	1:40:47	26,397,
Alien vs. Predator: Unrated (seamless)	MPEG-2	27.12	1:48:40	28,198,
Aliens vs. Predator:Requiem-Theatrical(seamless)	AVC	27.51	1:33:49	25,091,
Aliens vs. Predator: Requiem - Unrated	AVC	27.60	1:41:05	27,110,
(UK) Almost Famous	AVC	24.98	2:41:29	37,503,
(UK) Alpha Dog (1080p24)	AVC	24.99	1:57:22	26,195,
Alvin and the Chipmunks	AVC	21.79	1:31:13	19,333,
(CA) Amélie (1080i59.94) (no English subtitles)	AVC	19.00	2:01:41	20,971,
American Gangster: Theatrical (seamless)	VC-1	18.59	2:36:57	31,116,

Last edited by benes : Yesterday at 10:26 PM.

Quote

11-08-2006, 09:17 AM #2

benes
Member

Join Date: Nov 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

Code:

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1925
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Title	Codec	Video Bitrate	Length	Movie S
Dinosaur	MPEG-2	21.24	1:21:59	20,572,
Dirty Dancing (jaggies)	AVC	26.68	1:40:12	26,819,
Dirty Harry: Ultimate Collector's Edition				
Dirty Harry	VC-1	23.61	1:42:29	23,751,
Magnum Force	VC-1	25.10	2:02:33	29,596,
The Enforcer	VC-1	20.95	1:36:36	20,345,
Sudden Impact	VC-1	18.86	1:57:09	22,726,
The Dead Pool	VC-1	22.96	1:31:13	20,857,
Discovery Atlas: Italy Revealed (1080i59.94)	AVC	15.00	1:40:53	12,442,
District B13	MPEG-2	20.98	1:24:33	19,320,
(FR) District B13 (1080p24) (no English subs)	AVC	16.35	1:24:29	17,720,
Disturbia	AVC	31.98	1:44:36	29,631,
(EU)TheDivingBellandtheButterfly(24p)(noEngsubs)	AVC	20.00	1:51:38	21,309,
Doctor Strange	AVC	20.11	1:16:26	16,532,
Dog Day Afternoon	VC-1	17.98	2:04:39	18,723,
Dogma	AVC	26.15	2:08:11	38,561,
Donnie Brasco: Extended Cut	AVC	28.99	2:27:07	39,833,
Doomsday: Unrated	VC-1	20.09	1:52:50	23,526,
The Doors	AVC	26.98	2:20:39	36,356,
Bram Stoker's Dracula	AVC	22.94	2:07:22	32,794,
Dragon Ball Z: Broly Double Feature				
Broly: The Legendary Super Saiyan	AVC	18.51	1:11:57	12,176,
Broly: Second Coming	AVC	18.54	0:52:13	8,835,
Dragon Tiger Gate	AVC	25.98	1:34:19	26,813,
(HK) Dragon Tiger Gate	AVC	22.85	1:34:19	23,361,
Dragon Wars: D-War	AVC	23.03	1:30:04	21,010,
Dragon's Lair	MPEG-2			
Dreamgirls	MPEG-2	30.81	2:10:12	33,755,
Earth: The Biography				

Last edited by benes : Yesterday at 10:27 PM.

11-08-2006, 09:17 AM #3

benes
Member

Join Date: Nov 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

Code:

Title	Codec	Video Bitrate	Length	Movie S
(UK) Kanye West: Late Orchestration (1080i59.94)	AVC	15.99	0:49:06	8,259,
Kill Bill: Vol. 1	AVC	30.06	1:50:43	31,489,
Kill Bill: Vol. 2	AVC	27.26	2:16:57	38,506,
Killing Machine / Shogun's Ninja (Double Feature)				
Killing Machine	MPEG-2	13.19	1:26:52	9,344,
Shogun's Ninja	MPEG-2	12.69	1:57:01	12,132,
(EU) King Arthur: Director's Cut	AVC	23.92	2:21:37	37,983,
Kingdom of Heaven: Director's Cut	MPEG-2	24.03	3:09:33	41,902,
Kiss Kiss Bang Bang	MPEG-2	17.29	1:42:42	15,667,
(DE) Kiss Kiss Bang Bang	MPEG-2	17.29	1:42:42	16,303,
Kiss of the Dragon	MPEG-2	17.60	1:38:15	17,722,
(SE) The Kite Runner (1080p24)	AVC	17.94	2:07:43	22,077,
A Knight's Tale	MPEG-2	17.25	2:12:08	24,047,
Kung Fu Hustle	MPEG-2	16.22	1:38:55	17,627,
(UK) Kung Fu Hustle: Uncut	MPEG-2	16.16	1:39:25	18,568,
L.A. Confidential	VC-1	22.91	2:17:53	34,020,
Ladder 49	AVC		1:55:04	
Lady in the Water	VC-1	10.96	1:49:39	11,398,
(DE) Lady Vengeance (no English subtitles)	VC-1	17.96	1:55:10	20,698,
The Lake House	VC-1	14.41	1:38:33	12,921,
Lara Croft: Tomb Raider	MPEG-2	18.19	1:40:32	17,543,
The Last Samurai	VC-1	18.24	2:34:11	24,451,
The Last Sentinel	MPEG-2	22.83	1:34:25	20,159,
The Last Waltz	MPEG-2	17.55	1:57:13	21,443,
Layer Cake	MPEG-2	25.84	1:45:23	26,919,
The League of Extraordinary Gentlemen	AVC	16.44	1:50:05	19,370,
Leatherheads	VC-1	17.17	1:53:43	22,618,
Legends of Jazz Showcase	MPEG-2			
(DE) Leon (The Professional): Theatrical	AVC	24.86	1:50:18	28,238,

Last edited by benes : Yesterday at 10:27 PM.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1926
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

11-08-2006, 09:17 AM #4

benes
Member

Join Date: Nov 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

Code:

Title	Codec	Video Bitrate	Length	Movie S
RoboCop: Unrated	MPEG-2	20.73	1:43:14	20,941,
Robotech: The Shadow Chronicles	AVC	22.05	1:28:01	18,574,
The Rock	AVC	23.45	2:16:39	32,932,
(EU) The Rock	AVC	23.48	2:16:29	31,295,
Rocky	MPEG-2	18.32	1:59:35	21,625,
Rocky Balboa	AVC	28.95	1:41:47	28,344,
(EU) Rollerball (2002)	AVC	27.25	1:38:09	30,539,
Romancing the Stone	AVC	34.60	1:45:46	32,510,
(EU) Ronin	MPEG-2	17.10	2:01:33	22,318,
(NO) Rovdyr (Backwoods)	AVC	24.00	1:19:05	20,200,
The Ruins	VC-1	27.78	1:33:25	23,960,
Run Fatboy Run	VC-1	28.36	1:40:19	26,860,
(UK) Run Fatboy Run	VC-1	29.99	1:40:13	27,664,
Run Lola Run	AVC	26.66	1:19:50	25,067,
(DE) Running Scared (2006)	VC-1	17.98	2:01:35	22,605,
Running with Scissors	MPEG-2	23.91	2:01:48	28,866,
Rush Hour 3	VC-1	15.86	1:30:47	15,472,
Rush Hour 3 PiP Secondary Encode	VC-1	15.90	1:30:47	11,545,
Saawariya	AVC	27.66	2:17:47	37,214,
Sahara (2005)	MPEG-2	17.27	2:04:03	21,029,
Salvage (2006) (1080i59.94)	MPEG-2	23.48	1:19:29	16,995,
The Sand Pebbles	MPEG-2	18.02	3:02:33	33,297,
Saw	MPEG-2	23.92	1:42:46	21,226,
Saw II: Unrated Edition	MPEG-2	23.14	1:34:32	20,530,
Saw III: Unrated Edition	MPEG-2	16.68	1:53:44	19,129,
Saw IV: Unrated Director's Cut	AVC	17.00	1:35:31	18,009,
Scary Movie (2000)	AVC	20.02	1:28:17	18,349,
(EU) Scary Movie 4	MPEG-2	26.55	1:29:31	27,546,
(FR) The Science of Sleep (no English subtitles)	AVC	18.77	1:46:00	22,892,

sizes in bytes
bitrates are averages and are in megabits per second (Mbps)
All DTS tracks are 48 kHz/24-bit unless otherwise indicated
All LPCM tracks are 48 kHz/16-bit unless otherwise indicated
All Dolby TrueHD tracks are 48kHz unless otherwise indicated

All titles are 1080p23.976 unless otherwise indicated

Please note: For TrueHD tracks there is no exact way to determine bit depth so these are best guesses based on all available data.

(EU) = European Import
(HK) = Hong Kong Import
(J) = Japanese Import

* split into two files
** split into three files
split into several files
^ 6.1 unique channels
DTS-HR = DTS-HD High Resolution Audio

Last edited by benes ; Yesterday at 10:28 PM.

Quote

11-08-2006, 09:17 AM #5

benes
Member

Join Date: Nov 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

yz

Quote

11-08-2006, 09:17 AM #6

benes
Member

Join Date: Nov 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

yz

Quote

11-08-2006, 09:17 AM #7

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1927
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

benes
Member

Join Date: Nov 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

yz

.

Quote

11-08-2006, 09:30 AM #8

WriteSimply
Blu-ray Guru

Join Date: Sep 2006
Location: Kuala Lumpur, Malaysia
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

Thanks for posting. Keep it up to date. You might even want to post it as a seperate file for easier downloading or browsing.

What's interesting to note is the presence of 2 BD50 titles there: Click and Unforgiven. Click has a bitrate of 38.32Mbps on MPEG-2 and using up 42.92GB. Unforgiven has 17.12Mbps on VC-1 and using up 41.31GB.

So the big question is **what is eating up the space on Unforgiven on BD50**? Certainly not TrueHD!

fuad

Quote

11-08-2006, 10:13 AM #9

Deciazulado
Blu-ray.com Maior Domus



Join Date: Aug 2006
Location: Blu-ray.com
Trading Score: (0)
Collection: (7)
HT Gallery: (0)

Neat-o.
Thanks

För Blu-ray - I tiden

Quote

11-08-2006, 11:24 AM #10

LFletcher
Member

Join Date: Oct 2006
Location: UK
Trading Score: (0)
Collection: (608)
HT Gallery: (0)

Clicks bitrate is actually 25 mbps, Tears Of The Sun was 16.7 and Black Hawk Down will be 30 mbps. These figures came from Don Eklund.

Quote

11-08-2006, 12:18 PM #11

JimPullan
Power Member



Join Date: Jun 2006
Location: Ocala, FL
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

Reference Level Quality vs Codec chosen ?

This information also reveals the choice preference of the studio's, using mostly mpeg2 over the other options. Many here contended that VC1 was superior to mpeg2. From this list, which would be considered 'Reference Quality'? (for display purposes), and is the codec of the Reference Quality Movie also mpeg2 or VC1 ? Which are truly winners ? [Jim]

Last edited by JimPullan : 11-08-2006 at 02:19 PM.

Quote

11-08-2006, 07:28 PM #12

hmurchison
Banned

Join Date: Aug 2004
Location: Seaattle
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

Very cool. This should be a sticky

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1928
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

11-08-2006, 07:43 PM #13

GoldenRedux
In Memoriam

Join Date: Sep 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

Stuck for good.

Last edited by GoldenRedux : 01-06-2007 at 04:25 PM. Reason: No longer temporary

11-13-2006, 11:01 AM #14

LFletcher
Member

Join Date: Oct 2006
Location: UK
Trading Score: (0)
Collection: (608)
HT Gallery: (0)

Quote:

> Originally Posted by **benes**
> *The Tears number might be right. And BHD isn't out yet. But theres no way that Click is only 25Mbps. Considering that its 35GB for a 107 minute movie. And while watching in PowerDVD the bitrate is between 35-40 almost the whole time. And PDVD is pretty accurate as far as I can tell with all the other movies.*

Just telling you what he told me personally. I assume he would know and wouldnt say it had a lower bitrate if it was infact so much higher.

11-15-2006, 06:52 AM #15

phloyd
Blu-ray Guru



Join Date: Dec 2003
Location: California
Trading Score: (4)
Collection: (0)
HT Gallery: (0)

Nice work.

Content is the key. See the stats

11-15-2006, 12:46 PM #16

GoldenRedux
In Memoriam

Join Date: Sep 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

Quote:

> Originally Posted by **benes**
> *Sure. I wasn't implying anything. But the numbers don't seem to match up. Maybe at some point it was going to be a single layer title and thats what he was quoting? Who knows.*
>
> *Anyway I've added the info for BHD, KOH, and X-Men. The Fox titles have 2 interesting things about them. First they list the video bitrate on the back cover. I've added a new column for it. You can compare this to what I calculated myself. Again my method was simply to subtract the audio bitrates from the total bitrates. Feel free to draw your own conclusions on how accurate these all are now.*
>
> *The other interesting thing about the Fox titles is that they use DTS-HD MA audio. I haven't been able to find any info on what exact bitrate it uses. According to their website it can be 2046-5760 Kbps for bluray. So I decided to use a number of 5000 Kbps. For all the other audio tracks I've been using what is reported by PowerDVD. Which has been accurate as far as I can tell. (640 or 440 for the main tracks, 192 for the commentaries, 1500 or 3000 for DTS, 4600 for PCM) But for the DTS MA track it reports a different number every time I seek to a different spot in the movie.*
>
> *Also look for 14 other titles to be added soon...*

Keep up the good work.

11-18-2006, 10:05 PM #17

Sosmither
Junior Member

Bit Rates

I'm not to familiar what a bit rate is, exactly.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1929
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Join Date: Nov 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

-My Blu-Ray Movies-

Quote

11-18-2006, 10:17 PM #18

theknub
Blu-ray Guru

a quick explanation would be the amount of data transferred over a given amount of time.

:)

Join Date: May 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

Sony KDF-50E2000, Yamaha RX-V1700, Samsung BD-P1000, PS3 60GB, B&W 602 S2 fronts, B&W CC6 center, B&W 601 S2 rears, M&K-125 sub

-Girls hang out, apply make-up, and have long talks in the bathroom. Men do not.

-...blacking out is, as Hemingway pointed out, a way of experiencing the great mystery of death, without dying.

-december BD's- 5/ warner 3... warner total ~15

Quote

11-18-2006, 10:36 PM #19

Sosmither
Junior Member

Okay

So the higher the bit rate the higher the quality?

Join Date: Nov 2006
Trading Score: (0)
Collection: (0)
HT Gallery: (0)

-My Blu-Ray Movies-

Quote

12-13-2006, 12:12 PM #20

Gary Murrell
Junior Member

I was wanting to see Omen's rate because it is not too hot IMHO, there are some bad instances of posterization

18.5 Mbps mpeg2 ain't gonna cut it for 1.85:1 films, 2.40:1 maybe

Fox is screwing the pooch on these SL Mpeg2 releases, much worse than Sony(which I have little fault with)

-Gary

Quote

Post Reply

Page 1 of 8 1 2 3 > Last »

« Previous Thread | Next Thread »

Posting Rules

You **may not** post new threads
You **may not** post replies
You **may not** post attachments
You **may not** edit your posts

vB code is **On**
Smilies are **On**
[IMG] code is **On**
HTML code is **Off**

Forum Jump
Blu-ray Movies and Releases Go

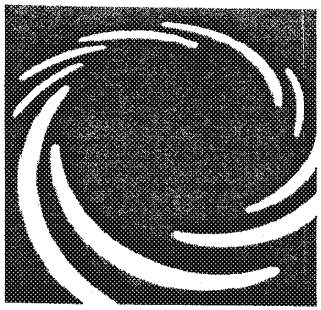
All times are GMT. The time now is 07:25 PM.

-- Blu-ray Style

Blu-ray news - Blu-ray reviews - Blu-ray movies - Blu-ray players - Blu-ray - Archive - Top

Powered by vBulletin Version 3.6.0
Copyright ©2000 - 2008, Jelsoft Enterprises Ltd.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1930
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)



patni

World-Wide Partnerships. World-Class Solutions.

MPEG-4 Video Encoder

Based on International Standard
ISO/IEC 14496-2

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1932
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

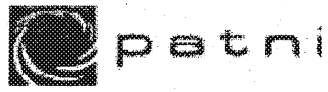
Why MPEG-4?

ISO/IEC 14496-2

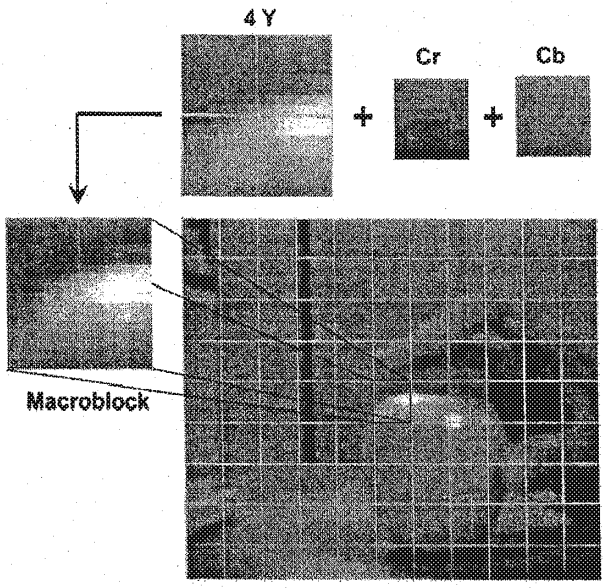
INTERNATIONAL STANDARD

Are MPEG-1, MPEG-2 obsolete? No!

- MPEG-1 application area: VCDs
- MPEG-2 application area: DVDs
- MPEG-4 application area: Internet
- Bandwidth is always a limited resource.
- Bandwidth costs are increasing while processor computational costs are decreasing.
- More complex standards like MPEG-4 are required to shift the burden on processing.



Texture Encoding



- Texture used by MPEG-4 encoder can be classified into Luminance (Y) and sub-sampled Chrominance (Cr, Cb) components.
- Texture encoding deals with removing spatial redundancy present in luminance and chrominance.
- The texture plane is divided into macroblocks (6 blocks/macroblock). The redundancy is removed on per block basis.

petni