ISMA
Internet Streaming Media Alliance

Presidio of San Francisco
P.O. Box 29920
572 B Ruger Street
San Francisco, CA 94129-0920

T 415.561.6276
F 415.561.6120

www.isma.tv

ISMA
INTERNET STREAMING MEDIA ALLIANCE

ISMA Advertising Insertion
Version 1.0
Technical Working Paper

December 2007

ISMA SPECIFICATION LIMITATIONS AND CONDITIONS OF USE

LEGAL LIMITATIONS AND CONDITIONS OF USE

USERS OF THE ISMA SPECIFICATION ARE NOT PERMITTED OR AUTHORIZED TO STATE OR CLAIM THAT THEIR PRODUCTDS OR APPLICATIONS COMPLY WITH THE SPECIFICATION, PENDING ISMA'S DEVELOPMENT AND IMPLEMENTATION OF A COMPLIANCE OR CERTIFICATION PROGRAM AND USER'S EXPRESS AGREEMENT WITH THE TERMS AND CONDITIONS THEREOF. BY REQUESTING OR USING THE SPECIFICATION, USER AGREES TO THIS LIMITATION AND CONDITION.

Table of Contents

1 Document Status

This document is a technical working paper from the ISMA. It provides a discussion of many of the issues and the possible solutions, in inserting advertisements (and generally stream switching between) RTSP/RTP streams. A specification can be built from this, taking into account the needs of a particular environment or deployment. Those developing such specifications are encouraged to refer to this white paper and liaise their work back to ISMA, in the interests of industry convergence.

2 Introduction

Advertisement insertion is when there is broadcast streamed (video) program content with interspersed advertisements. Though the program is logically broadcast (there are many clients displaying the program simultaneously, as it is transmitted), when an advertisement break occurs, the terminals may display different advertisements. At the end of the advertisement break, all terminals return to displaying the broadcast program.

This permits the advertisements to be customized: by geographic location, time of day, to the client using client-supplied information (such as cookies), and so on.

The advertisement is spliced into the stream at a splice point. The splice point may be in one of two places in this specification: in-network, typically at the edge-server; or in the final terminal.

Advertisements typically come in standard 'lengths' (duration, such as 30 seconds).

This document specifies the simple model: program stream – advertisement – back to the program stream.

The design allows the broadcast stream to identify what kind of advertisement should be inserted at each interval, using a URL. These URLs are typically (but not required to be) parameterized CGI URLs referring to a selection function on a server. For example, an agency may have sold the slots differently, depending on when they occur in the program, and it needs to tell the advertisement server (via the splice-point) "provide an advertisement sold for the first break in an international soccer game", "provide an advertisement for the last break in an early-evening children's program", and so on. Which advertisement each client then sees is further refined by the client-specific-advertisement logic (how many times they have seen what, what their previous reaction to advertisements has been, and so on).

3 References

3.1 Normative

[RFC2119] IETF RFC 2119 (March 1997): "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner

[RFC2326] IETF RFC 2326 (April 1998): "Real Time Streaming Protocol (RTSP)", H. Schulzrinne, A. Rao, R. Lanphier.

[RFC2616] IETF RFC 2616 (June 1999): "Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee.

[RFC2965] IETF RFC 2965 (October 2000): "HTTP State Management Mechanism", D. Kristol, L. Montulli.

[RFC3550] IETF STD 0064/RFC 3550 (July 2003) "RTP: A Transport Protocol for Real-Time Applications", H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson.

[SCTE35] ANSI/SCTE 35 2004: " Digital Program Insertion Cueing Message for Cable "

[HDREXT] IETF ID draft-ietf-avt-rtp-hdrext (August 2007): "A general mechanism for RTP Header Extensions", D. Singer, H. Desineni

[ISMAFRTP] Fast Channel Changing in RTP, Version 1.0, ISMA Technical White Paper, December 2007, http://www.isma.tv/

4 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Introduction”, are normative, unless they are explicitly indicated to be informative.

5 Goals and Requirements

The design must be resilient: if the splice operation fails for any reason, or the splice-point becomes overloaded or deems for some reason that per-client advertisements are not appropriate (e.g. an advertisement server failure, bandwidth limits, etc.), or the user tunes-in in the middle of an advertisement-break, something suitable must be displayed (for example, a broadcast advertisement).

The design must support splicing in-network (‘edge server’) or at the terminal (‘client’).

The bandwidth overhead of the signaling etc. must be low.

In the case of edge-server splicing, it should be possible to use an advertisement-unaware terminal. However, it is not a requirement that the resulting stream obey a tight buffer model either at the RTP or the media (audio or video) decoder.

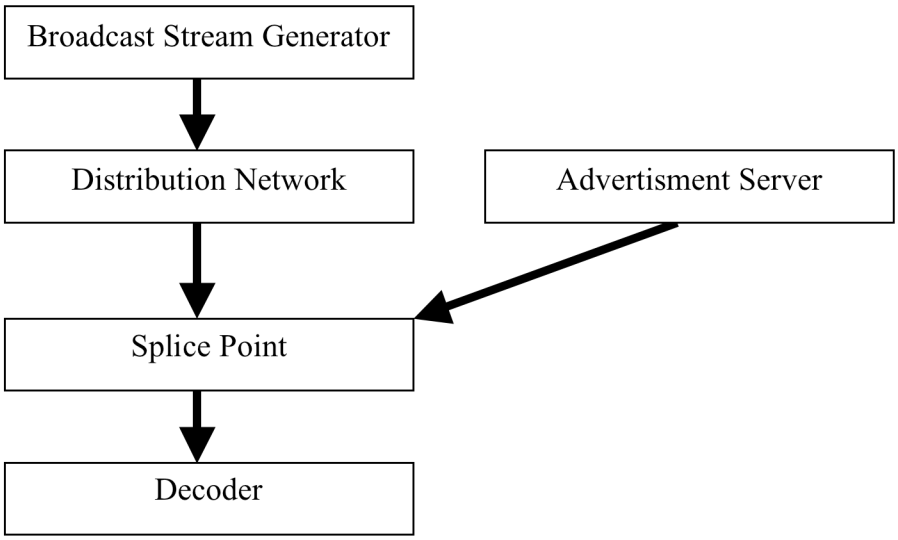
6 Architecture and Overview

6.1 Introduction

The basic structure of the system is that there is a *broadcast stream generator*, which makes a suitably tagged broadcast program stream of RTP packets [RFC3550].

This is reflected through a suitable distribution network (which might be as simple as a multicast backbone). Either at some point in the network (typically at an edge-server), or in the client, advertisements are spliced into the stream. This point is called the *splice-point* in this document. The advertisement is supplied from an *advertisement server* (which may be co-located with other servers, including the edge server). The advertisement may also be tagged to give a warning before it ends.

This specification covers both edge-server and client splice-points. A given network deployment should choose which is to occur. Hybrid architectures are also possible, where edge-servers do splicing anyway, and ad-aware clients re-do for more personal effects; or network nodes specialize the broadcast advertisements, e.g. for geographic areas, which are then replaced again by clients.



Advertisements typically come in standard 'lengths' (duration, such as 30 seconds). In the case where there are multiple possible lengths, the URL can identify the length of the slot that occurs. The length is also explicitly identified when the advertisement break starts.

The splice point splices in a suitable advertisement for each client, into the RTP stream. The stream decoder in the client is unaware that the splicing operation has occurred. At the end of the advertisement break, the splice point returns to forwarding the broadcast stream.

There are tag-commands in the stream. The command set is general, and may be acted on in any place. This specification covers specifically client and edge-server insertion in detail; however, tags are provided to signal other kinds of advertisement break (national, regional, and so on).

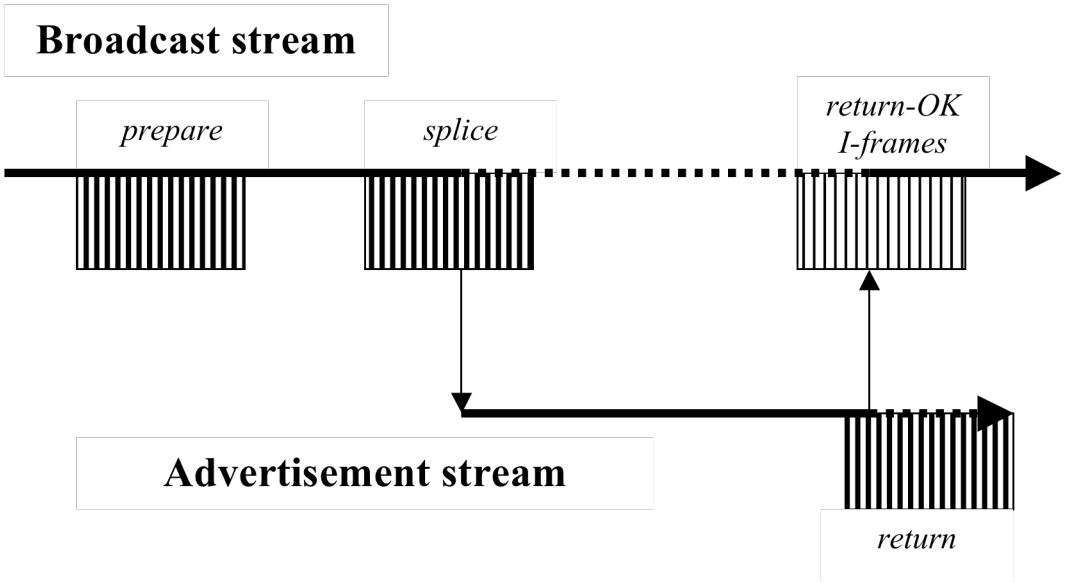
For any break, in the broadcast stream, before the insert is to happen, there is a "*prepare*" signal. This is repeated in several RTP packets, in case any are lost. It tells the splice-point the 'type' of advertisement that should be inserted, and roughly how long remains before the ideal splice time. The splice-point, for each client, resolves that URL supplying the cookies and other identity information it has.

A short while later, a repeated signal in the broadcast stream says "*splice*". Again this is repeated to be loss resilient; each instance marks the time difference between its occurrence and the ideal splice time. The splice-point stops forwarding program packets, and instead, splices in the advertisement for each stream, re-numbering the advertisement RTP packets so that they are in sequence.

At the end of the advertisement, there is a repeated "*return*" indication. The splice-point scans the broadcast stream looking for I-frames (random access points) which are also labeled as "*return OK*", and at the first one, drops the advertisement stream and returns to the broadcast (program) stream, continuing to re-number to make the sequence numbers contiguous for the client. If the end of the advertisement stream is reached without an I-frame, then the splice-back happens anyway, with consequential video break-up on the client.

In a broadcast with more than one stream (e.g. audio and video), the video stream should carry the splicing information. If neither stream is video, then a suitable stream should be chosen. In the remainder of this document, only one stream is described; it should be understood that associated audio is switched at the same time. It is assumed that at most one stream is differentially coded (has I frames and P or B frames), and that this is the stream that is tagged.

The diagram below shows the temporal structure of the splicing operation done at the splice point.



6.2 Edge-server Splicing

Edge-server splicing has the following advantages:

1. No changes to the terminal necessary; the resulting stream looks like a continuous broadcast (except for buffer model and tight bandwidth management). This means that 'legacy' set-top or other terminal devices may be used.
2. The 'last hop' network link from the edge-server to the client only carries a single stream. This may be the most bandwidth-constrained link.

Edge-server insertion has the following disadvantages:

1. Only the basic client identity known to the edge-server can be forwarded to the advertisement server. There is no opportunity for the advertisement server to supply and use 'cookie' information directly from the client.
2. Consequently, different accounts etc. at the client will get the same advertisement; personalization is not possible.
3. The program and advertisement have to use identical encodings and setup information.
4. The edge-servers are burdened both by the traffic and the splicing logic.
5. Tight buffer-model management across the splice point might not be possible; buffer-model violations might occur at the decoder.

Buffer fullness requirements can be signaled with a buffer fill tag in header extension, as outlined in the ISMA fast tune-in document [ISMAFRTP].

6.3 Client Splicing

Client insertion has the following advantages:

1. The selection of advertisement can use full domain-specific 'cookie' information, which is also user-specific on systems having separate accounts for separate users.
2. Both RTP [RFC3550] and HTTP [RFC2616] delivery of advertisements is possible.
3. The client terminal is aware of the stream switch and could possibly tolerate re-initializing decoders (though there is a possibility of glitches if this happens).
4. The edge-server load is smaller (at least in CPU).

Disadvantages of Client side insertion

Client-side insertion has the following disadvantages:

1. The client terminal must handle insertion (new set-top boxes).
2. The clients are more complex and require more buffering and processing. The client has to receive and monitor at least two streams in parallel.
3. The network out to the client must be able to carry both advertisement and program streams, in parallel. This may overload the last-hop link.

7 Specification

7.1 RTP Tag Structure

7.1.1 Common Structure

The streams are tagged using the RTP header extension technique specified in [HDREXT]. The set of advertisement URLs is supplied 'out of band' (possibly during stream setup), and indexed.

All the tags start with a 1-byte function code.

The codes used here fall into two ranges: instruction, and report. In the section specifying the splicing point behavior, the splicing point is required either to (a) rewrite packets containing tags, to remove the tags or (b) change all the operation instruction codes to report codes. A downstream system receiving the tags in the report range will know that at least one upstream system has already done the splicing, and that it may be (because of the splicing) that not all the needed tags for further splicing appear in the stream.

The 'instruction' range is 0-63; the 'report' range is 64-127. Values 128-255 are reserved.

The 'prepare' and 'splice' commands each occupy a range of codes; the 16 values in the range indicate the kind of advertisement break that is to occur [SCTE35]. Half of this range is further supported by indicating a URL to the advertisement server. This URL is typically (but is not required to be) a 'dynamic' URL (e.g. CGI) that carries the appropriate parameters for the server to interpret and select an advertisement.

The assigned function codes are:

Code Name	Instruction	Report
Prepare	0-15	64-79
Splice	16-31	80-95
Return	62	126
Return-OK	63	127

The 16 values for the prepare and splice signals indicate the break-type; values 0-7 indicate that the break is further supported by a URL. The assigned meanings for the break-types are TBD, and something like:

Break Type	Code
Client-specific	0
Local	8

Regional	9
National	10

In a given deployment, the required distance between the 'prepare' and splice, the length of each signal, the duration of the advertisement break, and the number of I-frames in the 'Return OK' interval, should be specified. Clearly these should be set so that variations in server performance, packet loss etc., still allow the system to function as intended.

7.1.2 Prepare

One or two bytes follow the operation code.

The first byte is an unsigned integer giving the length of the advertisement break, in seconds.

If the break-type indicates it is URL-supported, the second byte is an unsigned integer providing the URL index value, and indicates which advertisement URL should be used during this advertisement break. The index value ranges from 1 thru 255. The value 0 is reserved to indicate that no URL is yet assigned; two bytes following the function code, where the URL index is 0, are equivalent to one following byte with no URL index.

7.1.3 Splice

Two bytes follow the operation code.

The first byte is an unsigned integer giving the length of the advertisement break, in seconds.

The second byte is a signed integer providing a time-offset, in tenths of a second, of the time difference between the RTP time of the packet containing this splice signal, and the RTP time of the packet containing the 'splice' signal with a zero time-offset. There must be one splice signal in the stream with a zero time-offset. Both positive and negative values may be used in splice signals.

7.1.4 Return

The return signal has no other content than the operation code. It is placed at the end of the advertisement stream, as soon as a return to the program stream is permissible, and is repeated from there to the end of the advertisement stream.

Note that the return signal might not be present (an untagged stream or file); instead, the splice point MUST time the end of the break using the duration and time-offset fields in the 'splice' instruction.

7.1.5 Return OK

The return-OK signal has no other content than the operation code. It is placed in the broadcast stream on I-frames (sync. points) at which a 'clean entry' to the broadcast stream may be effected. It should be repeated on several I-frames.

7.1.6 Other signals

To support richer modes of operation, other signals are also possible. For example, stream switching can be supported by a command like 'splice' but without a return expectation ('switch').

For inter-program ads, where the expectation is that a new program will start on the 'return' signal, another variant on 'splice' is possible: 'splice-and-switch'. This command acts as a splice, but has an extra parameter, the URL index of the program to switch to on return. Since this command also has the length of the ad break in it, it serves as the prepare signal for the program switch at the end of the break. When the

return signal is found at the end of the break, the new program is switched to (no 'return-ok' signal is needed if this is an on-demand stream).

7.2 Signaling

7.2.1 SDP

The header extension is declared in SDP using the SDP signaling defined in [HDREXT]. The URI used to declare this header extension is <http://www.isma.tv/rtpheaderext/adinsert>.

The `extensionattributes` syntax element, for this extension, gives the URI of the configuration file. If it is absent, then the stream has already been spliced (and in this case only 'report' indications may appear in the stream).

Example (this should appear all in one line in SDP):

```
a=extmap:2/sendonly http://www.isma.tv/rtpheaderext/adinsert
http://www.adblast.com/ismaconfig.txt
```

7.2.2 Configuration file

The configuration file contains a series of lines. Each line starts with a verb and is followed by parameters.

In this version of the specification, only one verb is defined: URL. The parameters for this verb are (a) the index value for this URL and (b) an advertisement URL; the verb and these parameters are separated by whitespace. The index value ranges from 1 thru 255; the value 0 is reserved.

If the splice point is the terminal (i.e. splicing and decoding are co-located), the advertisement URLs MAY be HTTP URLs to MP4 files, or MAY be RTSP URLs for RTP streams.

If the splice point is in network, the advertisement URLs MUST be HTTP URLs to MP4 files, or MAY be RTSP URLs for RTP streams.

Example:

```
URL 1 rtsp://www.ads.com/euroads/afternoon?family=yes
URL 23 http://www.example.com/worldads/latenight?family=no
```

7.3 File format support

These commands may be stored in a track of the MP4 file format. A timed meta-data track is used, as specified in amendment 1 of the ISO base media file format. The sample entry name (which is not yet registered with the MP4 registration authority) should be 'isad' (ISMA Ad Insertion), and the track should be linked to the video track of the content, if any, or else the audio track.

The sample format is exactly the same as the command format defined above.

One extra box is placed in the same entry, providing the URL of the configuration file.

```
aligned(8) class AdConfigBox extends FullBox('isac', version = 0, 0) {
   string URL;
}
```

The URL is a null-terminated string in UTF-8 characters.

7.4 RTSP and HTTP capability

The splice-point must support 'cookies' [RFC2965] over RTSP [RFC2326] (even though this is not part of the RTSP specification, the cookie specification can be applied directly to RTSP).

In the case of edge-server insertion, the edge server SHOULD supply the client ID, in some way that may be deployment-specific, to the advertisement server.

7.5 Network

The network carries both program material and (pre-buffered) advertisement material, up to the splice point. It MUST have the bandwidth to supply both streams without congestion. In the case of client-side insertion, this includes the last-hop link.

7.6 Broadcast stream generator

The broadcast stream generator MUST insert matching, in-sequence, prepare, splice, and return-OK signals at the appropriate intervals. When a signal is repeated, only the timing estimate may change.

The recommended intervals are:

a) the prepare signal should occur in the range 6-4 seconds before the zero splice point;

b) the splice signal should occur from 0.75 seconds before the zero splice point, to 0.25 seconds after;

c) for an advertisement break of X seconds (e.g. a 30-second 'spot'), the return-OK signal should occur on I-frames from 1 second early (X-1) up to 5 seconds 'late' (X+5).

The encoding and SDP signaling of the broadcast stream and the advertisement streams MUST be identical – same codecs, screen size, audio sampling rate and channel count, and so on – for the splice to 'invisible' to the decoder.

There MUST be suitable watchable material in the broadcast stream (e.g. an broadcast advertisement) throughout the advertisement break. That material should be 'interruptible' for the 1-second interval symmetrically around the splice and return-OK zero-points (e.g. a still screen).

7.7 Advertisement Server

For deployments using edge-server (in-network) splicing, the advertisement server MUST supply advertisement streams over RTP using RTP control. This may be after a RTSP REDIRECT response (typically 'moved temporarily')..

The encoding and SDP signaling of the broadcast stream and the advertisement streams MUST be identical – same codecs, screen size, audio sampling rate and channel count, and so on – for the splice to be 'invisible' to the decoder.

In RTP advertisement streams, the stream SHOULD contain the 'return-OK' signal. If it is absent, the client returns at the expected ad-break length, if known, or else the end of stream (which may cause visual breakup if the stream to be entered or returned to is not at an I frame).

7.8 Splice Point

As soon as the SDP is received at the splice point, the splice point should fetch the configuration file. The SDP file MAY be re-written to remove the configuration file URL.

The splice point SHOULD re-write all the in-stream tags to the report range; it MUST do this re-write if it re-wrote the SDP. It SHOULD not re-write the tags if the splice operation is not performed.

At the prepare signal, the splice point should check the time-estimate, the amount of buffering available at the splice point, and the expected stream-opening latency. The splice point SHOULD have enough buffering for the expected latency; if it does not (e.g. an overloaded edge-server) it SHOULD not attempt the splice. As soon as there is sufficient buffer space available, the splice point should open the

advertisement stream and start to buffer it. It MUST be prepared for HTTP or RTSP re-directs to result from opening the URL.

At the first 'splice' instruction with a time-estimate of zero or less, the splice point MUST splice in the advertisement stream. This involves forward the packets (if RTP) or a/v content (if HTTP) downstream. The timestamps and sequence numbers (e.g. in RTP) MUST be re-written so that the advertisement stream has its first decodable content at the zero splice point.

As soon as the return-OK tags are seen in the advertisement stream the splice point MUST scan the broadcast stream for 'return-OK' signals. At the first of these seen after a 'return' signal has been detected, the splice point MUST drop the advertisement stream and return to forwarding the broadcast stream, re-writing timestamps and sequence numbers to make a continuous stream.

7.9 Decoder (terminal)

It is impossible to ensure a tight decoder buffer model across the splices (e.g. the H.264 HRD). The client has to have a large enough buffer that buffer-model mismatches across the splice points do not matter.

Similarly, it is possible that across the splice there is a slight deviation from the intended short-term averaged bit-rate. The channel needs to cope with that.

The decoder MUST ignore any advertising insertion RTP header extensions present.