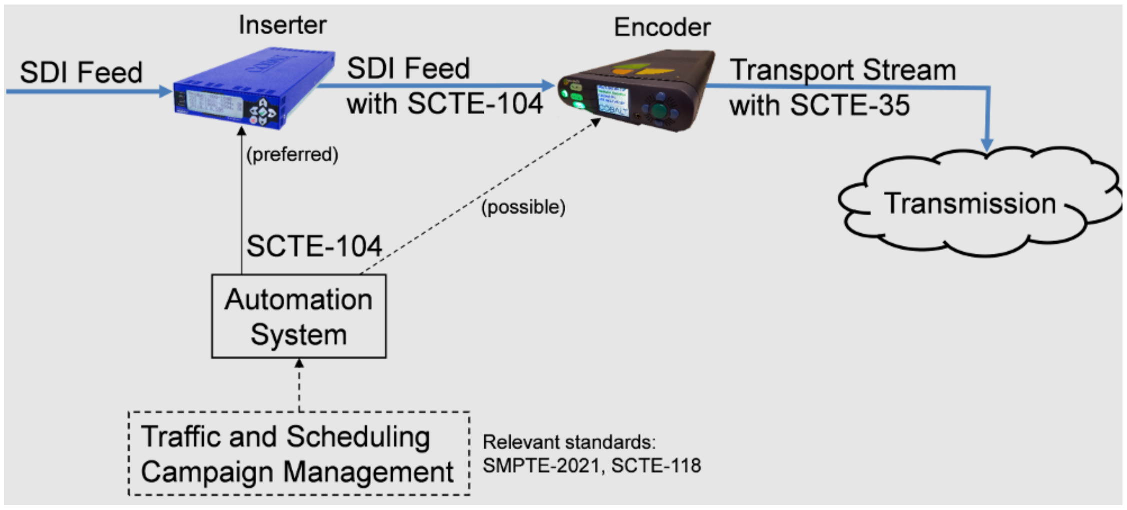
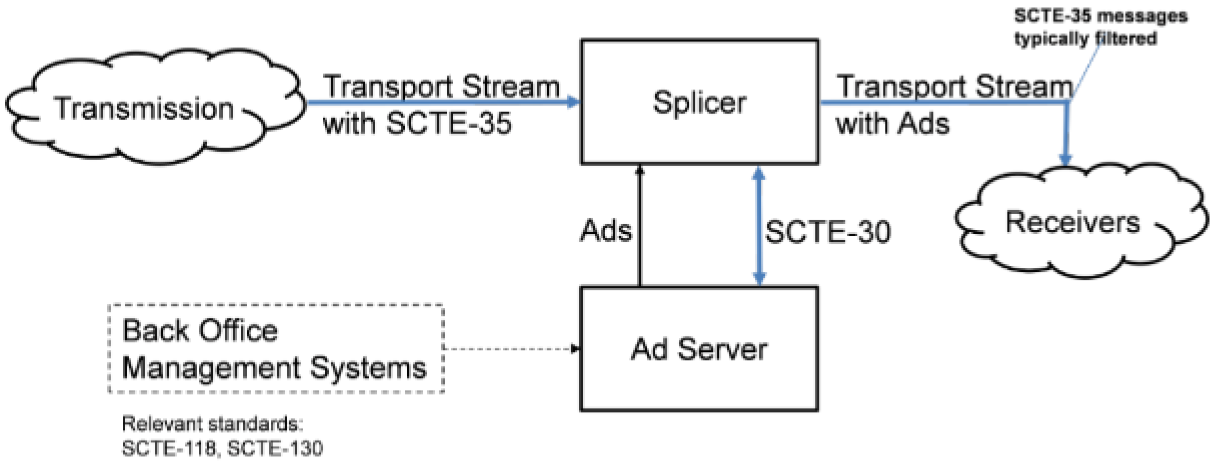
Inserts an SCTE-104/35 AD Marker into a .m3u8 file

There are two differents identifying markers aka SCTE-104, SCTE-35 that created for distinguishing media stream like the original broadcast and advertisement. SCTE-104 is mainly created at SDI Feed and SCTE-35 is created at Encoder.

The SDI feed from the network goes into an inserter that will add the SCTE-104 markers in the VANC using SMPTE-2010.The inserter does this under the control of an automation/playout system.The interface between the automation system and the inserter is also defined by SCTE-104.The output of the inserter is an SDI signal "decorated" with the SCTE-104 markers.An encoder converts this SDI signal into a compressed bitstream; the SCTE-104 markers are translated into SCTE-35 sections in the bitstream.It is also possible to skip the inserter and have the automation system directly control the encoder (again using the SCTE-104 network interface protocols), but this is discouraged by the standards.


Inserter
Encoder
SDI Feed
SDI Feed with SCTE-104
Transport Stream with SCTE-35
(preferred)
(possible)
Transmission
SCTE-104
Automation System
Traffic and Scheduling Campaign Management
Relevant standards: SMPTE-2021, SCTE-118

The local affiliate receives the transport stream decorated with SCTE-35 markers from the programmer side. This transport stream goes into a splicer, which reads the markers, contacts an Ad Server using the SCTE-30 protocol to request a replacement ad, and splices the replacement ad in the correct place in the transport stream. The output of the splicer is a new transport stream with the national low-priority ad replaced by a local ad. The splicer will also typically filter out the markers (so as not to facilitate the work of "commercial killers"). Back office management systems will interact with the ad server to facilitate the selection of the ad to be played, and to record the fact that the ad has been played for tracking purposes.


SCTE-35 messages typically filtered
Transmission
Transport Stream with SCTE-35
Splicer
Transport Stream with Ads
Receivers
Ads
SCTE-30
Back Office Management Systems
Ad Server
Relevant standards: SCTE-118, SCTE-130

SCTE-35 is a standard used in the cable television industry to signal the insertion of targeted advertising content within a video stream. The SCTE-35 ad marker is used to indicate the start and end of an ad insertion opportunity within a live stream.

- In HLS (HTTP Live Streaming), SCTE-35 ad markers are inserted into the .m3u8 playlist file that is used to serve the video stream. The markers indicate the location of an ad break within the stream, and the ad server can then dynamically insert an ad into that break.
- By including SCTE-35 markers within the .m3u8 playlist, it allows for the live stream to be dynamically updated with new ads, providing a flexible and scalable solution for targeted advertising in live streams.

The below code reads a `.m3u8` playlist file, extracts the times of the segments, and sorts them into an array of `AdMarker` structures. The code then scans the playlist, and whenever the time of the next ad marker is less than the time of the current segment, an SCTE-104 ad marker `#EXT-X-CUE-OUT` is inserted into the output file, with a duration of 30 seconds and a URI pointing to the next segment.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

const std::string input = "input.m3u8";
const std::string output = "output.m3u8";

struct AdMarker {
  double time;
  std::string url;
};

bool operator<(const AdMarker& a, const AdMarker& b) {
  return a.time < b.time;
}

int main()
{
    std::vector<std::string> playlist;
    std::vector<AdMarker> markers;

    // Read playlist
    std::ifstream file(input);
    std::string line;
    while (std::getline(file, line))
    {
        playlist.push_back(line);
        if (line[0] != '#')
        {
            // Extract time and add to markers
            size_t pos = line.find(",");
            double time = std::stod(line.substr(0, pos));
            markers.push_back({time, line});
        }
    }

    // Sort markers by time
    std::sort(markers.begin(), markers.end());

    // Insert ad markers
    std::ofstream stream(output);
    for (const auto& line : playlist)
    {
        stream << line << "\n";
        if (line[0] != '#')
        {
            // Check if it's time to insert ad marker
            size_t pos = line.find(",");
            double time = std::stod(line.substr(0, pos));
            while (markers.size() > 0 && markers.front().time < time)
            {
                stream << "#EXT-X-CUE-OUT:DURATION=30,URI=" << markers.front().url << "\n";
                markers.erase(markers.begin());
            }
        }
    }

    return 0;
}
```

The below code reads an input `.m3u8` playlist file into memory, and then generates an output playlist file. After writing the header, the code iterates over the segments in the input playlist, writing each segment to the output playlist, followed by the SCTE-35 ad marker. The SCTE-35 ad marker in this example is a hard-coded string, but it can be replaced with a dynamic value based on your specific use case.

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

const std::string input = "input.m3u8";
const std::string output = "output.m3u8";
const std::string ad_marker = "#EXT-X-SCTE35:<SCTE-35 AD MARKER>\n";

std::vector<std::string> read_playlist(const std::string& filename)
{
    std::vector<std::string> playlist;
    std::ifstream file(filename);
    std::string line;
    while (std::getline(file, line))
    {
        playlist.push_back(line);
    }
    return playlist;
}

int main()
{
    // Read playlist
    auto playlist = read_playlist(input);

    // Create output playlist
    std::ofstream stream(output);

    // Write header
    stream << "#EXTM3U\n";

    // Write segments
    for (const auto& segment : playlist)
    {
        stream << segment << "\n";
        stream << ad_marker;
    }

    return 0;
}
```