mod_fsv

About

mod_fsv implements functions to record and play back RTP video frames in FreeSWITCH.

- play_fsv
- record_fsv

Format Interface

You could use the file interface for audio only record/playback

Dialplan app:

```
<action application="record" data="/tmp/testrecord.fsv">
```

API:

```
 uuid_record <uuid> start /tmp/testrecord.fsv
```

uuid_record also can record video, but at this time it only records one leg of audio and video.

Recording in conference is also possible with:

```
conference 3000 record /tmp/testrecord.fsv
```

File Format

FSV probably means "FreeSWITCH Video". An FSV file is a native format to store audio (L16) and video (codec native) frames as seen inside RTP packets. The format is machine-dependent; especially there seems to be no allowance in the code for endian-ness (so files recorded on a big-endian machine for example would not work on a little-endian machine, etc).

The file header consists of a 4 byte version ID, 32 byte IANA video codec name (presumably RFC4855 and similarly-registered names), 128 byte video format descriptor (the content of the SDP format-specific parameters "fmtp" attribute field), 4 byte audio rate, 4 byte audio packetization time, and a creation time tag.

The frames are stored in the order received. A frame header, presumably 4 bytes in length (but stored as an "int", in native endian), is used; the top bit is one if frame is video, zero if audio; the 31 lower bits are used to store the frame length.

```
struct file_header {
        int32_t version;
        char video_codec_name[32];
        char video_fmtp[128];
        uint32_t audio_rate;
        uint32_t audio_ptime;
        switch_time_t created;
};
```

Conversation About Packetization and File Formatting

I'm going to come back and clean this up when i have a little more time to think about how to word it as documentation. – Chris Tunbridge Destreyf

Related Conversation:

irc #freeswitch

```
[14:38:27] <SwK> j0sh: fsv just strips the frames out and writes them to disk
[14:40:44] <j0sh> SwK: so it does a full depacketization? i didn't see any codec-specific RTP parsing, but
maybe I'm missing it
[14:42:16] <MikeJ> j0sh: rtp is already parsed out.. its just using the pointer to the data payload directly
[14:43:38] <j0sh> MikeJ: yes, but each codec has its own specific RTP format (aside from the standard RTP
headers), it seems those codec specific headers are included in the disk format
[14:44:11] <MikeJ> iirc we put our own header on their.. not the rtp header.. let me peek
[14:45:41] <j0sh> MikeJ: yeah I read the length/tag byte, was trying to feed the payload to ffmpeg but it isn't
working, the leftover RTP data for the codec is my guess as to why it's not working
[14:48:25] <MikeJ> look at struct file_header
[14:48:32] <MikeJ> that's OUR header we write to the file
[14:51:29] <MikeJ> so you've got 12 byte headers
[14:51:42] <MikeJ> per frame
[14:51:45] <MikeJ> and then the payload
[14:52:55] <j0sh> yeah it's not clear if the payload has codec specific RTP headers still in it
[14:53:40] <bkw_> it would, its a dumb record playback app
[14:54:07] <MikeJ> that's the 12
[14:55:45] <j0sh> MikeJ: isn't the fsv type/length tag just 4 bytes per frame?
[14:59:07] <j0sh> because if that is the case (which i suspect), the original 12-ish rtp header bytes have to
be re-added before feeding it into the ffmpeg rtp demuxer
[14:59:31] <MikeJ> re added?
[14:59:43] <MikeJ> are they not already there?
[15:05:04] <j0sh> but i see now that the original rtp headers should still be there (looking at the code now)
[15:05:18] <MikeJ> exactly!!
[15:05:39] <MikeJ> so you read the file header..
[15:05:47] <MikeJ> then you read 12 bytes + payload size
[15:05:57] <MikeJ> and that should be a complete rtp frame
```

Other

Try calling 9193 and 9194 in the 'vanilla' dialplan with a video phone.