```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++

Copyright (c) 2001 BeComm Corporation

Filename:

    speaker.c

Abstract:

    Bead to send audio to an output device.

    TODO - need to add
    - RTP (or other) format decryption
    - pause/stop/resume control edge
    - queue management (?)
    - device configuration

Owner:

    Guy Carpenter (guyc)

----------------------------------------------------------------------------
---*/

#define SOS_DEBUG_ZONE "/beads/speaker"

#include <sosstrings.h>
#include <sosmultimedia.h>

SOS_SOURCE_VERSION (
    "$Id: speaker.c,v 1.25 2001/10/22 23:28:14 guyc Exp $"
);

#define SPEAKER_BEAD_NAME "Speaker"
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Configuration
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++*/

// REVISIT - device name should not be hard coded.
#define DEVICE_CLASS "audioout"
#define DEVICE_NAME "default"
```

```
42
43
44  /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++++
45  Types
46  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++*/
47  typedef struct _SPEAKER_CONTEXT {
48      SOS_REGOBJECT *         AudioOutDevice;
49      SOS_IAUDIOOUT *         IAudioOut;
50      SOS_AUDIO_FORMAT        Format;
51      SOS_IAUDIOCONTEXT *     IAudioContext;
52      char *                  DeviceName;
53      SOS_BOOLEAN             FormatIsSet;
54      SOS_ISAMPLECLOCK *      ISampleClock;
55  } SPEAKER_CONTEXT;
56
57
58  /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++++
59  Context
60  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++*/
61  SOS_STATUS
62  Speaker_ContextCreate(
63      SPEAKER_CONTEXT **      Context
64  )
65  {
66      SOS_STATUS status = SOS_Success;
67      SPEAKER_CONTEXT *context = SOS_Mem_Alloc(sizeof(*context));
68      if (context) {
69          SOS_memset(context, 0, sizeof(*context));
70          // REVISIT - this should come from the path context
71          context->DeviceName = SOS_strdup(DEVICE_NAME);
72      } else {
73          status = SOS_ErrorResourceAllocation;
74      }
75      *Context = context;
76      return status;
77  }
78
79  SOS_STATUS
80  Speaker_ContextDestroy(
81      SPEAKER_CONTEXT *       Context
```

```
{
    SOS_STATUS status = SOS_Success;

    if (Context) {
        if (Context->AudioOutDevice) {
            SOS_RegObject_Release(Context->AudioOutDevice);
        }
        if (Context->IAudioContext) {
            SOS_Interface_Release(Context->IAudioContext);
        }

        if (Context->ISampleClock) {
            SOS_Interface_Release(Context->ISampleClock);
        }

        if (Context->DeviceName) {
            SOS_Mem_Free(Context->DeviceName);
        }

        SOS_Mem_Free(Context);
    }
    return status;
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Local Functions
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++*/
SOS_STATUS
Speaker_DeviceOpen(
    SPEAKER_CONTEXT *      Context
)
{
    SOS_STATUS status;

/*
 * NOTE : if we return an error, make sure the device is not
 * still opened or referenced.
 */
    status = SOS_Registry_DeviceGet(
        DEVICE_CLASS,
        Context->DeviceName,
        &Context->AudioOutDevice);
```

```
126     if (SOS_SUCCEEDED(status)) {
127         status = SOS_RegObject_InterfaceGet(
128             Context->AudioOutDevice,
129             SOS_IAUDIOOUT_ID,
130             (void**)&Context->IAudioOut
131         );
132     }
133 
134     if (SOS_SUCCEEDED(status)) {
135         status = Context->IAudioOut->Open(
136             Context->IAudioOut,
137             &Context->Format);
138     }
139 
140     if (SOS_FAILED(status)) {
141         SOS_RegObject_Release(Context->AudioOutDevice);
142         SOS_Interface_Release(Context->IAudioOut);
143         Context->AudioOutDevice = NULL;
144         Context->IAudioOut = NULL;
145     }
146 
147     return status;
148 }
149 
150 
151 SOS_STATUS
152 Speaker_DeviceClose(
153     SPEAKER_CONTEXT *      Context
154 )
155 {
156     SOS_STATUS status = SOS_Success;
157 
158     if (Context->IAudioOut) {
159         status = Context->IAudioOut->Close(
160             Context->IAudioOut
161         );
162         SOS_RegObject_InterfaceRelease(Context->IAudioOut);
163         Context->IAudioOut = NULL;
164     }
165 
166     SOS_RegObject_Release(Context->AudioOutDevice);
167     Context->AudioOutDevice = NULL;
168 
169     return status;
```

```
171
172
173 SOS_STATUS
174 Speaker_DeviceWrite(
175     SPEAKER_CONTEXT *      Context,
176     SOS_MESSAGE *          Message
177 )
178 {
179     SOS_STATUS status;
180     int byteCount = SOS_Message_LengthGet(Message);
181     char *buffer = SOS_Mem_Alloc(byteCount);
182
183     if (buffer) {
184         SOS_Message_DataRemove(
185             Message,
186             0,
187             byteCount,
188             buffer,
189             NULL);
190
191 #if 0
192         {
193             SOS_UINT32 queuedSamples;
194             Context->IAudioOut->QueuedSamplesGet(
195                 Context->IAudioOut,
196                 &queuedSamples
197             );
198             SOS_Debug_StringPrint(
199                 "%d queued samples in speaker before writing %d\n",
200                 queuedSamples,
201
… byteCount/Context->Format.Channels/(Context->Format.SampleBits/8)
202             );
203         }
204 #endif
205
206         status = Context->IAudioOut->Write(
207             Context->IAudioOut,
208             0, /* no delay */
209             buffer,
210             byteCount
211         );
212
213         SOS_Mem_Free(buffer);
```

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.