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

$Id: mp3decoder.c,v 1.20 2001/10/26 16:16:04 guyc Exp $

Copyright (c) 2001 BeComm Corporation

Filename:

    mp3decoder.c

Abstract:

    Wrapper for the mpegaudiodecoder class.

    This could/should become a general-purpose codec filter
    for filters implemented via the SOS_IAUDIODECODE interface.

Owner:

    Guy Carpenter (guyc)

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

#define SOS_DEBUG_ZONE "/beads/mp3decoder"
#include <sosstrings.h>
#include <sosmultimedia.h>

SOS_SOURCE_VERSION("$Id: mp3decoder.c,v 1.20 2001/10/26 16:16:04 guyc Exp
$");

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Named Constants
--------------------------------------------------------------------------------
---*/

/*
 * Name of bead
 */
static const char BEAD_NAME[]   = "mp3decoder";

/*
```

```
 */
#define DECODER_CLASS_NAME "mpegaudiodecoder"

/*
 * Name of audio context class
 */
#define AUDIOCONTEXT_CLASS_NAME "pcmcontext"

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Structs
----------------------------------------------------------------------------
---*/

typedef struct _MP3_CONTEXT {
    SOS_IAUDIODECODE*  IAudioDecode; /* Interface to an audio decoder */
    SOS_BOOLEAN        FormatIsSet;  /* A flag used to determine if the
                                        AudioFormat path attribute has
                                        been set yet*/
    SOS_IAUDIOCONTEXT *IInAudioContext;
    SOS_IAUDIOCONTEXT* IOutAudioContext;/* Propagates the format of the
PCM
                                           audio stream down the path. */
    SOS_UINT32         BytesPerSample;
    SOS_UINT32         SampleCount;
} MP3_CONTEXT;

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Helper Functions
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++*/

static
void*
Mp3Decoder_PathContextInterfaceGet(
    SOS_PATH*              Path,
    const char *           Name,
    SOS_INTERFACE_ID       InterfaceId
)
{
    SOS_STATUS status;
    SOS_REGOBJECT *object;
    void *interface = NULL;
```

```
81      status = SOS_Path_AttributeGet(
82          Path,
83          Name,
84          &object
85      );
86
87      if (SOS_SUCCEEDED(status)) {
88          status = SOS_RegObject_InterfaceGet(
89              object,
90              InterfaceId,
91              &interface
92          );
93          SOS_RegObject_Release(object);
94      }
95
96      return interface;
97  }
98
99  static
100 SOS_STATUS
101 Mp3Decoder_AudioContextCreate(
102     MP3_CONTEXT*          Context,
103     SOS_PATH*             Path
104 )
105 {
106     SOS_STATUS status = SOS_Error;
107     SOS_IAUDIOCONTEXT *iAudioContext = NULL;
108
109     /*
110      * If an audio context already exists we will use it to
111      * extract the timestamp and pass it along.
112      */
113     Context->IInAudioContext = Mp3Decoder_PathContextInterfaceGet(
114         Path,
115         SOS_AUDIOCONTEXT_NAME,
116         SOS_IAUDIOCONTEXT_ID
117     );
118
119     iAudioContext = SOS_Interface_CreateFromClassName(
120         AUDIOCONTEXT_CLASS_NAME,
121         SOS_IAUDIOCONTEXT_ID
122     );
123
124     SOS_ASSERT_ASSUMPTION(
```

```
126         "Couldn't create audio context"
127     );
128 
129     if (iAudioContext) {
130         status = SOS_Path_AttributeSet(
131             Path,
132             SOS_AUDIOCONTEXT_NAME,
133             SOS_Interface_ObjectPeek(iAudioContext)
134         );
135     }
136 
137     Context->IOutAudioContext = iAudioContext;
138 
139     return status;
140 }
141 /*++
142 Routine Name:
143 
144     Mp3Decoder_FormatSet
145 
146 Routine Description:
147 
148     Attempts to set the AudioFormat path attribute for the Path that
149     is passed in.  The AudioFormat path attribute is used by
150     subsequent beads to determine the format the audio data they are
151     receiving is in.
152 
153 Parameters:
154 
155     MP3_CONTEXT* Context - [in]
156         The session context.
157 
158     SOS_PATH*    Path    - [in]
159         The path to set the attribute for.
160 
161 Return Value:
162 
163     SOS_STATUS -
164        SOS_Success if everything succeeded.
165 
166        SOS_True if more data is required to determine the audio format.
167 
168        other error codes are returned if an error occuerd
169 
```

```
171 static
172 SOS_STATUS
173 Mp3Decoder_FormatSet(
174     MP3_CONTEXT* Context,
175     SOS_PATH*    Path
176 )
177 {
178     SOS_AUDIO_FORMAT        format;
179     SOS_STATUS              status        = SOS_Success;
180 
181     SOS_DEBUGOUT_FUNC_TRACE("Mp3Decoder_FormatSet called\n");
182 
183     status = Context->IAudioDecode->FormatGet(
184         Context->IAudioDecode,
185         &format
186     );
187 
188     SOS_DEBUGOUT_MAJOR_EVENT(
189         "MP3 format is %u/%u/%u\n",
190         format.Frequency,
191         format.SampleBits,
192         format.Channels
193     );
194 
195     /* check for SOS_Success instead of SOS_SUCCCEEDED
196        because SOS_True is returned if the
197        decoder needs more data */
198     if (SOS_Success == status) {
199 
200         status = Context->IOutAudioContext->FormatSet(
201             Context->IOutAudioContext,
202             &format
203         );
204 
205         SOS_ASSERT_ASSUMPTION(
206             SOS_SUCCEEDED(status),
207             "Could not set format in audio context"
208         );
209 
210         if (SOS_SUCCEEDED(status)) {
211             Context->FormatIsSet = SOS_True;
212             Context->BytesPerSample =
… (format.SampleBits+SOS_BITSPERBYTE-1)/
213                 SOS_BITSPERBYTE*format.Channels;
```

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.