```
/*

+-----------------------------------------------------------------
------+
 |    Copyright (c) 1996-1998 Vail Systems, Inc.  All Rights
Reserved    |

+-----------------------------------------------------------------
------+
 $Id: mc_vr.c,v 1.1 1999/04/03 17:26:21 alex Exp $

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

  Module : mc_vr.c

  This module holds Webley ASR code.

  HISTORICAL INFORMATION:

  10-04-96  alex:Vail  Created

*****************************************************************
*************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <time.h>
#include <dirent.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <rpc/rpc.h>
#include <sys/fcntl.h>
#include <memory.h>
#include <setjmp.h>
#include <unistd.h>
#include <math.h>
#include <errno.h>
#include "vlib_scsa.h"
#include "notif_scsa.h"
#include "job.h"
#include "parus_api.h"
#include "app_api.h"
```

```
#include "mcall.h"

#undef  DEBUG
#define DEBUG(x)  x

static int vr_allocated = 0;
static int nuance_flag = 0;
static int nuance_checked = 0;
static char nuance_package[MAX_STRING_SIZE + 1];
static int contacts_rsrc = -1;
static int load_contacts_flag = 0;
static int load_corp_flag = 0;
static int load_lists_flag = 0;
extern int trunk_num;
extern char app_name[];
extern char app_home[];
extern char msg_home[];
extern char msg_host[];
extern char *convertPhone();
extern char *vr_convert();
extern char *processPrompt();
extern char *processIniName();
int vrAllocNuance(int, char *);
int vrGoNuance(char *, int *, char *, char *, int,int,struct
VR_RES *,int,int);
static int vrResNuance(struct VR_RES *, struct ASR_RES *, char
*);
static int vrGrammarDynamic(char *, char *, char *);
/
* ----------------------------------------------------------------
----------- */
int vrNuance()
{
char *np;

      if (!nuance_checked) {
            nuance_checked = 1;
            np = GetConfigVar(app_name, "NUANCE");
            if (!np)
                  nuance_flag = 0;
            else if (!strcmp(np, "YES") || !strcmp(np, "yes"))
                  nuance_flag = 1;
            else
                  nuance_flag = 0;
      }
      return(nuance_flag);
}
/
* ----------------------------------------------------------------
----------- */
int vrExit()
{
```

```
register int ret_val;

      if (vrNuance())
            return(vrExitNuance());

      vrDealloc();
      DEBUG(printf("<%d>vrExit() ret = %d\n", trunk_num,
ret_val));
      return 0;
}
/
* ----------------------------------------------------------------
---------- */
int vrDealloc()
{
register int ret_val;

      if (vrNuance())
            return(vrDeallocNuance());

      if ( vr_allocated ) {
            ret_val = app_dealloc_vr_psp();
            DEBUG(printf("<%d>vrDealloc() ret = %d\n", trunk_num,
ret_val));
            vr_allocated = 0;
      }
      return 0;
}
/
* ----------------------------------------------------------------
---------- */
int vrAlloc(int vr_type, char *path)
{
register int retries = 6, ret_val = R_FALSE;

      if (vrNuance())
            return(vrAllocNuance(vr_type, path));

      if ( !vr_allocated ) {
         /* Try to allocate voice recognition port */
             while (retries--) {
                  ret_val = app_alloc_vr_psp(path);
                 DEBUG(printf("<%d>vrAlloc(%d) ret = %d\n",
                  trunk_num, vr_type,ret_val));
            if (ret_val == R_TRUE )
                  break;
            sleep(1);
             }
               if (ret_val != R_TRUE ) {
                  fprintf(stderr,"<%d>ERROR :
vrAlloc()\n",trunk_num);
               }
```

```
                else
                   vr_allocated = vr_type;
        }
        else
              return R_TRUE;
           return ret_val;
}
/
* ------------------------------------------------------------------
---------- */
static int vrCheckStatus(struct VR_RES *vtk, char *voc, char *mn)
{
static int flag = 0;

      if (vrNuance())
            return(R_FALSE);

      fldtmf();
      DEBUG(printf("<%d>vrCheckStatus() result=%d\n", trunk_num,
                                          vtk[0].condition));
      switch( vtk[0].condition ) {
            case APP_VR_TIMEOUT:
                  break;
            case APP_VR_INT_TIMEOUT :
                  /*-->vr_too_slow
                   *-->Faster please
                   */
                        strcpy(mn, vr_too_slow);
                  break;
            case APP_VR_SPOKE_TOO_SOON:
                  /*-->vr_too_soon
                   *-->Sorry, I wasn't ready.  Please try again.
                   */
                        strcpy(mn, vr_too_soon);
                  break;
            case APP_VR_SPOKE_TOO_LOUD:
                  /*-->vr_too_loud
                   *--> Softer please
                   */
                        strcpy(mn, vr_too_loud);
                  break;
            case APP_VR_SPOKE_TOO_SOFT:
                  /*-->vr_too_soft
                   *-->Louder please
                   */
                        strcpy(mn, vr_too_soft);
                  break;
            case APP_VR_TALKED_TOO_LONG:
                  /*-->vr_too_long
                   *-->Please repeat, try not to speak too long
                   */
                        strcpy(mn, vr_too_long);
```

```
                break;
          default:
                if (  strcmp(voc, "yncqh" ) ) {
                          strcpy(mn, vp_no_result_alt);
                }
                else  {
                     return R_TRUE;
                }
                break;

     }
     return R_FALSE;
}
/
* ----------------------------------------------------------------
---------- */
int addTimeVR(int t)
{
static int timeInVr = 0;

     if ( t > 0 )
          timeInVr += t;
     return timeInVr;
}
/
* ----------------------------------------------------------------
---------- */
int speechStartTimeout(char *mode, int val)
{
static int speechTimeout = 6;
int ret_val;

     ret_val = speechTimeout;
     if (!strcmp(mode, "set")) {
          speechTimeout = val;
          ret_val = speechTimeout;
     }
     else if (!strcmp(mode, "get") ) {
          ret_val = speechTimeout;
          speechTimeout = 6;
     }
     return ret_val;

}
/
* ----------------------------------------------------------------
---------- */
int vrContPSP(char *msg, /* voice prompt filename; barge in or no
barge in */
      int *retp,      /* indicates if dtmf is expected on input
and
                   if a dtmf was actually entered on output
```

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.