```
/*=========================================================================
FILE: example.c  -- Template for BREW Program
=========================================================================*/

/*=========================================================================
INCLUDES AND VARIABLE DEFINITIONS
=========================================================================== */
#include "AEEModGen.h"          // Module interface definitions
#include "AEEAppGen.h"          // Applet interface definitions
#include "AEEShell.h"           // Shell interface definitions
#include "AEEFile.h"            // File interface definitions
#include "AEENet.h"             // Socket interface definitions
#include "AEESound.h"           // Sound Interface definitions

#include "example.bid"


/*-------------------------------------------------------------------
Applet structure. All variables in here are reference via "pMe->"
-------------------------------------------------------------------*/
// create an applet structure that's passed around. All variables in
// here will be able to be referenced as static.
typedef struct _example {
      AEEApplet      a ;       // First element of this structure must be AEEApplet
    AEEDeviceInfo  DeviceInfo; // always have access to the hardware device information
    IDisplay*      pIDisplay;  // give a standard way to access the Display interface
    Ishell*        pIShell;    // give a standard way to access the Shell interface

    // add your own variables here...

} example;

/*-------------------------------------------------------------------
Function Prototypes
-------------------------------------------------------------------*/
static  boolean example_HandleEvent(example* pMe, AEEEvent eCode,
                                                  uint16 wParam, uint32 dwParam);
boolean example_InitAppData(example* pMe);
void    example_FreeAppData(example* pMe);


/*=========================================================================
FUNCTION: AEEClsCreateInstance

DESCRIPTION
This function is invoked while the app is being loaded. All Modules must provide this
function. Ensure to retain the same name and parameters for this function.
In here, the module must verify the ClassID and then invoke the AEEApplet_New() function
that has been provided in AEEAppGen.c.

After invoking AEEApplet_New(), this function can do app specific initialization. In this
example, a generic structure is provided so that app developers need not change app
specific initialization section every time except for a call to IDisplay_InitAppData().
This is done as follows: InitAppData() is called to initialize AppletData
instance. It is app developers responsibility to fill-in app data initialization
code of InitAppData(). App developer is also responsible to release memory
allocated for data contained in AppletData -- this can be done in IDisplay_FreeAppData().

PROTOTYPE:
   int AEEClsCreateInstance(AEECLSID ClsId,IShell * pIShell,IModule * po,void ** ppObj)
```

```
PARAMETERS:
      clsID: [in]: Specifies the ClassID of the applet which is being loaded

      pIShell: [in]: Contains pointer to the IShell object.

      pIModule: pin]: Contains pointer to the IModule object to the current module to
which
      this app belongs

      ppObj: [out]: On return, *ppObj must point to a valid IApplet structure. Allocation
      of memory for this structure and initializing the base data members is done by
      AEEApplet_New().

DEPENDENCIES
  none

RETURN VALUE
  AEE_SUCCESS: If the app needs to be loaded and if AEEApplet_New() invocation was
     successful
  EFAILED: If the app does not need to be loaded or if errors occurred in
     AEEApplet_New(). If this function returns FALSE, the app will not be loaded.

SIDE EFFECTS
  none
===========================================================================*/
int AEEClsCreateInstance(AEECLSID ClsId, IShell *pIShell, IModule *po, void **ppObj)
{
      *ppObj = NULL;

      if( ClsId == AEECLSID_EXAMPLE )
      {
            // Create the applet and make room for the applet structure
            if( AEEApplet_New(sizeof(example),
                        ClsId,
                        pIShell,
                        po,
                        (IApplet**)ppObj,
                        (AEEHANDLER)example_HandleEvent,
                        (PFNFREEAPPDATA)example_FreeAppData) )
// the FreeAppData function called after sending EVT_APP_STOP to HandleEvent function
            {
                  //Initialize applet data, this is called before sending EVT_APP_START
            // to the HandleEvent function
                  if(example_InitAppData((example*)*ppObj))
                  {
                        //Data initialized successfully
                        return(AEE_SUCCESS);
                  }
                  else
                  {
                        //Release the applet. Free memory allocated for the applet when
                        // AEEApplet_New was called.
                        IAPPLET_Release((IApplet*)*ppObj);
                        return EFAILED;
                  }
        } // end AEEApplet_New
    }

      return(EFAILED);
}
```

```
/*===========================================================================
FUNCTION SampleAppWizard_HandleEvent

DESCRIPTION
      This is the EventHandler for this app. All events to this app are handled in this
      function. All APPs must supply an Event Handler.

PROTOTYPE:
      boolean SampleAppWizard_HandleEvent(IApplet * pi, AEEEvent eCode, uint16 wParam,
uint32 dwParam)

PARAMETERS:
      pi: Pointer to the AEEApplet structure. This structure contains information
specific
      to this applet. It was initialized during the AEEClsCreateInstance() function.

      ecode: Specifies the Event sent to this applet

   wParam, dwParam: Event specific data.

DEPENDENCIES
  none

RETURN VALUE
  TRUE: If the app has processed the event
  FALSE: If the app did not process the event

SIDE EFFECTS
  none
===========================================================================*/
static boolean example_HandleEvent(example* pMe, AEEEvent eCode, uint16 wParam, uint32
dwParam)
{
    switch (eCode)
      {
        // App is told it is starting up
        case EVT_APP_START:
                // Add your code here...

            return(TRUE);

        // App is told it is exiting
        case EVT_APP_STOP:
            // Add your code here...

                  return(TRUE);

        // App is being suspended
        case EVT_APP_SUSPEND:
                // Add your code here...

                  return(TRUE);

        // App is being resumed
        case EVT_APP_RESUME:
                // Add your code here...

                  return(TRUE);
```

```
        // An SMS message has arrived for this app.
        // Message is in the dwParam above as (char *)
        // sender simply uses this format "//BREW:ClassId:Message",
           example //BREW:0x00000001:Hello World
        case EVT_APP_MESSAGE:
                // Add your code here...

                  return(TRUE);

        // A key was pressed. Look at the wParam above to see which key was pressed.
        // Key codes found in AEEVCodes.h. Example "AVK_1" means the "1" key was pressed.
        case EVT_KEY:
                // Add your code here...

                  return(TRUE);


        // If nothing fits up to this point then we'll just break out
        default:
            break;
   }

   return FALSE;
}

// this function is called when your application is starting up
boolean example_InitAppData(example* pMe)
{
    // Get the device information for this handset.
    // Reference all the data by looking at the pMe->DeviceInfo structure
    // Check the API reference guide for all the handy device info you can get
    pMe->DeviceInfo.wStructSize = sizeof(pMe->DeviceInfo);
    ISHELL_GetDeviceInfo(pMe->a.m_pIShell,&pMe->DeviceInfo);

    // The display and shell interfaces are always created by
    // default, so we'll asign them so that you can access
    // them via the standard "pMe->" without the "a."
    pMe->pIDisplay = pMe->a.m_pIDisplay;
    pMe->pIShell   = pMe->a.m_pIShell;

    // Insert your code here for initializing or allocating resources...


    // if there have been no failures up to this point then return success
    return TRUE;
}

// this function is called when your application is exiting
void example_FreeAppData(example* pMe)
{
    // insert your code here for freeing any resources you have allocated...

    // example to use for releasing each interface:
    // if ( pMe->pIMenuCtl != NULL ) {        // check for NULL first
    //    IMENUCTL_Release(pMe->pIMenuCtl)   // release the interface
    //    pMe->pIMenuCtl = NULL;             // set to NULL so no problems trying to free
                                             // at a later time
    // }

}
```

```
/*=========================================================================

Menu Sample --

=========================================================================*/

// 1. include file

#include "AEEMenu.h"               // Menu Interface definitions


// 2. add space for menu object(s) in applet structure

typedef struct _CdemoApp
{
      AEEApplet   a;                     // first element must be AEEApplet
      IMenuCtl*   m_pIMainMenu;          // pointer to Menu object
      IMenuCtl*   m_pIMenu2;             // optional menu
} CdemoApp;



// 3a. In initialization code, create menu object

if (ISHELL_CreateInstance(pMe->a.m_pIShell, AEECLSID_MENUCTL,
                          (void **)&pMe->m_pIMainMenu) != SUCCESS){
            pMe->m_pIMainMenu = NULL;
            return FALSE;
            }
            else {

// 3b. Build Menu; use menu strings from resource file

IMENUCTL_AddItem(pMe->m_pIMainMenu, DEMO_RES_FILE,     // name of resource file
            IDS_MENU_CHOICE1, IDS_MENU_CHOICE1,        // resource id, menu id
            NULL, 0);                                  // keep ids the same

IMENUCTL_AddItem(pMe->m_pIMainMenu, DEMO_RES_FILE,  // add second item
            IDS_MENU_CHOICE2, IDS_MENU_CHOICE2,
            NULL, 0);

IMENUCTL_AddItem(pMe->m_pIMainMenu, DEMO_RES_FILE,  // add last item
            IDS_MENU_CHOICE3, IDS_MENU_CHOICE3,
            NULL, 0);


// 3c. Set menu attributes

IMENUCTL_SetTitle(pMe->m_pIMainMenu, DEMO_RES_FILE,     // set title contained
            IDS_MENU_TITLE, NULL);                      // in resource file

IMENUCTL_SetColors(pMe->m_pIMainMenu, ……);              // set colors
IMENUCTL_SetStyle(pMe->m_pIMainMenu, ……);               // set styles

Return TRUE;
}
```

```
// 4. In your Event Handler, to activate and display menu

case EVT_APP_START:     // for example – could be in other routines

IMENUCTL_SetActive(pMe->m_pIMainMenu, TRUE);            // activate


// 5. To deactivate and erase menu

IMENUCTL_SetActive(pMe->m_pIMainMenu, FALSE);           // de-activate menu
IDISPLAY_ClearScreen(pMe->a.m_pIDisplay);               // clear display


// 6. To Process Menu in the Event Handler

// standard event handler prototype

static Boolean HandleEvent (Iapplet * pi, AEEEvent eCode, unit16 wParam, uint32,
dwParam) {

CdemoApp *pMe = (CdemoApp *) pi;                // cast and copy pointer

case EVT_KEY:                                   // process key press
      if (IMENUCTL_IsActive(pMe->m_pIMainMenu)) {  // if menu is active

            // let menu event handler process key stroke
            if (IMENUCTL_HandleEvent(pMe->m_pIMainMenu, eCode, wParam, dwParam))
                  Return TRUE;
      }
      // process other keys yourself

case EVT_COMMAND:                               // process select

      if (IMENUCTL_IsActive(pMe->m_pIMainMenu)) {  // if menu is active

            // get id of selected item
            uint16 item = IMENUCTL_GetSel(pMe->m_pIMainMenu);

            // dispatch on selection
            switch (item) {

                  case IDS_MENU_CHOICE1:

                  case IDS_MENU_CHOICE2:

                  case IDS_MENU_CHOICE3:
```

// 7. Free menu object(s) when exiting program; typically in CleanUp()

```
if (pMe->m_pIMainMenu){
      IMENUCTL_Release(pMe->m_pIMainMenu);      // release if allocated
      pMe->m_pImainMenu = NULL;
      }
if (pMe->m_pIMenu2){
      IMENUCTL_Release(pMe->m_pIMenu2);         // release other menus
      pMe->m_pIMenu2 = NULL;
      }
```

```
/*=========================================================================

Sound Sample --

=========================================================================*/

// 1. include file

#include "AEESound.h"               // Sound Interface definitions
#define NUM_TONES 6                 // number of tones in tune
#define TONE_DURATION 250           // duration of tone in milliseconds


// 2. add space for sound object in applet structure

typedef struct _CdemoApp
{
      AEEApplet   a;                      // first element must be AEEApplet
      ISound*     m_pISound;              // pointer to sound object
      AEESoundToneData tone[NUM_TONES];   // list of tones to play
      int16       tone_counter;           // number of tones left to play
} CdemoApp;


// 3. Define callback prototype

// Sound callback function to be registered with ISound interface

static void SoundCBFn(void * pUser, AEESoundCmd eCBType,
                        AEESoundStatus eSPStatus, uint32 dwParam);


// 4. In initialization code, create sound object

if (ISHELL_CreateInstance(pMe->a.m_pIShell, AEECLSID_SOUND,
                          (void **)&pMe->m_pISound) != SUCCESS)
            pMe->m_pISound = NULL;
      else
      {

      // initialize tone list for Happy Birthday

      pMe->tone[0].eTone = AEE_TONE_4;    pMe->tone[0].wDuration =
TONE_DURATION;
      pMe->tone[1].eTone = AEE_TONE_4;    pMe->tone[1].wDuration =
TONE_DURATION;
      pMe->tone[2].eTone = AEE_TONE_2;    pMe->tone[2].wDuration =
TONE_DURATION;
      pMe->tone[3].eTone = AEE_TONE_4;    pMe->tone[3].wDuration =
TONE_DURATION;
      pMe->tone[4].eTone = AEE_TONE_POUND; pMe->tone[4].wDuration =
TONE_DURATION;
      pMe->tone[5].eTone = AEE_TONE_8;    pMe->tone[5].wDuration =
TONE_DURATION*3;
      }
      return TRUE;
}
```

```
// 5. To play a tune

      // register the sound callback function
      // this will be called at the end of each tone
      // inputs are pointer to ISound object, address of callback function,
      //  pointer to app storage structure

      ISOUND_RegisterNotify(pMe->m_pISound,&SoundCBFn,pMe);

      pMe->tone_counter = NUM_TONES;      // set counter to number of tones in
                                          // tune so callback function can
                                          // determine when tune is finished

      // play tune
      // inputs: pointer to ISound object, pointer to an array of tones,
      //         number of tones

      ISOUND_PlayToneList(pMe->m_pISound, pMe->tone, NUM_TONES);  // play tones

}

// 6. To stop a tune

      // cancel sound callback function

      ISOUND_RegisterNotify(pMe->m_pISound,NULL,NULL);

      ISOUND_StopTone(pMe->m_pISound);    // stop sound


// 7. Example of a callback function

/*=========================================================================

FUNCTION SoundCBFn

DESCRIPTION
      This is the callback function that is registered with ISound interface.
    It is called when the tune is finished playing to return to the menu.

PROTOTYPE:
   void SoundCBFn
   (
      AEE_SOUNDCBIndType  eCBType,
      AEESoundStatus eSPStatus,
      const void *   pClientData,
      void *         dwParam
   );

PARAMETERS:
   eCBType: Type of ISound callback.
   eSPStatus: Status contained in callback. The status depends on eCBType.
   pClientData: A reference to correlate this transaction
   dwParam: Additional info specific to callback type and status type
```

```
===========================================================================*/
static void SoundCBFn( void * pUser, AEESoundCmd eCBType,
                       AEESoundStatus eStatus, uint32 dwParam )
{
   CdemoApp* pMe = (CdemoApp*)pUser;

   if ( pMe == 0 )      // return if improperly called
      return;

   if (eStatus == AEE_SOUND_PLAY_DONE)    // Tone finished

       // Since Play Done is called after each tone, decrement counter
       // if zero, tune is completed
   {
         if (!--pMe->tone_counter)
         {
            // cancel sound callback function

            ISOUND_RegisterNotify(pMe->m_pISound,NULL,NULL);

            // *** Insert your processing here ***

            ISHELL_LoadResString(pMe->a.m_pIShell, FJRDEMO_RES_FILE, IDS_DONE,
                                 szText, STR_BUF_SIZE);

            IDISPLAY_ClearScreen(pMe->a.m_pIDisplay);  // clear screen

            IDISPLAY_DrawText(pMe->a.m_pIDisplay,AEE_FONT_BOLD,szText,
                              -1,0,0,NULL, IDF_ALIGN_CENTER | IDF_ALIGN_MIDDLE);

            IDISPLAY_Update (pMe->a.m_pIDisplay); // display message

         }
   }
}

// 8. Release sound object when exiting


      if (pMe->m_pISound){                    // if object is allocated
            ISOUND_Release(pMe->m_pISound);  // then release it
            pMe->m_pISound = NULL;
            }
```

```
/*============================================================================

Graphics sample

============================================================================*/

// 1. Include file

#include "AEEGraphics.h"      // Graphics Interface definations

// 2. Reserve space for graphics object

typedef struct _CdemoApp
{
      AEEApplet   a;                // first element must be AEEApplet
      IGraphics*  m_pIGraphics;     // point to graphics object
} CdemoApp;


// 3. In initialization code, create graphics object

      // create graphics object
      // inputs are pointer to shell object, class id and
      //        address to receive pointer to graphics object
      // if successful, object is created and pointer returned

      if (ISHELL_CreateInstance(pMe->a.m_pIShell, AEECLSID_GRAPHICS,
                                (void **)&pMe->m_pIGraphics) != SUCCESS)
            pMe->m_pIGraphics = NULL;



// 4. To use graphics object

      AEERect     rect;             // storage for structures to define shapes
      AEECircle   cir;
      AEETriangle tri;


      IDISPLAY_ClearScreen(pMe->a.m_pIDisplay); // clear screen using
                                                // IDisplay function

      IGRAPHICS_Setxxx(pMe->m_pIGraphics,...);  // set non-default values

      // use of Translate function to offset graphic (which provides
      // some white space from upper left corner of the screen)
      // inputs are pointer to IGraphics object, x offset and y offset
      // offsets are computed based on the previously-stored screen size

      IGRAPHICS_Translate(pMe->m_pIGraphics,
                          (int16) -(pMe->m_ScreenRect.dx/10),
                          (int16) -(pMe->m_ScreenRect.dy/10));
```

```
      rect.x = 0;             // define rectangle in world coordinates
      rect.y = 0;
      rect.dx = 40;
      rect.dy = 30;

      IGRAPHICS_DrawRect(pMe->m_pIGraphics, &rect);   // draw rectangle

      cir.cx = ;              // define circle
      cir.cy = ;
      cir.r = ;

      IGRAPHICS_DrawCircle(pMe->m_pIGraphics, &cir);  // draw circle

      tri.x0 = ;              // define triangle
      tri.y0 = ;
      tri.x1 = ;
      tri.y1 = ;
      tri.x2 = ;
      tri.y2 = ;

      IGRAPHICS_DrawTriangle(pMe->m_pIGraphics, &tri);// draw triangle

      IGRAPHICS_Update (pMe->m_pIGraphics);     // display result using
                                                // IGraphics function


// 5. Free graphics object when done

      if (pMe->m_pIGraphics){
            IGRAPHICS_Release(pMe->m_pIGraphics);
            pMe->m_pIGraphics = NULL;
            }
```

```
// Basic BREW debug logfile writer (pseudocode).

// 1. include file

#include AEEFILE.H
#define APP_LOG_FILE "app_log.txt" // Filename to receive log data

// 2. add space for file object(s) in applet structure

typedef struct _CdemoApp {
      AEEApplet   a;
      IFileMgr*   pFileMgr;         // pointer to file manager object
} CdemoApp;

// 3. In application initialization code, create file manager object

      if ( ISHELL_CreateInstance(pMe->ishell, AEECLSID_FILEMGR,
            (void **)&pMe->pFileMgr) != SUCCESS ) {
            return ENOMEMORY;
      }

// 4. WriteLog function
// To avoid losing log files prior in event of program crash:
// open file, write data and close file.

void WriteLog(CdemoApp *pMe, const char* szLogMessage)  {
      IFile*  pFile = NULL;   // note: IFile pointer is local

      // check pointers are valid
      if (NULL == pMe || NULL == pMe->pFileMgr)
            return;

      // Open the log file for append
      pFile = IFILEMGR_OpenFile(pMe->pFileMgr, APP_LOG_FILE, _OFM_APPEND);

      if (NULL == pFile) { // if file doesn't exist, create it
           pFile = IFILEMGR_OpenFile(pMe->PFileMgr, APP_LOG_FILE, _OFM_CREATE);
      }

      // Write to the log file
      if (pFile) {
            IFILE_Write(pFile, szLogMessage, STRLEN(szLogMessage));

            // Close the file, flushing in-memory buffers.
            IFILE_Release(pFile);
      }
}

// 5.  In application cleanup code, release file manager object

      if (pMe->pFileMgr) {
            IFILEMGR_Release(pMe->pFileMgr);
            pMe->pFileMgr = NULL;
      }
```

```
/*=========================================================================

IWeb Sample --

=========================================================================*/

// 1. include file

#include "AEEWeb.h"                // Web Interface definitions


// 2. reserve space for web object(s), responses and callback functions in
//    applet structure

typedef struct _MyApp
{
      AEEApplet     a;                  // first element must be AEEApplet
      IWeb*         pIWeb;              // pointer to web object
      IWebResponse* pIWebResponse       // pointer to response
      WebRespInfo*  pWebRespInfo;       // pointer to info about web response
      ISource*      pISource;           // pointer to response stream object
      AEECallback   WebCBStruct;        // structure for Iweb callback function
} MyApp;


// 3. In initialization code, create iweb object

if (ISHELL_CreateInstance(pMe->a.m_pIShell, AEECLSID_WEB,
                          (void **)&pMe->pIWeb) != SUCCESS){
            pMe->pIWeb = NULL;
            return FALSE;
            }
            else {

            // 3a. Set up the callback function to receive response from server

            CALLBACK_INIT(&pMe->WebCBStruct, IWebCB, pMe);  // out, in, in
            Return TRUE;
            }


// 4.  To issue an HTTP request to a URL, use IWEB_GETResponse macro
//     Note use of inter parentheses and two pIWeb pointers!
//     URL must be fully specified; could be within a C-string variable

IWEB_GETResponse(pMe->pIWeb,
                  (pMe->pIWeb, &pMe->IWebResponse,
                   &pMe->WebCBStruct,
                   “http://www.someplace.com/somepage.html”,
                   WEBOPT_END));

return; // response processed in callback
```

```
// 5.  Response from the webserver is received by the callback function

void IWebCB (MyApp* pMe) {

      // get info about the response
      pMe->pWebRespInfo = IWEBRESP_GetInfo(pMe->pWebResponse);

      // 5.a  check error code
      if (!WEB_ERROR_SUCCEEDED(pMe->pWebRespInfo.nCode)) {
            DisplayError(…);
            return;
            }

      // 5.b  get pointer to Source object
      pMe->pISource = pMe->pWebRespInfo->pisMessage);
      if (pMe->pISource == NULL) {
            ProcessError(…);
            return;
            }

      // 5.c  register Isource Read callback
      CALLBACK_INIT(&pMe->WebCBStruct, ReadFromWebCB, pMe);  // out, in, in

      // 5.d  post a read; data is processed by ISource callback
      ISOURCE_Readable(pMe->pISource, &pMe->WebCBStruct);
      return;
      }


// 6.  Data from webserver is read from ISource stream
//     Data arrives in chunks and is processed chunk at a time.

void ReadFromWebCB(MyApp* pMe) {

      char buf[1024];         // allocate buffer
      int byteCount;

      // read data from stream; get number of bytes read
      byteCount = ISOURCE_Read(pMe->pISource, (char*)buf, sizeof(buf));

      switch (byteCount) {
            case ISOURCE_WAIT:      // Buffer empty, but more data expected
                                    // post another read
                  ISOURCE_Readable(pMe->pISource,&pMe->WebCBStruct);
                  return;

            case ISOURCE_ERROR:     // Error occurred
                  ProcessError(…);
                  return;

            case ISOURCE_END:       // Buffer empty; all data received
                  ProcessData(…);
                  return;

            default:                // data read; copy from chunk buffer
                  MEMCPY(finalDestination, buf, byteCount);
                  finalDestination =+ byteCount;
```

```
                // post another read
                ISOURCE_Readable(pMe->pISource,&pMe->WebCBStruct);
                return;
        }
    }


// 7. NOTE: No Web related events in event handler


// 8. Free object(s) when exiting program; typically in CleanUp()

    // cancel callback
    CALLBACK_Cancel(&pMe->WebCBStruc);

    if (pMe->pIWebResp)
    {
        IWEBRESP_Release(pMe->pIWebResp);
        pMe->pIWebResp = NULL;
    }

    if (pMe->pIWeb)
    {
        IWEB_Release(pMe->pIWeb);
        pMe->pIWeb = NULL;
    }

// 9. Don't forget to free any dynamically allocated buffers, for example:

    FREE(finalDestination);
```

```
/*=========================================================================

         Copyright © 2000-2002 QUALCOMM Incorporated.
                    All Rights Reserved.
                  QUALCOMM Proprietary/GTDR
=========================================================================*/

// Demonstrates use of File, Image Control, GPS and Web APIs
// Uses callback functions for GPS and Web

#include "AEEAppGen.h"
#include "helloworld.bid"
#include "AEEFile.h"
#include "AEEImageCtl.h"
#include "AEEPosDet.h"
#include "AEEWeb.h"
#include "AEEStdLib.h"

#define FORALL(p,a)        for (p = (a) + ARRAY_SIZE(a) - 1; p >= (a); --p)
#define FOR_ALL_WEBACTIONS(pApp, var, exp)     {WebAction *var; FORALL(var,
(pApp)->m_awa) { exp; } }

typedef struct _CHelloWorldApp CHelloWorldApp;

typedef struct WebAction {
   CHelloWorldApp *     pParent;
   AEECallback          cb;
   IWebResp *           piWResp;
   IGetLine *           piGetLine;
   int                  nLines;
   int                  nBytes;
   uint32               uStart;
   uint32               uRecvStart;
   IPeek *              pipPostData;
   char *               pszPostData;
} WebAction;

struct _ChelloWorldApp {
   AEEApplet            a;
   IFileMgr *           m_pFileMgr;
   IWeb *               m_pIWeb;
   IPosDet *            m_pIPosDet;    // pointer to Position object
   ISource *            m_pISource;
   IImageCtl *          m_pIImage;
   AEEGPSInfo *         m_pGPSinfo;    // pointer to returned GPS data struct
   AEECallback          m_cbgps;
   AEECallback          m_cbimg;
   WebAction            m_awa[1];
};

char serviceURL[100];
IFile * file;

// declare function prototype

static boolean HelloWorld_HandleEvent(CHelloWorldApp * pme, AEEEvent
eCode,uint16 wParam, uint32 dwParam);
```

```
// Initialize Applet Data

static boolean InitAppData(CHelloWorldApp * pApp) {
   IShell * pIShell = pApp->a.m_pIShell;
   int code = 0;

   // create all interfaces
   if( (ISHELL_CreateInstance(pIShell, AEECLSID_FILEMGR,
            (void**)(&pApp->m_pFileMgr)) != SUCCESS) ||
      (ISHELL_CreateInstance(pIShell, AEECLSID_WEB,
            (void **)(&pApp->m_pIWeb)) != SUCCESS) ||
        (ISHELL_CreateInstance(pIShell, AEECLSID_POSDET,
            (void **)(&pApp->m_pIPosDet)) != SUCCESS) ||
        (ISHELL_CreateInstance(pIShell, AEECLSID_IMAGECTL,
            (void**)(&pApp->m_pIImage)) != SUCCESS)) {
      IAPPLET_Release((IApplet*)pApp);
      return FALSE;
   }

   {
      // initialize Web options structure
      int    i = 0;
      WebOpt awo[10];
      awo[i].nId  = WEBOPT_CONNECTTIMEOUT;
      awo[i].pVal = (void *)10000;
      i++;
      awo[i].nId  = WEBOPT_FLAGS;
      awo[i].pVal = (void *)WEBREQUEST_NOWAITCONN;
      i++;
      awo[i].nId  = WEBOPT_PROXYSPEC;
      awo[i].pVal = (void *)"http:///";
      i++;
      awo[i].nId  = WEBOPT_PROXYSPEC;
      awo[i].pVal = (void *)"*:///http://webproxy.yourdomain.com:8080";
      i++;
      awo[i].nId  = WEBOPT_END;
      IWEB_AddOpt(pApp->m_pIWeb,awo);
   }
   FOR_ALL_WEBACTIONS(pApp, p, p->pParent = pApp);
   return TRUE;
}

// Create Instance of Applet -- Called when Applet is launched

int AEEClsCreateInstance(AEECLSID ClsId,IShell * pIShell,IModule * pMod,void **
ppObj)
{
   CHelloWorldApp * pApp = NULL;
   *ppObj = NULL;

   if(!AEEApplet_New( sizeof(CHelloWorldApp),
                     ClsId,
                     pIShell,
                     pMod,
                     (IApplet**)ppObj,
                     (AEEHANDLER)HelloWorld_HandleEvent,
                     NULL))
      return(ENOMEMORY);

      pApp = (CHelloWorldApp *)*ppObj;
```

```
      if (!InitAppData(pApp)) {          // initialize applet data (above)
            *ppObj = NULL;
            return EFAILED;
      }

      return(AEE_SUCCESS);
}

// Callback function for HTTP read - enter when data is available from server

static void imgCallback(void * p) {

      int bytesRead;
      char x[100];               // byte buffer
      IImage* pSplash = NULL;
      AEEImageInfo rImageInfo;
      CtlAddItem ci;
      AEERect rc;
      CHelloWorldApp * pApp = (CHelloWorldApp *)p;

      // read bytes from HTTP packet (source)
      bytesRead = ISOURCE_Read(pApp->m_pISource, x, 100);

      while (bytesRead != ISOURCE_END && bytesRead != ISOURCE_ERROR) {

            if (bytesRead != ISOURCE_WAIT)
                  IFILE_Write(file, x, bytesRead);   // (2) Write buffer x
            else {
                  ISOURCE_Readable(pApp->m_pISource, &pApp->m_cbimg);
                  return;
            }
            bytesRead = ISOURCE_Read(pApp->m_pISource, x, 100);

      }
      IFILE_Release(file);   // (3) No more data - Close file

      If (bytesRead != ISOURCE_ERROR) {   // if success, (4) load image from
                                                  file just created

        if((pSplash = ISHELL_LoadImage(pApp->a.m_pIShell,"map.png")) != NULL){
                  // if image loaded
                  IIMAGE_GetInfo(pSplash, &rImageInfo);
                  IDISPLAY_ClearScreen(pApp->a.m_pIDisplay);

                  MEMSET(&ci,0,sizeof(ci));
                  ci.pImage = pSplash;
                  rc.x = 0;
                  rc.y = 0;
                  rc.dx = 120;
                  rc.dy = 140;

                  // Note use of IIMAGECTL to create scrollable image

                  IIMAGECTL_SetProperties(pApp->m_pIImage,CP_BORDER);
                  IIMAGECTL_SetRect(pApp->m_pIImage, &rc);
                  IIMAGECTL_SetImage(pApp->m_pIImage,pSplash);
                  IIMAGECTL_SetActive(pApp->m_pIImage,TRUE);
                  IIMAGECTL_Redraw(pApp->m_pIImage);
```

```
                    IIMAGE_Release(pSplash);
                    IDISPLAY_Update(pApp->a.m_pIDisplay);

                    // Note: event handler performs image scrolling

                    return;
                }
        }
}

// Routine to process responses from Web Server

static void WebAction_GotResp(void *p)
{
   WebAction *pwa = (WebAction *)p;
   CHelloWorldApp * pApp = pwa->pParent;
   WebRespInfo *pwri;

   pwri = IWEBRESP_GetInfo(pwa->piWResp);
   pApp->m_pISource = pwri->pisMessage;

   // (1) Delete old file and create new one
   IFILEMGR_Remove(pApp->m_pFileMgr, "map.png");
   file = IFILEMGR_OpenFile(pApp->m_pFileMgr, "map.png", _OFM_CREATE);

   // ** should test for error !  (file == NULL)

   // setup callback to read data
   CALLBACK_Init(&pApp->m_cbimg,imgCallback,pApp);
   ISOURCE_Readable(pApp->m_pISource, &pApp->m_cbimg);
}


// Function to send URL to web server

static void WebAction_Start(WebAction *pwa, char *pszUrl) {
      CHelloWorldApp * pApp = pwa->pParent;

      // setup call back to read response from web server
      CALLBACK_Init(&pwa->cb, WebAction_GotResp, pwa);

      IWEB_GetResponse(pApp->m_pIWeb,
                           (pApp->m_pIWeb, &pwa->piWResp, &pwa->cb, pszUrl,
                              WEBOPT_HANDLERDATA, pwa,
                              WEBOPT_HEADER, "X-Method: GET\r\n",
                              WEBOPT_END));
}

// GPS Callback function - entered when GPS determination completes
//     GPSinfo contains lat/lon

static void gpsCallback (void * p) {

      CHelloWorldApp * pApp = (CHelloWorldApp *)p;

      // build URL by extracting lat/lon from GPSinfo
      sprintf(serviceURL,
            "http://localhost/BREWServer/WebForm1.aspx?latlon=%u^%u",
             pApp->m_pGPSinfo->dwLat, pApp->m_pGPSinfo->dwLon);
```

```
      // send URL to server
      FOR_ALL_WEBACTIONS(pApp, p, WebAction_Start(p, serviceURL) );
}

// Applet Event Handler

static boolean HelloWorld_HandleEvent(CHelloWorldApp * pMe, AEEEvent eCode,
uint16 wParam, uint32 dwParam)
{
   int code = 0;
   AECHAR szText[24];
   STR_TO_WSTR("Please Wait", szText, 24);

   // pass all events first to Image control event handler
   if (IIMAGECTL_HandleEvent(pMe->m_pIImage, eCode, wParam, dwParam))
      return TRUE;  // if handled, nothing left to do, exit

   switch (eCode){
      case EVT_APP_START:

         CALLBACK_Init(&pMe->m_cbgps,gpsCallback,pMe);     // Set GPS callback
         pMe->m_pGPSinfo = MALLOC(sizeof(AEEGPSInfo));     // allocate space
                                                           // for returned data

         //make call to determine location
         code = IPOSDET_GetGPSInfo(pMe->m_pIPosDet,
                              AEEGPS_GETINFO_LOCATION,
                              AEEGPS_ACCURACY_HIGHEST,
                              pMe->m_pGPSinfo,
                              &pMe->m_cbgps);

         // display “please wait” message
         IDISPLAY_DrawText(pMe->a.m_pIDisplay,
                           AEE_FONT_BOLD,
                           szText,
                           -1,
                           0,
                           0,
                           NULL,
                           IDF_ALIGN_CENTER | IDF_ALIGN_MIDDLE);
         IDISPLAY_Update (pMe->a.m_pIDisplay);
         return(TRUE);

      case EVT_APP_STOP:
         return(TRUE);

        // This is a better way to pass key events to image control
        case EVT_KEY:
        case EVT_KEY_HELD:
        case EVT_KEY_PRESS:
        case EVT_KEY_RELEASE:
              return (IIMAGECTL_HandleEvent(pMe->m_pIImage, eCode, wParam,
                      dwParam));

      default:
         break;
   }
   return(FALSE);
}
```