Writing Dynamic Link Libraries
for
The Windows ASPECT Script Language

C Language Interface Specification

Introduction

PROCOMM PLUS for Windows implements a powerful scripting language called the Windows ASPECT Script Language (ASPECT). One of ASPECT's great strengths is the ability to call functions embedded in Dynamic Link Libraries (or DLLs) written and compiled outside of ASPECT. This paper focuses on the writing of DLLs for ASPECT in the C language.

ASPECT to DLL Parameter Sequence

ASPECT generates a standard parameter block for every DLL call that it makes. This standard parameter block has five parameters, listed in order below. When writing a DLL to be called from ASPECT, every exported function that is to be accessible from within ASPECT should declare exactly these number and types of parameters:

1. [HWND] - a handle to the PROCOMM PLUS for Windows main window.
2. [HANDLE] - a handle to the instance of PROCOMM PLUS for Windows that made the DLL call.
3. [void far * far *] - a far array of far pointers to the data elements listed as parameters on the ASPECT script line that called the DLL function.
4. [LPBYTE] - a far array of bytes. Byte N of this array describes the type of element N stored in the array of data elements (i.e. parameter 3). The range of values in this array follows:

Byte Value	Meaning
0	n^{th} element is an ASPECT string
1	n^{th} element is an ASPECT integer
2	n^{th} element is an ASPECT long
3	n^{th} element is an ASPECT float

5. [int] - an integer containing the count of parameters provided on the ASPECT script line calling the DLL function.

An Example

Below is the code for a DLL with a function that generates a random number. Under a PASCAL style, stack-based parameter passing scheme, a variable number of parameters is not

normally possible; however, ASPECT provides an array-based parameter passing scheme as described by the parameter block above. To demonstrate the ability to pass a variable number of parameters, the DLL function listed below makes certain assumptions about the provided parameters based on the number of parameters that are available:

1. If two parameters are passed to the DLL function:
   a. the first parameter holds a number for seeding the random-number generator.
   c. the second parameter will hold the return value (i.e. the random number).

2. If one parameter is passed to the DLL function:
   a. the DLL function should seed the random number generator using the GetCurrentTime() function (from the Windows API).
   c. the single parameter will hold the return value (i.e. the random number).

Here is the code for the DLL. This is RANDINT.C:

```
#include <windows.h>
#include <stdlib.h>

#define IntegerType 1

int FAR PASCAL LibMain( HANDLE, WORD, WORD, LPSTR );
int FAR PASCAL RandomInt( HWND, HANDLE, void far * far *, LPBYTE, int );
int FAR PASCAL WEP( int );

HANDLE hInst;

int FAR PASCAL LibMain( HANDLE hInstance, WORD wDataSeg, WORD cbHeapSize,
  LPSTR lpszCmdLine )
{
  if (cbHeapSize)
    UnlockData(0);    /* make the data segment moveable */
  hInst = hInstance;
  return(TRUE);
}

int FAR PASCAL RandomInt( HWND PWWnd, HANDLE PWInst,
  void far * far *vdatptr, LPBYTE vtypeptr, int argcnt )
{
  if ( argcnt < 1 )                 /* at least one argument? */
    return( FALSE );                  /* if not, return FAILURE */
  if ( vtypeptr[0] != IntegerType )     /* is first arg an integer? */
    return( FALSE );                  /* if not, return FAILURE */
  if ( argcnt > 1 ) {                /* more arguments? */
    if ( vtypeptr[1] != IntegerType )   /* is second arg an integer? */
```

```
    return( FALSE );                /* if not, return FAILURE */
   srand( *(int far *)vdatptr[0] );    /* seed the generator with arg */
   *(int far *)vdatptr[1] = rand();    /* get a random number */
  }
  else {
   srand( (unsigned)GetCurrentTime() );/* seed the generator with time */
   *(int far *)vdatptr[0] = rand();    /* get a random number */
  }
  return( TRUE );                 /* return SUCCESS to ASPECT */
}

int FAR PASCAL WEP( int bSystemExit )
{
  return(TRUE);
}
```

Here is the project definition file. This is RANDINT.DEF:

```
LIBRARY RandInt
CODE MOVEABLE DISCARDABLE
DATA SINGLE MOVEABLE
HEAPSIZE 2048
EXPORTS
  WEP       @1
  RandomInt  @2
```

Here is an ASPECT script using the DLL. This is RANDINT.WAS:

```
;ASPECT script demonstrating RANDINT.DLL
integer hdll, rNum
proc cleanup
  dllfree hdll
  exit
endproc

proc main
string DLLPATH = "C:\PROWIN\ASPECT\RANDINT.DLL" ;path to the DLL

  when userexit call cleanup            ;point to cleanup routine
  dllload DLLPATH hdll                   ;load the DLL
  if (failure)
    usermsg "dllload failed"
    exit
  endif

  ;generate a random number letting the DLL use system time as seed
```

```
  dllcall hdll "RandomInt" rNum
  if (failure)
    usermsg "RandomInt call with system time seed failed"
    cleanup()
  else
    usermsg "RandomInt with system time seed returned %d" rNum
  endif

  ;generate a random number with the seed 65
  dllcall hdll "RandomInt" 65 rNum
  if (failure)
    usermsg "RandomInt call with seed 65 failed"
    cleanup()
  else
    usermsg "RandomInt with seed 65 returned %d" rNum
  endif

  cleanup()
endproc
```