Subject: Keyword DLM crashes with IDL 5.2?
Posted by Randall Skelton on Mon, 30 Apr 2001 16:29:21 GMT
View Forum Message <> Reply to Message

Hi all,

I am trying to port a DLM that works in IDL 5.4/5.3 to work with IDL 5.2.
As a demonstration of my problem I have modified the file testmodule.c to
include a simple procedure called 'testr' that takes a keyword and
distributed it below. It compiles under unix with:

gcc -Wall -Wimplicit -fpic -shared -I/usr/local/PACK/idl-5.3/external
testmodule.c -o testmodule.so

Execution in IDL is:

```
IDL> testr, 1, a, ERROR=b
% Loaded DLM: TESTMODULE.
IDL> print, a, b
        10          0
```

This routine compiles and works just fine under IDL 5.3, but it seg-faults
during execution with IDL 5.2. Without the keywords, the procedure works
fine, and I can trace the fault to calling 'IDL_KWGetParams' in IDL 5.2.
I've read the relevant sections of the EDG at least twice over and after
an afternoon of fiddling, I cannot find the bug. Any and all help is
greatly appreciated!

Randall

-- testmodule.dlm --

```
MODULE testmodule
DESCRIPTION Test code for loadable modules
VERSION 1.0
SOURCE Testing
BUILD_DATE APRIL 2001
FUNCTION        TESTFUN 0       IDL_MAXPARAMS
PROCEDURE       TESTPRO 0       IDL_MAX_ARRAY_DIM
PROCEDURE       TESTR   2       2  KEYWORDS
```

-- testmodule.c --

```
#include <stdio.h>
#include "export.h"

/* Handy macro */
#define ARRLEN(arr) (sizeof(arr)/sizeof(arr[0]))

/*
```

```
 * Define message codes and their corresponding printf(3) format
 * strings. Note that message codes start at zero and each one is
 * one less that the previous one. Codes must be monotonic and
 * contiguous.
 */
static IDL_MSG_DEF msg_arr[] =
{
#define M_TM_INPRO                      0
  {  "M_TM_INPRO",   "%NThis is from a loadable module procedure." },
#define M_TM_INFUN                      -1
  {  "M_TM_INFUN",   "%NThis is from a loadable module function." },
};

static IDL_VPTR ierr;
static IDL_VPTR idbg;
static IDL_KW_PAR err_pars[] = {IDL_KW_FAST_SCAN,
  {"DEBUG", IDL_TYP_LONG, 1, IDL_KW_OUT|IDL_KW_ZERO, NULL,
IDL_CHARA(idbg)},
  {"ERROR", IDL_TYP_UNDEF, 1, IDL_KW_OUT|IDL_KW_ZERO, NULL,
IDL_CHARA(ierr)},
  {NULL}
};

/*
 * The load function fills in this message block handle with the
 * opaque handle to the message block used for this module. The other
 * routines can then use it to throw errors from this block.
 */
static IDL_MSG_BLOCK msg_block;


/* Implementation of the TESTPRO IDL procedure */

static void testpro(int argc, IDL_VPTR *argv)
{
  IDL_MessageFromBlock(msg_block, M_TM_INPRO, IDL_MSG_RET);
}

static void IDL_CDECL testr(int argc, IDL_VPTR argv[], char *argk) {

    /* Local */
    IDL_LONG j;
    IDL_VPTR tmp = IDL_Gettmp();
    IDL_ULONG typ;

    /* Keywords */
    IDL_ALLTYPES error;
    error.l = 0;
```

```
    IDL_KWCleanup(IDL_KW_MARK);

    /* FAULT ON NEXT LINE */
    IDL_KWGetParams(argc, argv, argk, err_pars, NULL, 1);

     /* Get the result index from IDL */
     IDL_ENSURE_SCALAR(argv[0]);
     j = IDL_LongScalar(argv[0]);

     typ = 10;
     IDL_StoreScalarZero(argv[1], IDL_TYP_ULONG);
     tmp->type = IDL_TYP_ULONG;
     tmp->value.ul = typ;
     IDL_VarCopy(tmp, argv[1]);

     if (ierr) {IDL_StoreScalar(ierr, IDL_TYP_LONG, (IDL_ALLTYPES *) &error);}

     IDL_KWCleanup(IDL_KW_CLEAN);
}


/* Implementation of the TESTFUN IDL function */
static IDL_VPTR testfun(int argc, IDL_VPTR *argv)
{
  IDL_MessageFromBlock(msg_block, M_TM_INFUN, IDL_MSG_RET);
  return IDL_StrToSTRING("TESTFUN");
}

int IDL_Load(void)
{
  /*
   * These tables contain information on the functions and procedures
   * that make up the TESTMODULE DLM. The information contained in these
   * tables must be identical to that contained in testmodule.dlm.
   */
  static IDL_SYSFUN_DEF function_addr[] = {
    { testfun, "TESTFUN", 0, IDL_MAXPARAMS, 0},
  };

  static IDL_SYSFUN_DEF procedure_addr[] = {
    { (IDL_FUN_RET) testpro, "TESTPRO", 0, IDL_MAX_ARRAY_DIM, 0},
    { (IDL_FUN_RET) testr,   "TESTR",   2, 2,  IDL_SYSFUN_DEF_F_KEYWORDS},
  };


  /*
   * Create a message block to hold our messages. Save its handle where
```

```
   * the other routines can access it.
   */
  if (!(msg_block = IDL_MessageDefineBlock("Testmodule", ARRLEN(msg_arr),
                                           msg_arr)))
    return IDL_FALSE;

  /*
   * Register our routine. The routines must be specified exactly the same
   * as in testmodule.dlm.
   */
  return IDL_AddSystemRoutine(function_addr, TRUE, ARRLEN(function_addr))
    && IDL_AddSystemRoutine(procedure_addr, FALSE,
ARRLEN(procedure_addr));
}
```