

---

Subject: Passing string array to a DLM

Posted by [Randall Skelton](#) on Fri, 29 Nov 2002 18:25:23 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Hi all,

I've managed to stymie myself with a DLM yet again. In this case, I want to pass a single dimension string array from IDL into C and return a corresponding array of indicies. Accordingly from C, I want to be able to index through this string array calling a C function that processes the strings in a for-loop. Below is a sample that should segfault for IDL 5.x. My problem is basically that I cannot consistently index through the strings that are sitting in the `argv[0]->value.arr->data` pointer. I assume that this pointer points to the first IDL string structure so it is cast as `(IDL_STRING *)`. I have tried various ways of sub-scripting and augmenting the `'strings->s'` pointer and can get the next string in a hit-and-miss way that leaves me to believe I am truly missing something.

Call in IDL as:

```
a = test_passstrings(['blah', 'blah', 'blacksheep'])
```

Any ideas?

Thanks and happy thanksgiving :)

Randall

```
--- START: test.dlm ---  
MODULE TEST  
DESCRIPTION Testing passing a string array  
VERSION 1.0  
SOURCE Me  
BUILD_DATE November 2002  
FUNCTION TEST_PASSSTRINGS 1 1  
--- END: test.dlm ---
```

```
--- START: test.c ---
```

```
/* ANSI */  
#include <stdio.h>  
#include <stdlib.h>  
#include <math.h>  
#include <string.h>
```

```
/* IDL */  
#include "export.h"
```

```

/* Message Numbers */
#define TEST_ERROR 0

/* Useful macro */
#define ARRLen(arr) (sizeof(arr)/sizeof(arr[0]))

extern IDL_MSG_BLOCK msg_block;
extern void TEST_exit_handler(void);
extern int TEST_Startup(void);
extern IDL_VPTR IDL_CDECL
TEST_PassStrings(int argc, IDL_VPTR argv[], char *argk);

/* Globals here */
static char statusBuffer[256]; /* for IDL message passing */

static IDL_MSG_DEF msg_arr[] = {
    { "TEST_Error", "%NError: %s." },
    { "TEST_NOSTRINGARRAY", "%Nstring arrays not allowed %s"},
};

IDL_MSG_BLOCK msg_block;

int IDL_Load(void) {
    /* Define the message block */
    if (!(msg_block = IDL_MessageDefineBlock("TEST", ARRLen(msg_arr),
        msg_arr))) {
        return IDL_FALSE;
    }

    /* Call the startup function to add the routines to IDL. */

    /* Routines */
    if (!TEST_Startup()) {
        IDL_MessageFromBlock(msg_block, TEST_ERROR,
            IDL_MSG_RET, "Unable to initialize TEST interface");
    }
    return IDL_TRUE;
}

#ifdef __IDLPRE53__
/* FOR IDL < 5.3 */
/* Define the functions */
static IDL_SYSFUN_DEF TEST_functions[] = {
    {TEST_PassStrings, "TEST_PASSSTRINGS", 1, 1, 0},
};
#else
/* FOR IDL >= 5.3 */

```

```

/* Define the functions */
static IDL_SYSFUN_DEF2 TEST_functions[] = {
    {TEST_PassStrings, "TEST_PASSSTRINGS", 1, 1, 0, 0},
};
#endif

/* Startup call when DLM is loaded */
int TEST_Startup(void)
{

/* If IDL is pre-5.3 then change IDL_SysRtnAdd to IDL_AddSystemRoutine,
 * (NB: the parameters stay the same) */

#ifdef __IDLPRE53__
/* FOR IDL < 5.3 */
/* Add functions */
if (!IDL_AddSystemRoutine(TEST_functions, TRUE,
    ARRLEN(TEST_functions))) {
    return IDL_FALSE;
}
#else
/* FOR IDL >= 5.3 */
/* Add functions */
if (!IDL_SysRtnAdd(TEST_functions, TRUE,
    ARRLEN(TEST_functions))) {
    return IDL_FALSE;
}
#endif

/* Register the error handler */
IDL_ExitRegister(TEST_exit_handler);
return(IDL_TRUE);
}

/* Called when IDL is shutdown */
void TEST_exit_handler(void) {
    /* Nothing special to do in this case */
}

/*-----*/
/* Function: TEST_PassStrings
 *
 * Purpose: Returns an index associated with the given name.
 *
 * Called in IDL as:
 *   index = TEST_PassString('string')

```

```

*
* Where:
*   string = the field name (IDL_STRING)
*   index  = returned index number
*
*/
IDL_VPTR IDL_CDECL TEST_PassStrings(int argc, IDL_VPTR argv[], char *argk) {

/* Local */
    int i;
    IDL_STRING *strings;
    IDL_VPTR tmp;
    IDL_LONG *lonarr;
    IDL_LONG IDim[IDL_MAX_ARRAY_DIM];
    long n = 0;

/* Get the field index from IDL */
    IDL_ENSURE_SIMPLE(argv[0]);

/* Determine if passed argv is an array */
if (argv[0]->flags & IDL_V_ARR) {
/* ARGV IS AN ARRAY */

/* Ensure we only pass one dimension */
if (argv[0]->value.arr->n_dim > 1) {
    sprintf(statusBuffer, "Index cannot exceed one dimension");
    IDL_Message(IDL_M_NAMED_GENERIC, IDL_MSG_LONGJMP, statusBuffer);
}

/* Get the number of elements */
n = argv[0]->value.arr->n_elts;

/* Create the output string array */
IDim[0] = n;

lonarr = (IDL_LONG *) IDL_MakeTempArray((int) IDL_TYP_LONG, 1, IDim,
    IDL_ARR_INI_NOP, &tmp);

strings = (IDL_STRING *) argv[0]->value.arr->data;

/* ----- PROBLEM IS BELOW ----- */
for (i=0; i<n; i++) printf("Strings[%i]: %s\n", i,
    (char *) strings->s + i * sizeof(IDL_STRING));

/* for (i=0; i<n; i++) printf("Strings[%i]: %s\n", i,
    (char *) strings->s[i]); */
/* ----- PROBLEM IS ABOVE ----- */

```

```
/* Fill the array */
for (i=0; i<n; i++) lonarr[i] = i;

} else {

/* Allocate a scalar long to return */
tmp = IDL_GettmpLong(1);

}

return (tmp);
}

--- END: test.c ---
```

---