SCSI TOOLBOX, LLC

Example DMM External Program

Contents

Sales: 720.249.2641
General: 303.972.2072

Issuing a SATA SMART Self-Test via DMM External Program

This article will describe a sample C++ (Visual Studio 6) project which uses calls to the Developers Toolbox (DTB) to issue a SATA SMART Self-Test command to drives. The program is designed to be called from the DMM External Program test step.

DMM will pass the address of each drive under test to this program via command-line parameters. The program parses the command line, extracts the host adapter, target, and LUN of the drive, issues an ATA command to the drive, waits for the self-test to complete, then sends back a text message which DMM will enter into that drives log file.

This project is meant to be used as a template, and illustrates:

-how to parse the command line parameters to extract the drive address

-how to issue a command to DTB

-how to send information back to DMM to be logged

The project/program is called "DMMSataSelfTest" and can be downloaded using this link – http://scsitoolbox.com/ScryptCrypt/files/DMM/DMMSataSelfTest-90410.zip

It is an MFC-enabled console application, written in C++ using Visual Studio version 6.

The project is set up to link to the DTB library (vcpssl.lib) and to include the DTB header file (VCPSSLImports.h)

Sales: 720.249.2641
General: 303.972.2072

The key points of the program are shown here:

Parsing the command-line arguments

```
for (loop = 0; loop < argc; loop++) // get the command line arguements for ha, target, & lun
{
        sprintf(Args,"%s\n",argv[loop]);
        if ( strstr(Args,"HBA") != NULL) {
                p = Args;
                p = strtok( Args, "," );
                p+=4;
                sscanf(p,"%d",&ha);
                p = strtok(NULL,",");
                p+=4;
                sscanf(p,"%d",&tid);
                p = strtok(NULL,",");
                p+=4;
                sscanf(p,"%d",&lun);
                break;
        }
} // end of parsing command line args
```

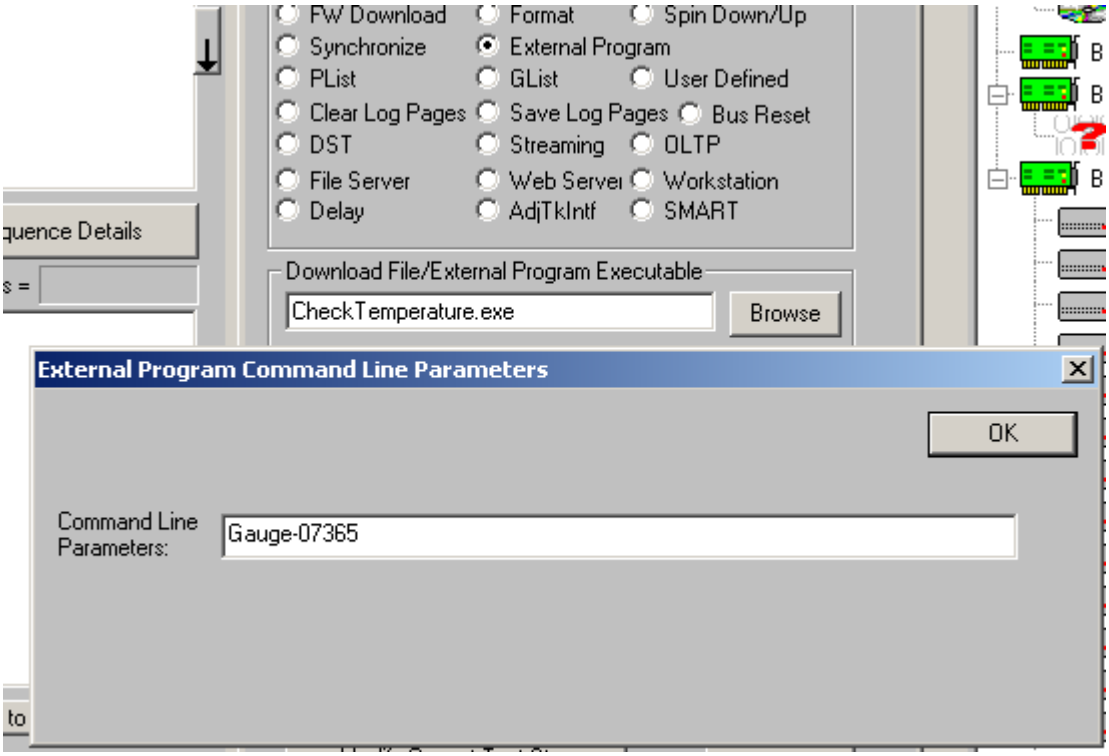
This code simply walks through the command-line arguments looking first for the string “HBA” then stepping through the comma-delimited address, extracting the HostBusAdapter, Target, and LUN values which DMM passes, in the form:

"DMMSataSelfTest.exe HBA=4,TID=7,LUN=0,SLOT=255,-other arguments"

The first argument is the external program name, followed by "HBA=*n*,TID=*n*,LUN=*n*,SLOT=*n*", followed by any other agreements which are specified in DMM at the time of defining the DMM test sequence. In the case of our example program we only care about the address of the device and will not be passing or looking for any other arguments.

IMPORTANT: The string that you enter in the edit box in the "External Program Command Line Parameters" dialog is appended to the HBA:Target:Lun:Slot information that is automatically passed to your external program. By way of an example, if device HBA=4,Target=7,Lun=0 is being tested, then the actual command line parameter your External Program will receive will be

"CheckTemperature.exe HBA=4,TID=7,LUN=0,SLOT=255,Gauge-07365"



Issuing a command via DTB

// send a self-test command, wait 2 minutes so it should be done

Sales: 720.249.2641
General: 303.972.2072

```
// we are using SAT to send the ATA command to a SATA drive connected to a SAS SAT-
//Compliant HBA
                Cmd[0] = 0xA1;
                Cmd[1] = 0x06;
                Cmd[2] = 0x00;
                Cmd[3] = 0xD4;
                Cmd[4] = 0x00;
                Cmd[5] = 0x01;
                Cmd[6] = 0x4F;
                Cmd[7] = 0xC2;
                Cmd[8] = 0x00;
                Cmd[9] = 0xB0;
                Cmd[10] = 0x00;
                Cmd[11] = 0x00;
                Cmd[12] = 0x00;
                Cmd[13] = 0x00;
                Cmd[14] = 0x00;
                Cmd[15] = 0x00;
                BufSize = 0;

// use DTB SCSI User Defined CDB to send the command to the drive
nRetCode = VCSCSIUserCdbTimeout(ha, tid, lun, &Cmd[0], 12, DataDir, BufSize,0,15);
if (nRetCode != 1) // DTB will return 1 on success - so fail this test if it isn't 1
        return 1;

nRetCode = 0; // make sure we pass back "success" to DMM
```


Sales: 720.249.2641
General: 303.972.2072

Note that in the DMM environment we are talking to SATA drives via a SAT-compatible SAS HBA, and so we will use SAT to issue the command to the drive. This means we will embed the ATA task register command into a 12-byte SCSI command. We use the DTB VCSCSIUserDdbTimeout() function to issue the CDB with a 15 second timeout.

The command we are issuing is an ATA SMART short Self-Test Immediate in off-line mode. Refer to the ATA documents to see how you could change this command to issue an extended or conveyance self-test, in either off-line or captive mode.

Use this SAT form of SATA command as a template to issue any ATA task register command. Refer to the STB Document “UsingSAT.pdf” for more detail on issuing ATA commands with and without data phases using this method.

Returning information back to DMM log files

```
// the following lines do what is needed to send info back to the DMM log file
// the file name is important - you need to specify which devices log file this info will go to
//
strMMFName.Format("DMM_HBA%02dTID%03dLUN%02d",ha,tid,lun);
HANDLE hMMF = OpenFileMapping(FILE_MAP_ALL_ACCESS,
      FALSE,
      strMMFName);
if (!hMMF)
{
         dwLastError = GetLastError();
         hMMF = CreateFileMapping((HANDLE)INVALID_HANDLE_VALUE,
            NULL,
           PAGE_READWRITE,
           0,
           0x1000,
```


Sales: 720.249.2641
General: 303.972.2072

```
            strMMFName);
}
if (hMMF)
{
  char * pBlah = (char *)MapViewOfFile(hMMF,
                        FILE_MAP_ALL_ACCESS,
                        0,
                        0,
                        0x1000);
// we can only send a max of 4096 bytes via this method - so make sure we don't try to send more
MyStrLength = strlen(m_strDMMText);
if (MyStrLength > 4096)
        MyStrLength = 4096;
int * pIntPtr = (int *)pBlah;
*pIntPtr = 1;
strcpy(&pBlah[4],m_strDMMText);
FlushViewOfFile(pBlah,MyStrLength);
UnmapViewOfFile(pBlah);
CloseHandle(hMMF);
}
else
dwLastError = GetLastError();
```

Your program will always return status information to DMM via the exit code of the program, which DMM will interpret as passed or failed –

Ability to receive an ascii string from the program that will be logged to the DMM logfile:

Sales: 720.249.2641
General: 303.972.2072

After the external program has completed, DMM will retrieve, if available, an ascii string from a system memory-mapped file. The external program, prior to exiting, must store the ascii string into the system memory-mapped file so that DMM can retrieve it. The name of the memory mapped file will have the format “DMM_HBAnnTIDxxxLUNyy” (for example DMM_HBA03TID007LUN00). The size of the data in the memory-mapped file will be no larger than 4K.

The code snip above can be cut and pasted into your application – simply format the text data you want logged into the string m_strDMMText and it will be printed into each devices log file.

Sales: 720.249.2641
General: 303.972.2072