GOTrust Technology Inc.

SESDAPI User Guide

Revision	：1.0
Established Date	：2014-09-23
Revised Date	：See Document Revision History
Classification	：Confidential

Revision History

Revision	Issue Date	Author	Comments
1.0	2014-09-23	Louis	First release.

Table of Contents

1 Introduction to SESDAPI SDK

1.1 Overview

This document describes the architecture of the SDK and the usage of example programs.

1.2 Directory Structure

The SDK includes the following items:

- Android\
  - C\
    - libs\armeabi: the library files for armeabi platform (i.e., market Android phones).
    - libs\x86: the library files for x86 platform.
    - SDSCInclude: the header files.
    - Example: the example program.
  - JNI\
    - libs\armeabi: the library files for armeabi platform (i.e., market Android phones).
    - libs\x86: the library files for x86 platform.
    - Example: the example program.
- Linux\
  - x86: the library files for x86 platform.
  - x64: the library files for x86_64 platform.
  - SDSCInclude: the header files.
  - Example: the example program.
- Windows\
  - Windows_x86: the library files for x86 platform.
  - Windows_x64: the library files for x64 platform.

  - ■ SDSCInclude: the include files.
  - ■ Example: the example program.
- Document\
  - ■ SESDAPI_Android_API_Manual: Android JNI API manual
  - ■ SESDAPI_Linux_API_Manual: Linux API manual
  - ■ SESDAPI_Windows_API_Manual: Windows API manual
- Tools: the secure microSD testing tools.

2 API Usage

2.1 Example Usage Flow

The following figure shows an example flow while communicating with GO-Trust secure microSD via this API. Every step in the flow chart also highlights the corresponding function of the API.

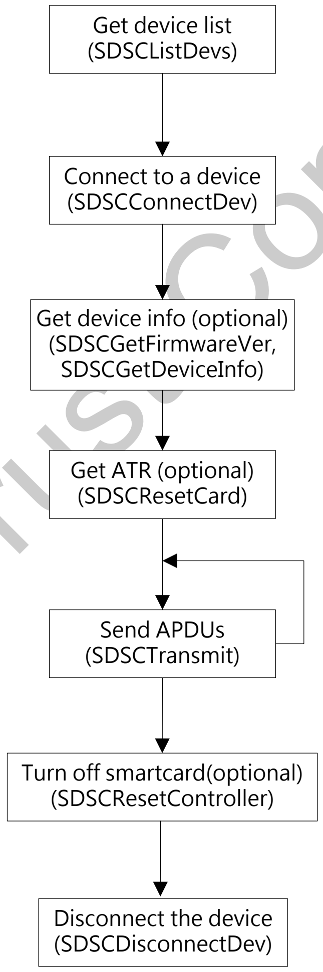


Figure 1 API usage flow chart

The details of each step are described as follows:

1. Call SDSCListDevs to get a list of secure microSD devices which are not connected yet.
2. Call SDSCConnectDev to establish a connection to a secure microSD.
3. Call SDSCGetFirmwareVer or SDSCGetDeviceInfo if you want to get the info of the secure microSD.
4. Call SDSCResetCard before starting to send APDUs.
   If the secure microSD has not been reset, this library will call SDSCResetCard automatically when the application firstly calls SDSCConnectDev.
5. Call SDSCTransmit to send APDUs.
6. Call SDSCResetController before SDSCDisconnectDev if you want to turn off the secure microSD.
7. Call SDSCDisconnectDev to close the connection to the secure microSD.

2.2 State Transition Diagram

The complete state transition diagram is shown in Figure 2. From this diagram developers can understand which functions are applicable for each state.

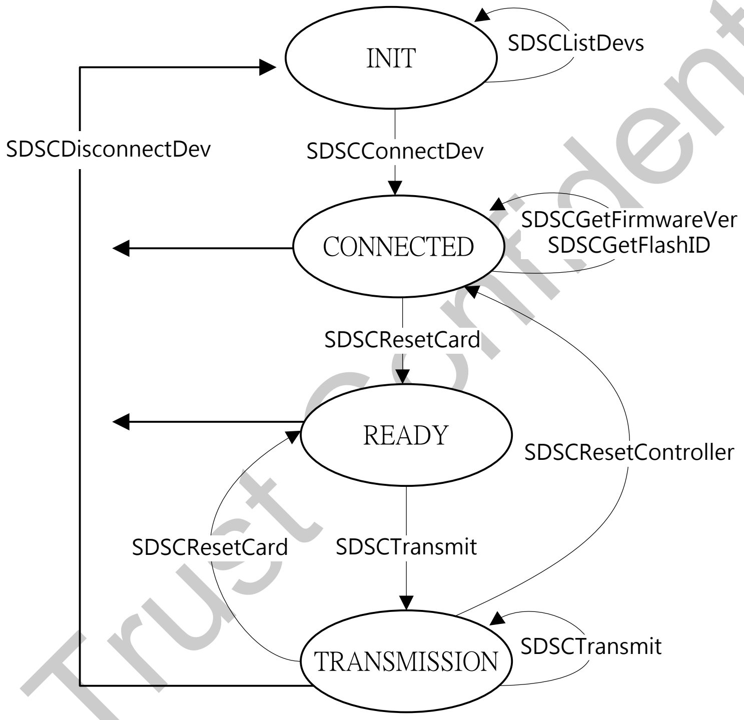


Figure 2 State transition diagram

SDSCDisconnectDev can be called at any time after the connection is established. This may be useful when any errors occur during communication and you want to restart a new connection. Another thing which should be paid attention to is that the entering of the CONNECTED state equals a cold reset for the GO-Trust secure microSD, while SDSCResetCard equals a warm reset.

2.3 Suggested Programming Model

Please note that a secure microSD can support only one connection at a time. API will return the error code SDSC_DEV_OCCUPIED_ERROR if an application tries to connect to a secure microSD which is already connected by another application or thread. If developers have the requirement of using secure microSD in multi-thread or multi-application manners, it is suggested that developers can develop a proxy application to handle sharing one connection with multiple clients. Please refer to the following figure for this concept.

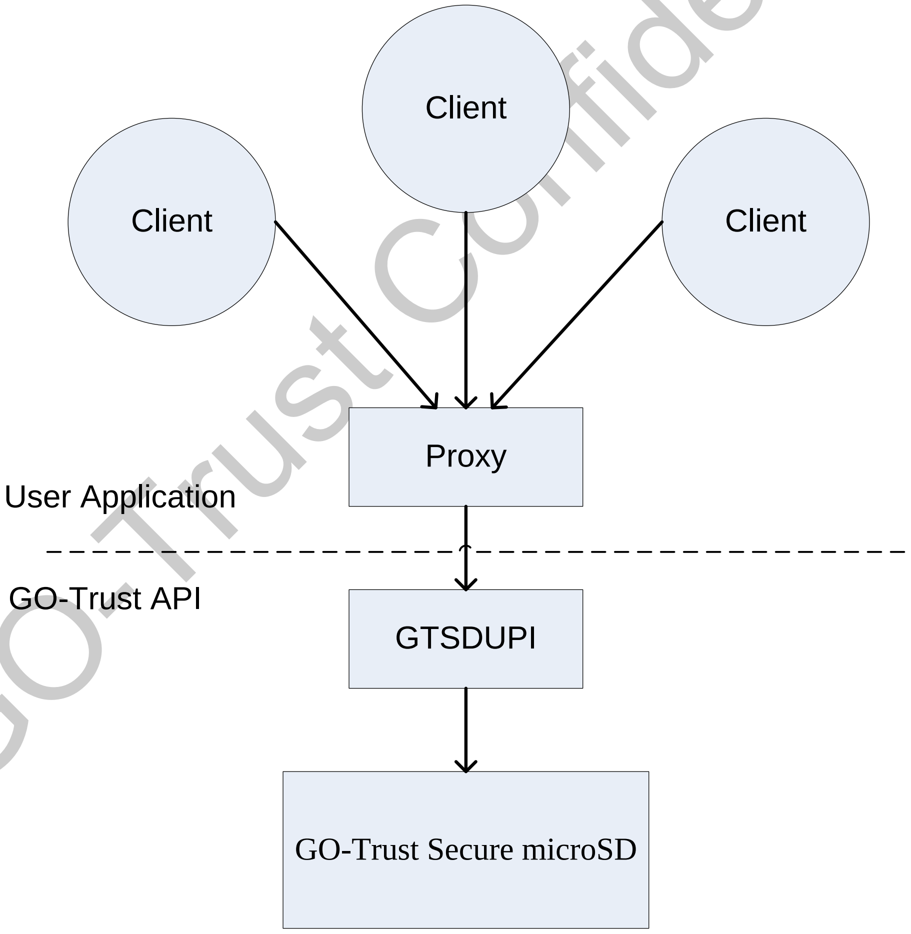


Figure 3 Suggested Programming Model

2.4 I/O File of Secure microSD

The connection between host applications and the secure microSD is established on a file on the secure microSD. Once the connection is established, the chosen file becomes a special I/O file for exchanging secure microSD commands and responses. The action for a secure microSD to enable a regular file as an I/O file is called “Bind”. Basically, any regular file on secure microSD can be bound as an I/O file. The only limitation is that the file name must be in 8.3 format and all English letters must be in upper case. In addition, the I/O file size and content may be modified during communication with the secure microSD, so developers must avoid binding a file which contains important user data as an I/O file.

To bind a file as an I/O file, an application can call the functions SDSCListDevs or SDSCConnectDev and pass the file path as a parameter. An application can re-bind another file at runtime by calling SDSCListDevs or SDSCConnectDev again, and the previous I/O file will return to a normally regular file.

Please note that the feature of binding any files as I/O file is supported by secure microSD with firmware version “0x8002530033000114” (or newer), while the older firmware versions can only use fixed-path I/O file.

2.4.1 I/O File Limitations on Android 4.4.2

There are two additional limitations to the I/O file on Android 4.4.2 (Kitkat). The first one is that Android 4.4.2 (Kitkat) restricts the microSD access permission of an application to only its own directory. Because of this restriction, the I/O file can only be located under the application own directory. The directory is named by the package name of the application. For example, if the package name of an application is “com.gotrust”, then its own directory will be “<microSD path>/Android/data/com.gotrust”.

The second limitation is that the I/O file size must be 1.99GB (2147483136 bytes). That is because Android 4.4.2 disables direct I/O, and therefore SESDAPI needs to use a larger I/O file to avoid file cache. Developers can use the function SDSCCreateIoFile to create the large I/O file. It is said that Android 4.4.3 will enable direct I/O again. If that is true, then the I/O file can be truncated to originally small size.

3 Usage of Example Programs

This section illustrates how to build and run the example programs included in the SDK.

3.1 Windows Example

3.1.1 Development Environment

Microsoft Visual C++ 2005 is required to build this example program.

3.1.2 Building Program

Step 1: Open project

Open the project file SDSCTest\SDSCTest.sln with Visual C++ 2005.

Step 2: Install the library

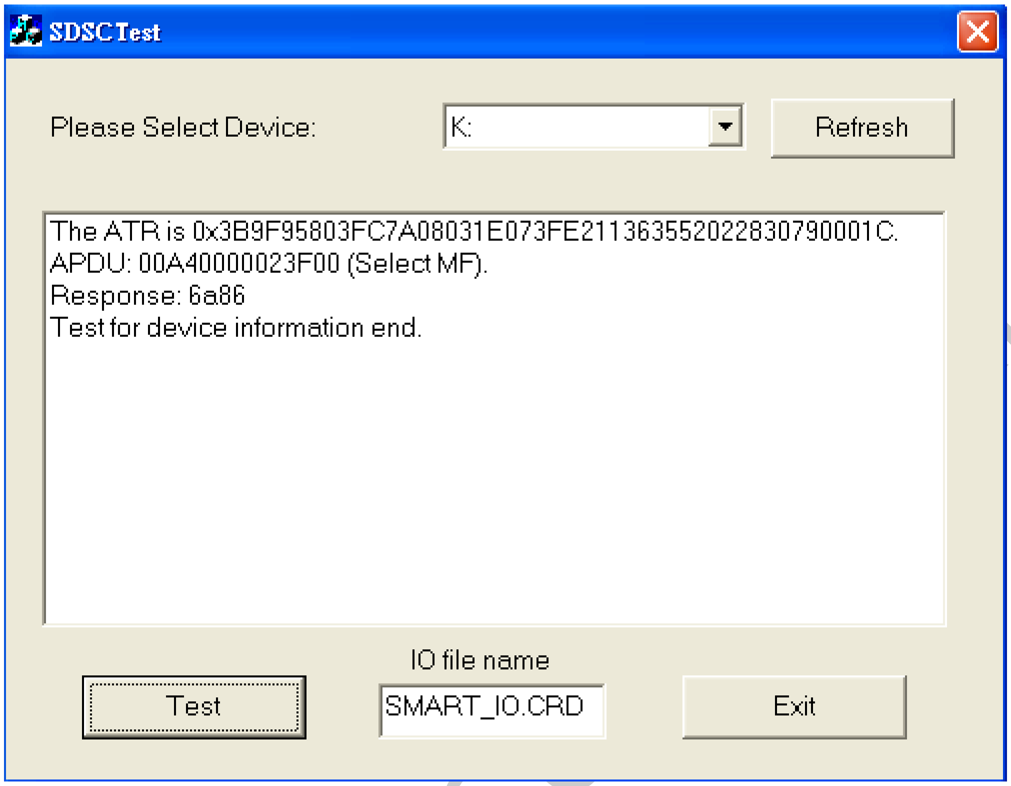
Copy SESDAPI.dll into the SDSCTest folder.

Step 3: Build the example program

In Visual C++ window, open menu “Debug” and click “Start without debugging”.

Step 4: Run the example

Click the ‘Test‘ button to test the GO-Trust secure microSD in the SDSCTest window.



```
The ATR is 0x3B9F95803FC7A08031E073FE21136355202283079000 1C.
APDU: 00A40000023F00 (Select MF).
Response: 6a86
Test for device information end.
```

Figure 4 Windows example result

3.2 Android – JNI Example

3.2.1 Development Environment

Please install Android SDK and the Eclipse IDE tool first before building the Android – JNI example program.

3.2.2 Building Program

Step 1: install Java Development Kit

Java Development Kit (JDK) is required for both Android SDK and Eclipse IDE.

Step 2: Install Android SDK and Eclipse

Download ADT bundle eclipse from https://developer.android.com/sdk/index.html and install it.
The ADT Bundle includes everything you need to begin developing apps.

Step 3: Configure your phone

Connect your PC with your phone via the USB cable, and enable USB debugging mode.
Install the ADB interface driver to make the connection between your PC and phone.
Then, type “adb devices” in console to detect your phone.

Step 4: Import the example project

Open Eclipse and import the example project.

Step 5: Configure the paths of libraries.

Open [Project] → [Properties] → [Java Build Path] tab and set the path of SESDAPI.jar:
Copy SESDAPI.jar into SDSCTest\project\libs.
Copy libSESDAPI.so into SDSCTest\project\libs\ armeabi.

Step 6: Run the example on your phone.

Click [Run] → [Run As] → [Android Application] and Eclipse will automatically install the APK to your phone and run it.

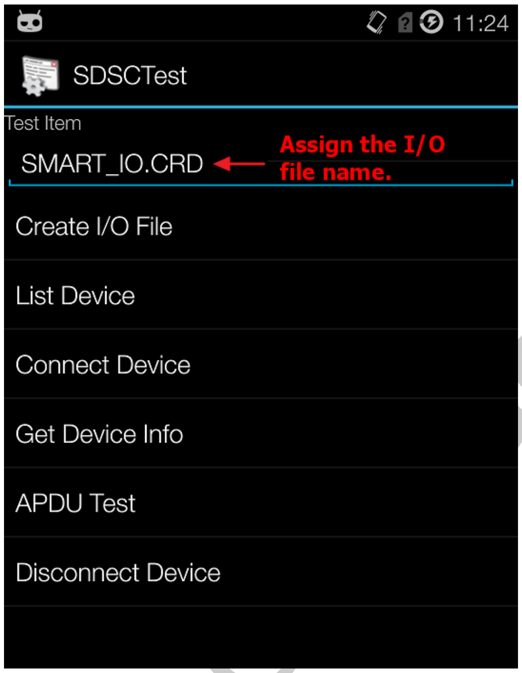
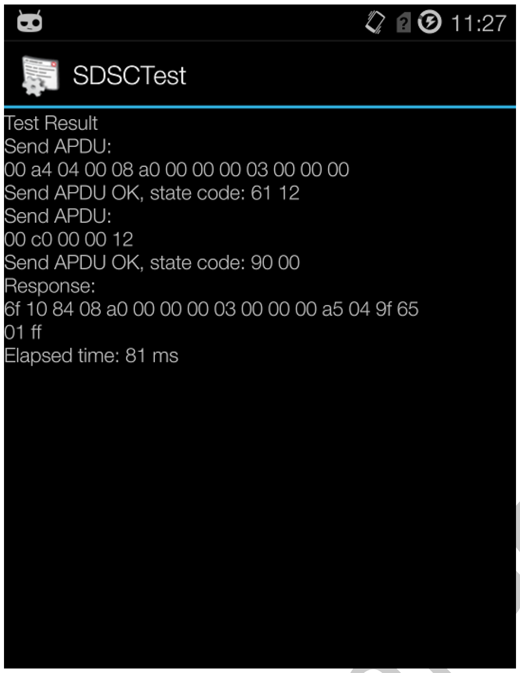


Figure 5 Android-JNI Example: Initial page



```
Test Result
Send APDU:
00 a4 04 00 08 a0 00 00 00 03 00 00 00
Send APDU OK, state code: 61 12
Send APDU:
00 c0 00 00 12
Send APDU OK, state code: 90 00
Response:
6f 10 84 08 a0 00 00 00 03 00 00 00 a5 04 9f 65
01 ff
Elapsed time: 81 ms
```

Figure 6 Android-JNI Example: Test result

3.3 Android– C Example

3.3.1 Development Environment

To build the Android – C example program, Android NDK are required. You can install Android NDK on Windows or Linux. In addition, GNU "make" utility must be installed because it is required for building programs.

3.3.2 Building Program

Step 1: Download and install Android NDK

Download Android NDK and extract the downloaded NDK archive, and add PATH environment variable of Android NDK.

Step 2: compiler and run the example on your phone.

Start console and switch to <SESDAPI_SDK_Root>\Android\ARM\C\Example. Type the following command to build the example program:

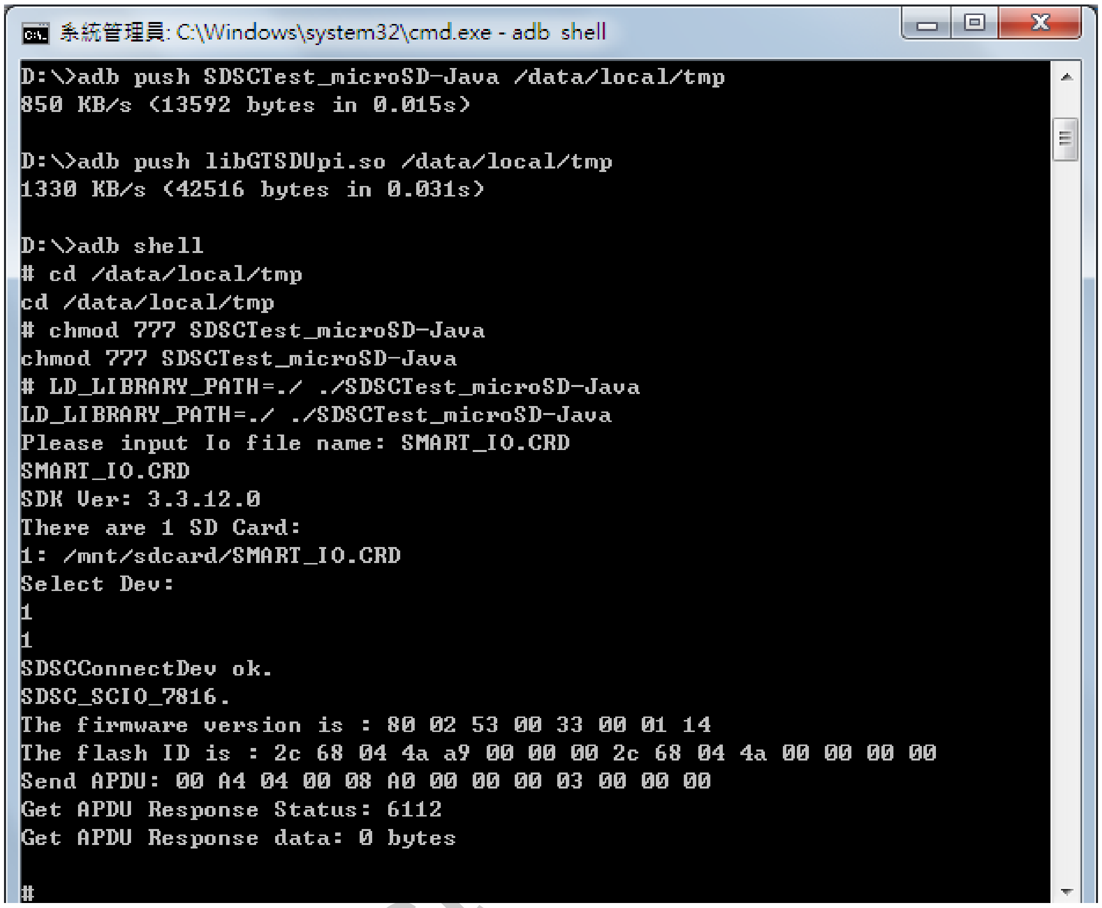
```
<NDK_Root>\ndk-build
```

Then upload the library and SDSCTest to your phone:

```
adb push libSESDAPI.so /data/local/tmp
adb push SDSCTest /data/local/tmp
adb shell chmod 777 /data/local/tmp/SDSCTest
```

Finally, run “adb shell” to login the phone’s shell and type the following command to run the example program.

```
LD_LIBRARY_PATH=/data/local/tmp/ /data/local/tmp/SDSCTest
```



系統管理員: C:\Windows\system32\cmd.exe - adb shell

```
D:\>adb push SDSCTest_microSD-Java /data/local/tmp
850 KB/s (13592 bytes in 0.015s)

D:\>adb push libGTSDUpi.so /data/local/tmp
1330 KB/s (42516 bytes in 0.031s)

D:\>adb shell
# cd /data/local/tmp
cd /data/local/tmp
# chmod 777 SDSCTest_microSD-Java
chmod 777 SDSCTest_microSD-Java
# LD_LIBRARY_PATH=./ ./SDSCTest_microSD-Java
LD_LIBRARY_PATH=./ ./SDSCTest_microSD-Java
Please input Io file name: SMART_IO.CRD
SMART_IO.CRD
SDK Ver: 3.3.12.0
There are 1 SD Card:
1: /mnt/sdcard/SMART_IO.CRD
Select Dev:
1
1
SDSCConnectDev ok.
SDSC_SCIO_7816.
The firmware version is : 80 02 53 00 33 00 01 14
The flash ID is : 2c 68 04 4a a9 00 00 00 2c 68 04 4a 00 00 00 00
Send APDU: 00 A4 04 00 08 A0 00 00 00 03 00 00 00
Get APDU Response Status: 6112
Get APDU Response data: 0 bytes

#
```

Figure 7 Android-C example result

NOTICE:

On Android 4.0 or above versions, a console program may not have the access permission to a microSD. Developers need to use a terminal application (such as Android Terminal Emulator) which has the permission WRITE_EXTERNAL_STORAGE to run the example program.

3.4 Linux Example

3.4.1 Development Environment

GNU gcc toolchain is required to build this example program.

3.4.2 Building Program

Step 1: Install the GNU gcc

Install the gnu gcc, then start console and typing “gcc -v” to check install gcc successfully.

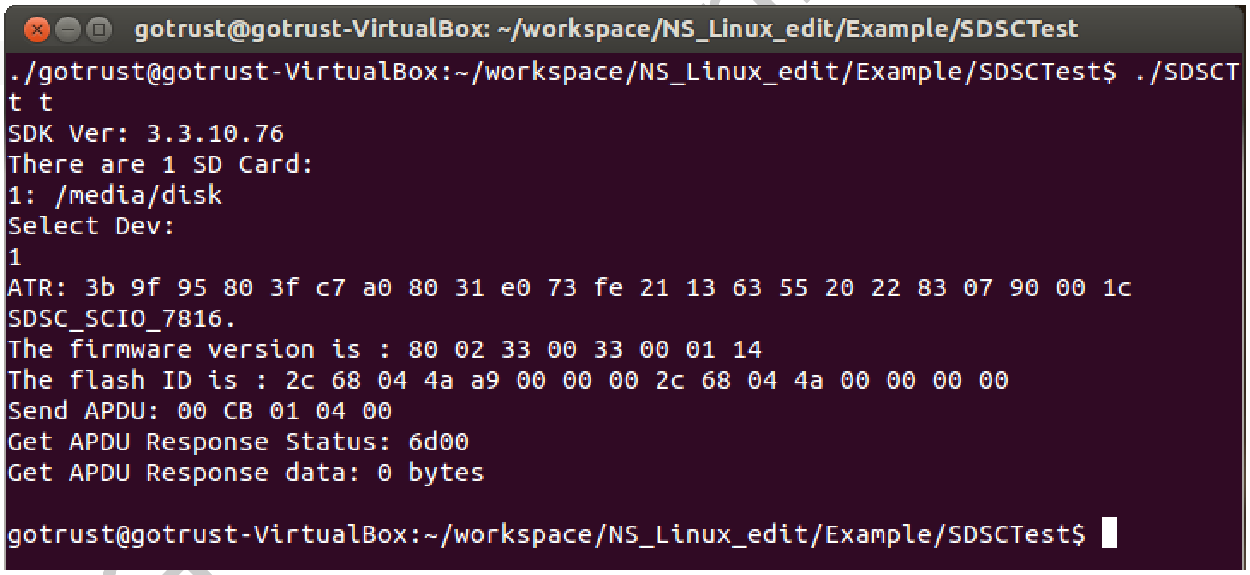
Step 2: Build the example program

Start console and then switch to the SDSCTest example directory. Type the following command to build the example program:

```
make
```

Step 3: Run test functions

Type “./SDSCTest” to run the example.



gotrust@gotrust-VirtualBox: ~/workspace/NS_Linux_edit/Example/SDSCTest

```
./gotrust@gotrust-VirtualBox:~/workspace/NS_Linux_edit/Example/SDSCTest$ ./SDSCT
t t
SDK Ver: 3.3.10.76
There are 1 SD Card:
1: /media/disk
Select Dev:
1
ATR: 3b 9f 95 80 3f c7 a0 80 31 e0 73 fe 21 13 63 55 20 22 83 07 90 00 1c
SDSC_SCIO_7816.
The firmware version is : 80 02 33 00 33 00 01 14
The flash ID is : 2c 68 04 4a a9 00 00 00 2c 68 04 4a 00 00 00 00
Send APDU: 00 CB 01 04 00
Get APDU Response Status: 6d00
Get APDU Response data: 0 bytes

gotrust@gotrust-VirtualBox:~/workspace/NS_Linux_edit/Example/SDSCTest$ 
```

Figure 8 Linux example result