SecuGen

Please read this document first

How to Develop Applications Compatible with All SecuGen® Fingerprint Readers

Contents

SG1-0030D-000 (09/19)

Purpose of This Document

The SecuGen FDx SDK Pro provides developers with a programming model for building applications that will support all SecuGen fingerprint readers including current, future and even legacy products.

This document describes best practices to help users have a seamless, uninterrupted experience in deploying and operating your application with any SecuGen fingerprint reader–now and in the future–without having to modify application source code.

Using the approach described in this document together with the most recent SecuGen SDK libraries and device drivers, it is possible to develop an application that works with any SecuGen reader attached to the system.

Recommended Application Flow

The following steps are recommended to ensure that the application can use any SecuGen fingerprint reader without code changes.

1. Instantiate an SGFPM (SecuGen Fingerprint Module) object.
2. Enumerate all SecuGen fingerprint readers currently attached to the system.
3. Initialize the SGFPM object with the device name returned by the enumeration.
4. Open the fingerprint reader.
5. Get device information such as width and height of the image.
6. Capture a fingerprint image.
7. Close the fingerprint reader.
8. Destroy the SGFPM object.

Device Initialization

The approach that is used to initialize the SecuGen fingerprint reader that is connected to the system will determine whether your application will work with all SecuGen readers including those released in the future.

Refer to the following functions in **sgfplib.h**:

```
SGFPM_DLL_DECL DWORD WINAPI  SGFPM_Init(HSGFPM hFpm, DWORD
devName);
virtual DWORD WINAPI  Init(DWORD devName);
```

The parameter that is passed as *devName* will determine whether the application supports specific SecuGen devices or all SecuGen devices. Use the following example to support all devices.

devName: SGDeviceList array

Example:

```
// Enumerate all the readers
DWORD ndevs = 0;
SGDeviceList *devlist = NULL;
rc = SGFPM_EnumerateDevice(hFPM, &ndevs, &devlist);
if (rc != SGFDX_ERROR_NONE)
   //Handle error
else
{
   if (ndevs > 0) {
   // Initialize the SGFPM Object with the first attached
   // enumerated
   rc = SGFPM_Init(hFPM, devlist[0].DevName);
   // Or you can use: rc = SGFPM_Init(hFPM, SG_DEV_AUTO);
   // Initialize the SGFPM Object with the first attached
   // reader found

}
```

Sample Source Code

The following sample source code demonstrates the programming sequence described above. To compile this sample, link the **sgfplib.lib** import library and include the **sgfplib.h** header file in your C++ project. For more information about these files, refer to the ***FDx SDK Pro Programming Manual***.

```
void run_fdx_sdk_sample() {
	DWORD rc = 0;
	HSGFPM hFPM = NULL;

	// Create a SGFPM object
	rc = SGFPM_Create(&hFPM);
	if (rc != SGFDX_ERROR_NONE) {
		std::cout << "ERR: SGFPM_Create returns " << rc << std::endl;
	}

	// Enmerate all the readers
	DWORD ndevs = 0;
	SGDeviceList *devlist = NULL;
	rc = SGFPM_EnumerateDevice(hFPM, &ndevs, &devlist);
	if (rc != SGFDX_ERROR_NONE) {
		std::cout << "ERR: SGFPM_EnumerateDevice returns " << rc << std::endl;
	}
	std::cout << "Detected device count (" << ndevs << ")" << std::endl;

	if (ndevs > 0) {
		// Initialize the SGFPM Object
		rc = SGFPM_Init(hFPM, devlist[0].DevName); // The first device enumerated
		//Or you can use: rc = SGFPM_Init(hFPM, SG_DEV_AUTO); // The first device
found
		if (rc != SGFDX_ERROR_NONE) {
			std::cout << "ERR: SGFPM_Init returns " << rc << std::endl;
		}

		// Open the fingerprint reader
		rc = SGFPM_OpenDevice(hFPM, USB_AUTO_DETECT);
		if (rc != SGFDX_ERROR_NONE) {
			std::cout << "ERR: SGFPM_OpenDevice returns " << rc << std::endl;
		} else {

			// Get device information
			SGDeviceInfoParam devInfo = { 0 };
			rc = SGFPM_GetDeviceInfo(hFPM, &devInfo);
			if (rc != SGFDX_ERROR_NONE) {
				std::cout << "ERR: SGFPM_GetDeviceInfo() returns " << rc <<
std::endl;
			} else {
				std::cout << "image width = " << devInfo.ImageWidth <<
std::endl;
				std::cout << "image height = " << devInfo.ImageHeight <<
std::endl;
				std::cout << "serial number = " << std::string((const
char*)devInfo.DeviceSN) << std::endl;
			}

			// LED On/Off
			for (int i = 0; i < 5; i++) {
				SGFPM_SetLedOn(hFPM, TRUE);
				Sleep(1 * 500);      // 500 ms
				SGFPM_SetLedOn(hFPM, FALSE);
				Sleep(1 * 500);
			}
		}

		// Close the device
		SGFPM_CloseDevice(hFPM);
	}

	// Destory the SGFPM object
	SGFPM_Terminate(hFPM);
```