Programming Manual for FDx SDK *Pro* for Android

For applications using SecuGen® fingerprint readers

SG1-0036A-007 (04/18)

Contents

Before You Begin

SecuGen SDK

SecuGen is proud to provide software developers with world class tools for the development of fingerprint biometric software applications. All SecuGen 1:1 SDKs, including this SDK, are provided completely free of charge for use only with SecuGen fingerprint readers.

Biometrics Overview

Biometrics is a method of recognizing a person based on physical or behavioral characteristics. Biometric information that is used identify people includes fingerprint, voice, face, iris, handwriting and hand geometry.

There are two key functions offered by a biometric system. One method is **identification**, a “one-to-many” (1:N) matching process in which a biometric sample is compared sequentially to a set of stored samples to determine the closest match. The other is **verification**, a “one-to-one” (1:1) matching process in which the biometric system checks previously enrolled data for a specific user to verify whether that individual is who he or she claims to be. The verification method provides the best combination of speed and security, especially where multiple users are concerned, and requires a user ID or other identifier for direct matching.

With an increasing reliance on online and mobile technology and other shared resources, more and more transactions of all types are initiated and completed online and remotely. This unprecedented growth in electronic transactions has underlined the need for a faster, more secure and more convenient method of user verification than passwords can provide. Using biometric identifiers offers advantages over traditional methods. This is because only biometric authentication is based on the identification of an intrinsic part of a human being. Tokens such as smart cards, magnetic stripe cards and physical keys, can be lost, stolen, duplicated or left behind. Passwords can be forgotten, shared, hacked or unintentionally observed by a third party. By eliminating these potential trouble spots, biometric technology can provide greater security, with convenience, needed for today’s complex electronic landscape.

Advantages of Using Fingerprints

The advantages of using fingerprints include widespread public acceptance, convenience and reliability. It takes little time and effort to scan one’s fingerprint with a fingerprint reader, and so fingerprint recognition is considered among the least intrusive of all biometric verification techniques. Ancient officials used thumbprints to seal documents thousands of years ago, and law enforcement agencies have been using fingerprint identification since the late 1800s. Fingerprints have been used so extensively and for so long, there is a great accumulation of scientific data supporting the idea that no two fingerprints are alike.

About SecuGen

SecuGen (www.secugen.com) provides biometric solutions for physical and network security employing advanced fingerprint recognition technology. The company’s comprehensive product line includes high quality optical fingerprint readers and sensor component, software and development kits that are used for a variety of innovative applications including Internet, enterprise network and desktop security, physical access control, time and attendance management and financial and medical records control. SecuGen patented products feature the industry’s longest warranty and are renowned for their accuracy, reliability and versatility. Based in Silicon Valley, SecuGen has been serving the global biometric community since 1998 and is an active member of the Biometrics Consortium (www.biometrics.org) and the BioAPI Consortium (www.bioapi.org).

About SecuGen Products

SecuGen Sensor Qualities

- **Excellent Image Quality:** Clear, distortion-free fingerprint images are generated using advanced, patent-pending optical methods. Quality imaging yields better sampling for minutiae data extraction.
- **Durability:** Mechanical strength tests show resistance to impact, shock and scratches.
- **Powerful Software:** Precise, fast processing algorithm ensures efficiency and reliability.
- **Ruggedness and Versatility:** Solid engineering and superior materials allows for use under extreme conditions.
- **Ergonomic Design:** Compact, modular design for seamless integration into small devices, ease of use and compatibility make it ideal for a broad range of applications.
- **Low Cost:** Products are developed to deliver high performance, zero maintenance at very affordable prices for general and industrial use.

Advantages of SecuGen Sensors Over Other Optical Sensors

- Unique optical method captures fine details, even from dry skin
- Extremely low image-distortion
- Reinforced materials
- Wear resistance
- Attractively small size
- Ease of integration
- Ready-to-use
- Low cost through longer life and no maintenance requirements

Advantages SecuGen Sensors Over Semiconductor (Capacitive) Sensors

- Non-metal, non-silicon components make it less susceptible to corrosion when exposed to salts, oil and moisture from skin and environment
- Superior surface properties eliminate need for costly coating and processing procedures
- Greater mechanical strength, wear-resistance and durability
- Broader range of applicability, especially for use in extreme conditions and climates
- Immunity from electrostatic discharge
- Low cost through longer life and no maintenance requirements

Strengths of SecuGen Software and Algorithms

- Unique image processing algorithm extracts fingerprint minutiae very accurately
- High signal-to-noise ratio processing algorithm screens out false features
- Highly efficient matching algorithm
- Fast overall process of extraction, matching and verification
- Encryption function to protect user privacy
- Compatibility with existing desktop, laptop PCs interface computers
- Ease in developing applications for various purposes

Chapter 1. Overview

SecuGen's FDx SDK *Pro* is designed to provide low level access to SecuGen's fingerprint readers using SecuGen's next-generation algorithm module. Programming with SecuGen's FDx SDK *Pro* is simple and easy to program and gives the most development flexibility among all SecuGen SDKs.

1.1. Features

- Uses SecuGen's new and improved next-generation algorithms
- Supports three kinds of fingerprint minutiae formats (or templates):
  - SG400: SecuGen's proprietary fingerprint minutiae format
  - ANSI378: Finger Minutiae Format for Data Exchange (ANSI INCITS 378-2004)
  - ISO19794-2: Biometric Data Interchange Formats--Finger Minutiae Data (ISO/IEC 19794-2:2005)
- Provides low-level APIs for image capture, feature extraction and matching
  - The following extraction and matching algorithms, which are incorporated in sgfpamx.dll in this SDK, support the ANSI INCITS 378-2004 standard and have been tested and proven to be MINEX Compliant (http://fingerprint.nist.gov/MINEX/):
    - SecuGen ANSI INCITS 378 Template Generator v3.5 (feature extraction algorithm)
    - SecuGen ANSI INCITS 378 Template Matcher v3.5 (matching algorithm)
- Gives a high degree of flexibility to developers of all kinds of applications and is easy to use

1.2. System Requirements

SecuGen USB readers capture a fingerprint image and digitize the image to an 8-bit gray-scale image at 500 DPI resolution. The host system then retrieves the image through its USB port for subsequent processing. SecuGen Hamster Plus™ (HSDU03P), Hamster IV™ (HSDU04P), Hamster PRO (HUPx), Hamster PRO 10™ (HU10) and Hamster PRO 20™ (HU20) USB readers are supported in this SDK. The following are the system requirements for Android devices using SecuGen USB readers:

Windows Development Environment
- IBM-compatible PC Pentium III or later
- Windows 7
- 1GB RAM
- 1GB available hard disk space
- Eclipse Helios Service Release 2
- Java JDK 1.6.0_24 or later
- Android SDK for Windows (installer_r20.0.3-windows)
- Android USB ADB Driver (adb_driver_x86XP_Eng_Multi)

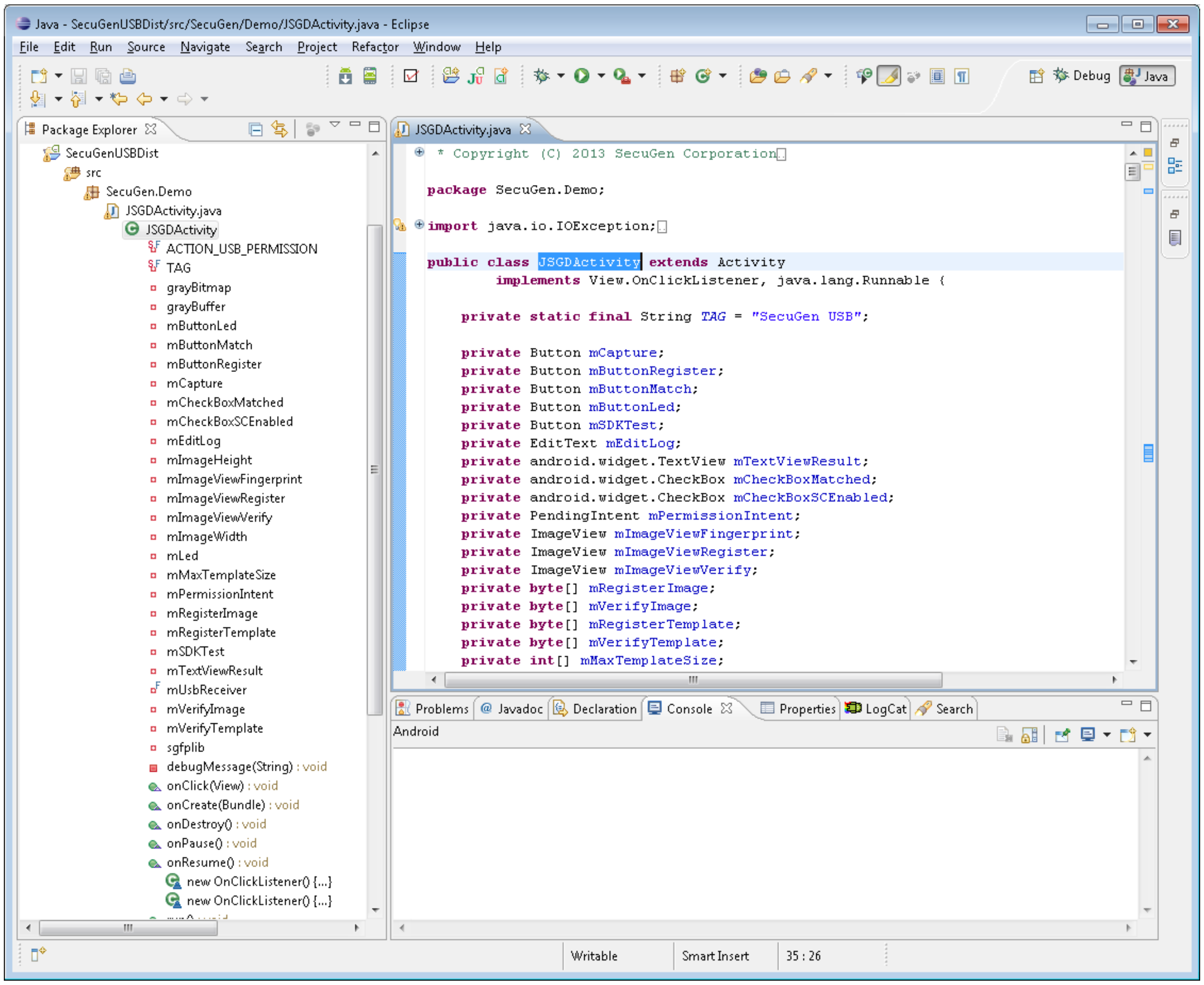
Android Device
- ARM based Android tablet or smart phone
- USB host controller on device
- Standard USB port or Micro-USB to USB OTG adapter
- Android OS Version 3.1 (Honeycomb) or later

Chapter 2. Installation

2.1. Installation

< WINDOWS>

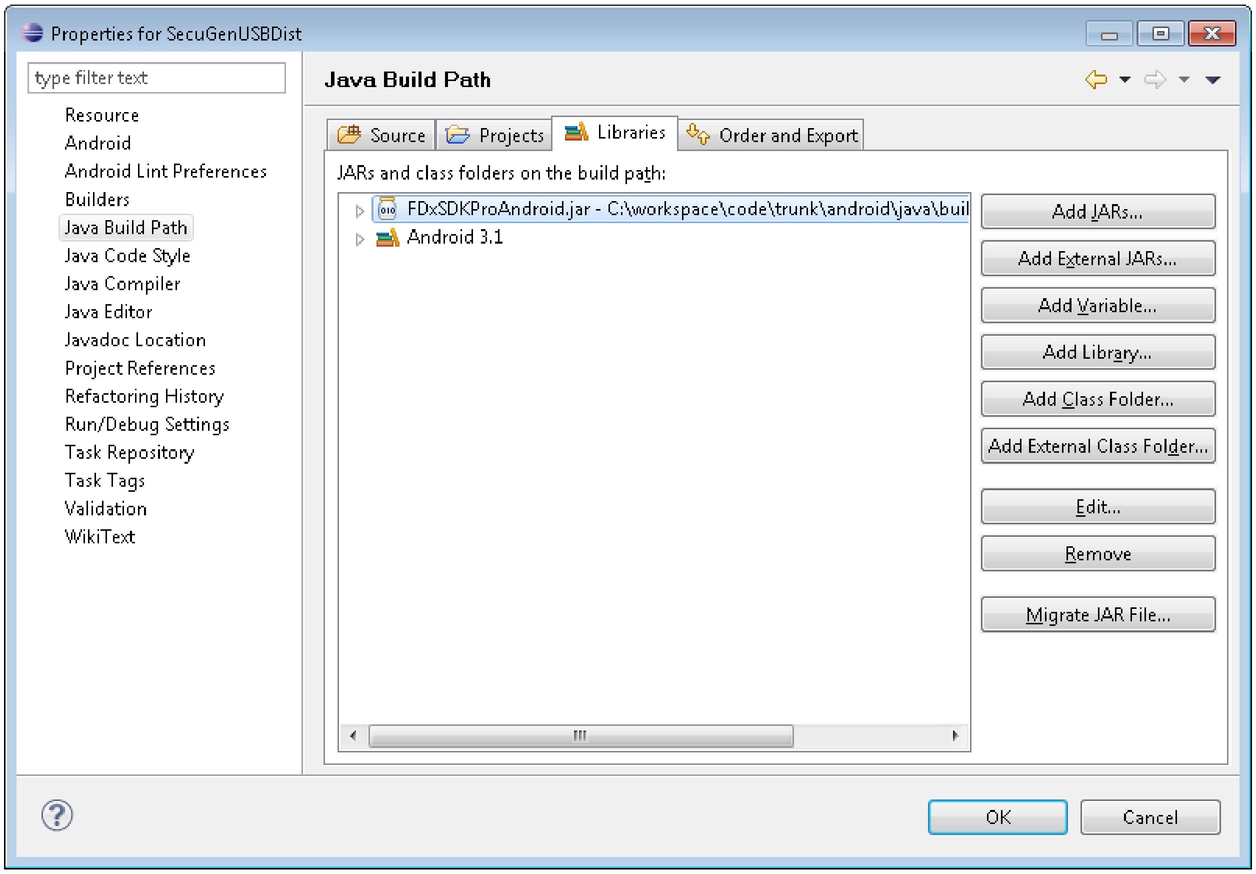
1. Import the "SecuGenUSBDist" project in the SDK distribution into your Eclipse environment. Refer to Appendix A for detailed instructions on importing the sample application into Eclipse.


Java - SecuGenUSBDist/src/SecuGen/Demo/JSGDActivity.java - Eclipse
File Edit Run Source Navigate Search Project Refactor Window Help
Debug
Java
Package Explorer
JSGDActivity.java
Problems
Javadoc
Declaration
Console
Properties
LogCat
Search
Android
Writable
Smart Insert
35 : 26

```
SecuGenUSBDist
  src
    SecuGen.Demo
      JSGDActivity.java
        JSGDActivity
          ACTION_USB_PERMISSION
          TAG
          grayBitmap
          grayBuffer
          mButtonLed
          mButtonMatch
          mButtonRegister
          mCapture
          mCheckBoxMatched
          mCheckBoxSCEnabled
          mEditLog
          mImageHeight
          mImageViewFingerprint
          mImageViewRegister
          mImageViewVerify
          mImageWidth
          mLed
          mMaxTemplateSize
          mPermissionIntent
          mRegisterImage
          mRegisterTemplate
          mSDKTest
          mTextViewResult
          mUsbReceiver
          mVerifyImage
          mVerifyTemplate
          sgfplib
          debugMessage(String) : void
          onClick(View) : void
          onCreate(Bundle) : void
          onDestroy() : void
          onPause() : void
          onResume() : void
            new OnClickListener() {...}
            new OnClickListener() {...}
```

```
* Copyright (C) 2013 SecuGen Corporation

package SecuGen.Demo;

import java.io.IOException;

public class JSGDActivity extends Activity
        implements View.OnClickListener, java.lang.Runnable {

    private static final String TAG = "SecuGen USB";

    private Button mCapture;
    private Button mButtonRegister;
    private Button mButtonMatch;
    private Button mButtonLed;
    private Button mSDKTest;
    private EditText mEditLog;
    private android.widget.TextView mTextViewResult;
    private android.widget.CheckBox mCheckBoxMatched;
    private android.widget.CheckBox mCheckBoxSCEnabled;
    private PendingIntent mPermissionIntent;
    private ImageView mImageViewFingerprint;
    private ImageView mImageViewRegister;
    private ImageView mImageViewVerify;
    private byte[] mRegisterImage;
    private byte[] mVerifyImage;
    private byte[] mRegisterTemplate;
    private byte[] mVerifyTemplate;
    private int[] mMaxTemplateSize;
```

2. Select project properties and add external jar "FDxSDKProAndroid.jar" to the "Java build Path" section.



JARs and class folders on the build path:

FDxSDKProAndroid.jar - C:\workspace\code\trunk\android\java\buil

Android 3.1

2.2. Included Files

Root directory

FDx SDK Pro Programming Manual (Android).pdf	This document
readme.txt	Release notes and important information about this SDK
USB Host Diagnostics (Android).pdf	Instructions to determine Android USB support level
FDxSDKProAndroid,jar	JAR file containing SecuGen SDK java classes
SecuGenUSB.apk	Installable Android demo application package

javadoc directory
HTML java documentation for the SecuGen class library

libs/armeabi directory
ARM based native libraries used by this SDK

libjnisgfdetect.so
libjnisgfdu03.so
libjnisgfdu04.so
libjnisgfdu05.so
libjnisgfplib.so
libjnisgnfiqlib.so
libsgfpamx.so

libs/x86 directory
Intel X86 based native libraries used by this SDK

- **libjnisgfdetect.so**
- **libjnisgfdu03.so**
- **libjnisgfdu04.so**
- **libjnisgfdu05.so**
- **libjnisgfplib.so**
- **libjnisgnfiqlib.so**
- **libsgfpamx.so**

SecuGenUSBDist directory
Contains Eclipse project for Android demo application

Chapter 3. Programming in Java

SecuGen's FDx SDK *Pro* was designed for ease in programming and most flexibility for developers. All SDK functions are integrated into the **JSGFPLib** class. The JSGFPLib class includes device initialization, fingerprint capture, minutiae extraction and matching functions.

3.1. Creating JSGFPLib

To use JSGFPLib, call **JSGFPLib()**, which instantiates a JSGFPLib object. Pass android.hardware.usb.UsbManager object to the constructor as a parameter.

```
private static final String ACTION_USB_PERMISSION = "com.android.example.USB_PERMISSION";
private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
       public void onReceive(Context context, Intent intent) {
          String action = intent.getAction();
          if (ACTION_USB_PERMISSION.equals(action)) {
             synchronized (this) {
                UsbDevice device = (UsbDevice)intent.getParcelableExtra(
                                                    UsbManager.EXTRA_DEVICE);
                if (intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                    if(device != null){
                        Log.d(TAG, "Vendor ID : " + device.getVendorId() + "\n");
                       Log.d(TAG, "Product ID: " + device.getProductId() + "\n");
                    }
                    else
                       Log.e(TAG, "mUsbReceiver.onReceive() Device is null");

                 }
                 else
                    Log.e(TAG, "mUsbReceiver.onReceive() permission denied for device "
                                + device);

             }
          }
       }
 };

PendingIntent mPermissionIntent = PendingIntent.getBroadcast(this, 0,
                       new Intent(ACTION_USB_PERMISSION), 0);
IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
registerReceiver(mUsbReceiver, filter);
JSGFPLibsgfplib = new JSGFPLib((UsbManager)getSystemService(Context.USB_SERVICE));
```

3.2. Initializing JSGFPLib

After the JSGFPLib object is created, it should be initialized using **JSGFPLiB,Init() JSGFPLib.Init()** takes the device name, loads the driver that corresponds to the device name and initializes the fingerprint algorithm module based on device information.

The table below summarizes the correlation among device name (device type), loaded device driver and initial image size when the **Init(JSGFPLibDeviceName devName)** function is called.

Device Name, Device Driver and Image Size

Device Name	Value	Device driver	Image Size (pixels)
SGDEV_FDU03	4	USB SDU03P driver	260*300
SGDEV_FDU04	5	USB SDU04P driver	258*336
SGDEV_FDU05	6	USB U20 driver	300*400
SGDEV_FDU06	7	USB UPx driver	260*300
SGDEV_FDU07	8	USB U10 driver	252*330

- **JSGFPLib.Init()**

```
long error = sgfplib.Init( SGFDxDeviceName.SG_DEV_AUTO);
```

3.3. Terminating JSGFPLib

JSGFPLib.Close() must be called prior to terminating the application. It frees up the memory used by the JSGFPLib object.

```
Long error = JSGFPLib.Close();
```

3.4. Opening the SecuGen Fingerprint Reader

To use a SecuGen fingerprint reader, call **JSGFPLib.OpenDevice().** The parameter (**devId**) of **JSGFPLib.OpenDevice()** can have different meanings depending on which type of fingerprint reader is used.

If only one USB fingerprint reader is connected to the PC, **devId** will be 0. If multiple USB fingerprint readers are connected to one PC, **devId** can range from 0 to 9. The maximum number of SecuGen USB readers that can be connected to one PC is 10.

In general, if only one USB reader is connected to the PC, then **0** or **USB_AUTO_DETECT** is recommended.

```
long error = sgfplib.OpenDevice(0);
```

3.5. Getting Device Information

Device information can be retrieved by calling **JSGFPLib.GetDeviceInfo()**, which obtains required device information such as image height and width. The device information is contained in the **SGDeviceInfoParam** structure.

```
SGDeviceInfoParam device_info;
error = JSGFPLib.GetDeviceInfo(device_info);
```

```
if (error == SGFDxErrorCode.SGSGFDX_ERROR_NONE)
{
    m_ImgWidth = device_info.ImageWidth;
    m_ImgHeight = device_info.ImageHeight;
}
```

3.6. Capturing a Fingerprint Image

After the reader is initialized, a fingerprint image can be captured. The JSGFPLib object provides three types of fingerprint image capture functions listed below. Captured fingerprints are 256 gray-level images, and image width and height can be retrieved by calling **JSGFPLib.GetDeviceInfo()**. The image buffer should be allocated by the calling application.

JSGFPLib.GetImage() captures an image without checking for the presence of a finger or checking image quality.
.

- **JSGFPLib.GetImage()**

```
[Example]
Byte[] buffer = new byte[m_ImageWidth*m_ImageHeight];
if  (JSGFPLib.GetImage(buffer)  ==  SGFDxErrorCode.SGSGFDX_ERROR_NONE)  //  Get  image  data
from device
{
   // Display image
   // Process image
}
```

JSGFPLib.GetImageEx() captures fingerprint images continuously, checks the image quality against a specified quality value and ignores the image if it does not contain a fingerprint or if the quality of the fingerprint is not acceptable. If a quality image is captured within the given time (the second parameter), JSGFPLib.GetImageEx() ends its processing.

- **JSGFPLib.GetImageEx()**

```
 [Example]
byte[] buffer = new byte[m_ImageWidth*m_ImageHeight];
long timeout = 10000;
long quality = 80;
if(JSGFPLib.GetImageEx(buffer, timeout, quality) == SGFDxErrorCode.SGFDX_ERROR_NONE)
{
   // Display image
}
```

3.7. Getting Image Quality

To determine the fingerprint image quality, use **GetImageQuality()**.

- **JSGFPLib.GetImageQuality()**

Int[] img_qlty;

```
JSGFPLib.GetImageQuality(ImageWidth, m_ImageHeight, fp_image, mg_qlty);
if (img_qlty[0] < 80)
    // Capture again
```

3.8. Smart Capture™ and Controlling Brightness

Depending on the fingerprint reader used, environmental factors and the specifications of the host system, the brightness of a fingerprint image may vary. The SecuGen device drivers use a technology called Smart Capture™ to dynamically adjust brightness to ensure the best image quality. Smart Capture is enabled by default.

To manually control the quality of a captured image, the image brightness should be adjusted by changing the brightness setting of the reader using **JSGFPLib.SetBrightness()**. This function is ignored if Smart Capture is enabled. Smart Capture can be disabled using **JSGFPLib.WriteData()**.

- **JSGFPLib. SetBrightness()**

```
JSGFPLib.SetBrightness(70);   // Set from 0 to 100.
```

3.9. Creating a Template

To register or verify a fingerprint, a fingerprint image is first captured, and then feature data (minutiae) is extracted from the image into a **template**. Minutiae are the unique core points near the center of every fingerprint, such as ridges, ridge endings, bifurcations, valleys and whorls.

Use **JSGFPLib.CreateTemplate()** to extract minutiae from a fingerprint image to form a template. The buffer should be assigned by the application. To get the buffer size of the minutiae, call **JSGFPLib.GetMaxTemplateSize().** It will return the maximum buffer size for data in one template. The actual template size can be obtained by calling **JSGFPLib.GetTemplateSize()** after the template is created. The **JSGFPLib.CreateTemplate()** API creates only one set of data from an image.

Note: Templates having the ANSI378 or ISO19794-2 format may be merged.

- **JSGFPLib.CreateTemplate()**

```
// Get a fingerprint image
err = JSGFPLib.GetImage(m_ImgBuf);

// Create template from captured image
err = JSGFPLib.GetMaxTemplateSize(maxTemplateSize);
byte[] minBuffer = new byte[maxTemplateSize[0]];

// Set information about template
SGFingerInfo finger_info;
finger_info.FingerNumber = SGFingerPosition.SG_FINGPOS_LI;
finger_info.ImageQuality = qlty[0];
finger_info.ImpressionType = SG_IMPTYPE_LP;
finger_info.ViewNumber = 1;

err = JSGFPLib.CreateTemplate(finger_info, m_ImgBuf, minBuffer);
```

3.10. Matching Templates

Templates are matched during both registration and verification processes. During registration, it is recommended to capture at least two image samples per fingerprint for a higher degree of accuracy. The minutiae data from each image sample can then be compared against each other (i.e. matched) to confirm the quality of the registered fingerprints. This comparison is analogous to a password confirmation routine that is commonly required for entering a new password.

During verification, newly input minutiae data is compared against registered minutiae data. Similar to the registration process, verification requires the capture of a fingerprint image followed by extraction of the minutiae data from the captured image into a template.

To match templates, FDx SDK *Pro* provides four kinds of matching functions. Each function requires two sets of template data for matching.

JSGFPLib.MatchTemplate():This function matches templates having the same format as the default format. When calling this function, each template should include only one sample (or view) per template. The default format is SG400 (SecuGen proprietary format) but can be changed by calling JSGFPLib.SetTemplateFormat().

JSGFPLib.MatchTemplateEx(): This function can match templates having different template formats. This function can also specify the template format for each template and can match templates that have multiple views per template.

JSGFPLib.MatchAnsiTemplate(): This function is the same as **JSGFPLib.MatchTemplateEx()** except that it supports only ANSI378 templates.

JSGFPLib.MatchIsoTemplate(): This fucntion is the same as **JSGFPLib.MatchTemplateEx()** except that it supports only ISO19794-2 templates.

Function	Template Format	Can match templates with different formats?
JSGFPLib.MatchTemplate	SG400 (System default)	No
JSGFPLib.MatchTemplateEx	Specified template format	Yes
JSGFPLib.MatchAnsiTemplate	ANSI378	No
JSGFPLib.MatchIsoTemplate	ISO19794-2	No

- **JSGFPLib.MatchTemplate()**

```
 byte[]RegTemplate1= new byte[maxTemplateSize[0]];
 byte[]RegTemplate2= new byte[maxTemplateSize[0]];
…
// Getfirst fingerprint image and create template from the image
err = JSGFPLib.GetImage(m_ImgBuf);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_RegTemplate1);

// Get second fingerprint image and create template from the image
err = JSGFPLib.GetImageEx(m_ImgBuf);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_RegTemplate2);

long sl = SGFDxSecurityLevel.SL_NORMAL;      // Set security level as NORMAL
boolean[] matched = new boolean[1];
err = JSGFPLib.MatchTemplate(m_ RegTemplate1, m_ RegTemplate2, sl, matched);
```

- **JSGFPLib.MatchTemplateEx()**

```
 byte[]RegTemplate1= new byte[maxTemplateSize[0]];
 byte[]RegTemplate2= new byte[maxTemplateSize[0]];
…
// Make SG400 template
err = JSGFPLib.SetTemplateFormat(SGFDxTemplateFormat.TEMPLATE_FORMAT_SG400);
err = JSGFPLib.GetImage(m_ImgBuf, 5000, NULL, qlty);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_RegTemplate1);

// Make ANSI378 template
```

```
err = JSGFPLib.SetTemplateFormat(TEMPLATE_FORMAT_ANSI378);
err = JSGFPLib.GetImage(m_ImgBuf);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_RegTemplate2);

long sl = SGFDxSecurityLevel.SL_NORMAL;      // Set security level as NORMAL
boolean[] matched = new boolean[1];
err = JSGFPLib.MatchTemplateEx(RegTemplate1,
                            SGFDxTemplateFormat TEMPLATE_FORMAT_SG400,
                            0,                // Must be 0 if template format is SG400
                            RegTemplate2,
                            SGFDxTemplateFormat TEMPLATE_FORMAT_ANSI378,
                            0,                // Currently only one sample
                            sl,
                            &matched);
```

- **JSGFPLib.MatchAnsiTemplate()**

```
Long err err;
boolean[] matched = new boolean[1];
matched[0] = false;
SGANSITemplateInfo sample_info = new SGANSITemplateInfo();
err = JSGFPLib.GetAnsiTemplateInfo(m_EnrollTemplate, sample_info);

boolean finger_found = false;
for (int i = 0; i < sample_info.TotalSamples; i++)
{
  if(sample_info.SampleInfo[i].FingerNumber == finger_pos)  // Try match for same finger
 {
    finger_found = true;
    err = JSGFPLib.MatchAnsiTemplate(m_EnrollTemplate,
                               i,
                               m_FetBufM,
                               0,
                               SGFDxSecurityLevel.SL_NORMAL
                               matched);
    if (matched)
      break;
  }
}
```

- **JSGFPLib.MatchIsoTemplate()**

```
long err;
boolean[] matched = new boolean[1];
matched[0] = false;

// ISO19794-2
SGISOTemplateInfo sample_info = new SGISOTemplateInfo();
err = JSGFPLib.GetIsoTemplateInfo(m_StoredTemplate, sample_info);

int found_finger = -1;
for (int i = 0; i < sample_info.TotalSamples; i++)
{
      // ISO19794-2
     err = JSGFPLib.MatchIsoTemplate(m_StoredTemplate,
                              i,
                              m_FetBufM,
                              0,
                              SGFDxSecurityLevel.SL_NORMAL,
                              matched);
```

```
            if (matched)
            {
               found_finger = sample_info.SampleInfo[i].FingerNumber;
               break;
            }
       }
```

3.11. Registration process

To register a fingerprint, a fingerprint image is first captured, and then feature data (minutiae) is extracted from the image into a template. It is recommended to capture at least two image samples per fingerprint for a higher degree of accuracy. The minutiae data from each image can then be compared against each other (i.e. matched) to confirm the quality of the registered fingerprints. This comparison is analogous to a password confirmation routine that is commonly required for entering a new password.

Overview of Registration Process

1. Capture fingerprint images: **JSGFPLib.GetImage()**
2. Extract minutiae from each captured fingerprint image: **JSGFPLib.CreateTemplate()**
3. Match each template to determine if they are acceptable for registration: **JSGFPLib.MatchTemplate()**
4. Save templates to file or database to complete registration

Example: Using two fingerprint images to register one fingerprint

```
err = JSGFPLib.GetMaxTemplateSize(m_MaxTemplateSize);
byte[] m_RegTemplate1 = new byte [MaxTemplateSize[0]];
BYTE*   m_RegTemplate2 = new byte [MaxTemplateSize[0]];

// Get first fingerprint image and create template from the image
err = JSGFPLib.GetImage(m_ImgBuf);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_RegTemplate1);

// Get second fingerprint image and create template from the image
err = JSGFPLib.GetImage(m_ImgBuf);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_RegTemplate2);

DWORD sl = SGFDxSecurityLevel.SL_NORMAL; // Set security level as NORMAL
Boolean[] matched = new Boolean[1];
err = JSGFPLib.MatchTemplate(m_RegTemplate1, m_RegTemplate2, sl, matched);

if (matched)
 // Save these templates somewhere
```

3.12. Verification Process

The verification process involves matching newly input minutiae data against registered minutiae data. Similar to the registration process, verification requires the capture of a fingerprint image followed by extraction of the minutiae data from the captured image into a template.

Overview of Verification Process

1. Capture fingerprint image: **JSGFPLib.GetImage()**
2. Extract minutiae data from captured image: **JSGFPLib.CreateTemplate()**
3. Match newly made template against registered templates: **JSGFPLib.MatchTemplate()**

   - Adjust the security level according to the type of application. For example, if fingerprint-only authentication is used, set the security level higher than **SL_NORMAL** to reduce false acceptance (FAR).

Example: Input minutiae data is matched against two registered minutiae data samples

```
DWORD err;
err = JSGFPLib.GetMaxTemplateSize(m_hFPM, &m_MaxTemplateSize);
byte[] m_ VrfTemplate1= new byte[m_MaxTemplateSize];

// Get first fingerprint image and create template from the image
err = JSGFPLib.GetImage(m_ImgBuf);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_ VrfTemplate1);

DWORD sl = SGFDxSecurityLevel.SL_NORMAL; // Set security level depending on applications.
boolean[] matched1 = new boolean[1];
boolean[] matched2 = new boolean[1];
err = JSGFPLib.MatchTemplate(m_RegTemplate1, m_ VrfTemplate1, sl, matched1);
err = JSGFPLib.MatchTemplate(m_RegTemplate2, m_ VrfTemplate1, sl, matched2);

if (err == SGFDxErrorCode.SGSGFDX_ERROR_NONE)
{
   if (matched1 && matched2)
        // Matched
   else
        // Not matched
}
```

3.13. Getting Matching Score

For improved quality control during the registration or verification process, a matching score can be used instead of a security level setting to determine the success of the operation. The matching score can be specified so that only sets of minutiae data that exceed the score will be accepted; data below the score will be rejected. The matching score may have a value from 0 to 199. **JSGFPLib.GetMatchingScore()** requires two sets of minutiae data of the same template format. **JSGFPLib.GetMatchingScoreEx()** requires two sets of minutiae data, but they can take different template formats.

```
int[] score = new int[1];
if (JSGFPLib.GetMatchingScore(m_RegTemplate1, m_RegTemplate2, score) ==
SGFDXErrorCode.SGFDX_ERROR_NONE)
{
    if (score > 100)
     // Enroll these fingerprints to database
    else
     // Try again
}
```

To understand how the matching scores correlate with typical security levels, refer to the chart below.

Security Level vs. Corresponding Matching Score

Constant	Value	Corresponding Matching Score
SL_NONE	0	0
SL_LOWEST	1	30
SL_LOWER	2	50
SL_LOW	3	60
SL_BELOW_NORMAL	4	70
SL_NORMAL	5	80
SL_ABOVE_NORMAL	6	90
SL_HIGH	7	100
SL_HIGHER	8	120
SL_HIGHEST	9	140

3.14. Template Format

The FDx SDK *Pro* supports three types of fingerprint template formats:

- SecuGen's proprietary template format (**"SG400"**)
- ANSI INCITS 378-2004 "Finger Minutiae Format for Data Exchange" (**"ANSI378"**)
- ISO/IEC 19794-2:2005 "Biometric Data Interchange Formats-- Finger Minutiae Data" (**"ISO19794-2"**)

As default, JSGFPLib creates SecuGen proprietary templates (TEMPLATE_FORMAT_SG400). To change the template format, use **JSGFPLib.SetTemplateFormat()**.

SG400 templates are encrypted for high security and have a size of 400 bytes. ANSI378 templates are not encrypted, and their size is variable depending on how many fingers are registered in the structure and how many minutiae points are found.

For more information about the ANSI378 template, refer to the standard document titled "Information technology - Finger Minutiae Format for Data Interchange," document number ANSI INCITS 378-2004, available at the ANSI website http://webstore.ansi.org.

For more information about the ISO19794-2 template, refer to the standard document titled "Information technology -- Biometric Data Interchange Formats -- Part 2: Finger Minutiae Data," document number ISO/IEC 19794-2:2005, available at the ISO website under Subcommittee JTC 1 / SC 37 (Biometrics):
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38746.

Once the template format is set, it will affect the execution of the JSGFPLib module.

The following APIs are affected by **JSGFPLib.SetTemplateFormat()**:

- **JSGFPLib.GetMaxTemplateSize()**
- **JSGFPLib.CreateTemplate()**
- **JSGFPLib.GetTemplateSize()**
- **JSGFPLib.MatchTemplate()**
- **JSGFPLib.GetMatchingScore()**

The following APIs work only when the template format is **TEMPLATE_FORMAT_ANSI378**:

- **JSGFPLib.GetTemplateSizeAfterMerge()**
- **JSGFPLib.MergeAnsiTemplate()**
- **JSGFPLib.GetAnsiTemplateInfo()**
- **JSGFPLib.MatchAnsiTemplate()**
- **JSGFPLib.GetAnsiMatchingScore()**

The following APIs work only when the template format is **TEMPLATE_FORMAT_ISO19794**:

- **JSGFPLib.GetIsoTemplateSizeAfterMerge()**
- **JSGFPLib.MergeIsoTemplate()**
- **JSGFPLib.GetIsoTemplateInfo()**
- **JSGFPLib.MatchIsoTemplate()**
- **JSGFPLib.GetIsoMatchingScore()**

The following APIs work with any template format:

- **JSGFPLib.MatchTemplateEx()**
- **JSGFPLib.GetMatchingScoreEx()**

- **Setting template format to ANSI378**
  ```
  JSGFPLib.SetTemplateFormat(SGFDxTemplateFormat TEMPLATE_FORMAT_ANSI378);
  ```

- **Setting template format to SG400**
  ```
  JSGFPLib.SetTemplateFormat(SGFDxTemplateFormat TEMPLATE_FORMAT_SG400);
  ```

- **Setting template format to ISO19794**
  ```
  JSGFPLib.SetTemplateFormat(SGFDxTemplateFormat TEMPLATE_FORMAT_ISO19794);
  ```

3.15. Manipulating ANSI378 Templates

The ANSI378 template format allows multiple fingers and multiple views per finger to be stored in one template. To support this feature, FDx SDK *Pro* provides the following special APIs:

- **JSGFPLib.GetTemplateSizeAfterMerge()**
- **JSGFPLib.MergeAnsiTemplate()**
- **JSGFPLib.GetAnsiTemplateInfo()**
- **JSGFPLib.MatchAnsiTemplate()**
- **JSGFPLib.GetAnsiMatchingScore()**

- **Merging two ANSI378 templates**
  After creating an ANSI378 template from a fingerprint image, additional ANSI378 templates can be merged into one template. To do this, use **JSGFPLib.MergeAnsiTemplate()**, which takes two ANSI378 templates and merges them into one template. The merged template size will be less than the sum of the sizes of all input templates. Call **JSGFPLib.GetTemplateSizeAfterMerge()** to obtain the exact template size of the merged template before using **JSGFPLib.MergeAnsiTemplate()**.

```
err = JSGFPLib.GetMaxTemplateSize(m_hFPM, &m_MaxTemplateSize);
byte[] m_Template1 = new byte[m_MaxTemplateSize];
byte[] m_Template2 = new byte[m_MaxTemplateSize];

// Get first fingerprint image and create template from the image
err = JSGFPLib.GetImage(m_ImgBuf);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_Template1);

// Get second fingerprint image and create template from the image
err = JSGFPLib.GetImage(m_ImgBuf);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_Template2);

// Save template after merging two templates - m_Template1, m_Template2
int[]  buf_size = new int[1];
err = JSGFPLib.GetTemplateSizeAfterMerge(m_Template1, m_Template2, buf_size);
byte[]  merged_template = new byte[buf_size[0]];
err = JSGFPLib.MergeAnsiTemplate(m_Template1, m_Template2, merged_template);
```

- **Getting information about an ANSI378 template**
  The ANSI378 template format allows multiple fingers and multiple views per finger to be stored in one template. To match one sample (view) against a sample in other template, information about the template may be needed. To get sample information about a template, use **JSGFPLib.GetAnsiTemplateInfo()**.

```
long err;
int matched_samples = 0;
```

```
SGANSITemplateInfo sample_info1 = new SGANSITemplateInfo;
SGANSITemplateInfo sample_info2 = new SGANSITemplateInfo;
err = JSGFPLib.GetAnsiTemplateInfo(g_EnrollData, sample_info1);
err = JSGFPLib.GetAnsiTemplateInfo(g_VrfData, sample_info2);

for (int i = 0; i < sample_info1.TotalSamples; i++)
{
   for (int j = 0; j < sample_info2.TotalSamples; j++)
   {
      boolean[] matched = new Boolean[1];
      err = JSGFPLib.MatchAnsiTemplate(g_EnrollData, i, g_VrfData, 0, sl, matched);
      if (matched[0])
            matched_samples++;
   }
}

if (err == SGFDxErrorCode.SGFDX_ERROR_NONE)
{
   if (matched_samples > 0)
      System.out.writeln("Found ‟ + matched_samples + ‟matched samples");
   else
      System.out.writeln("Cannot find matching sample");
}
else
   System.out.writeln("MatchTemplate() failed. Error = ‟ + err);
```

3.16. Manipulating ISO19794-2 Templates

The ISO19794-2 template format allows multiple fingers and multiple views per finger to be stored in one template. To support this feature, FDx SDK *Pro* provides the following special APIs:

- **JSGFPLib.GetIsoTemplateSizeAfterMerge()**
- **JSGFPLib.MergeIsoTemplate()**
- **JSGFPLib.GetIsoTemplateInfo()**
- **JSGFPLib.MatchIsoTemplate()**
- **JSGFPLib.GetIsoMatchingScore()**

- **Merging two ISO19794-2 templates**
  After creating an ISO19794-2 template from a fingerprint image, additional ISO19794-2 templates can be merged into one template. To do this, use **JSGFPLib.MergeIsoTemplate()**, which takes two ISO19794-2 templates and merges them into one template. The merged template size will be less than the sum of the sizes of all input templates. Call **JSGFPLib.GetIsoTemplateSizeAfterMerge()** to obtain the exact template size of the merged template before using **JSGFPLib.MergeIsoTemplate()**.

```
err = JSGFPLib.GetMaxTemplateSize(m_hFPM, &m_MaxTemplateSize);
byte[] m_Template1 = new byte[m_MaxTemplateSize];
byte[] m_Template2 = new byte[m_MaxTemplateSize];

// Get first fingerprint image and create template from the image
err = JSGFPLib.GetImage(m_ImgBuf);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_Template1);
```

```
// Get second fingerprint image and create template from the image
err = JSGFPLib.GetImage(m_ImgBuf);
err = JSGFPLib.CreateTemplate(null, m_ImgBuf, m_Template2);

// Save template after merging two templates - m_Template1, m_Template2
int[]  buf_size = new int[1];
err = JSGFPLib.GetTemplateSizeAfterMerge(m_Template1, m_Template2, buf_size);
byte[]  merged_template = new byte[buf_size[0]];
err = JSGFPLib.MergeIsoTemplate(m_Template1, m_Template2, merged_template);
```

- **Getting information about an ISO19794-2 template**
  The ISO19794-2 template format allows multiple fingers and multiple views per finger to be stored in one template. To match one sample (view) against a sample in other template, information about the template may be needed. To get sample information about a template, use **JSGFPLib.GetIsoTemplateInfo()**.

```
DWORD err;
BOOL matched = FALSE;

// ISO19794-2
SGISOTemplateInfo sample_info = {0};
err = JSGFPLib.GetIsoTemplateInfo(m_hFPM, m_StoredTemplate, &sample_info);

matched = FALSE;
int found_finger = -1;
for (int i = 0; i < sample_info.TotalSamples; i++)
{
    // ISO19794-2
    err = JSGFPLib.MatchIsoTemplate(m_hFPM, m_StoredTemplate, i, m_FetBufM, 0, SL_NORMAL,
                                 &matched);
     if (matched)
      {
          found_finger = sample_info.SampleInfo[i].FingerNumber;
          break;
      }
}

if (err == SGFDX_ERROR_NONE)
{
   if (found_finger >= 0)
     m_ResultEdit.Format("The fingerprint data found. Finger Position: %s",
                         g_FingerPosStr[found_finger]);
   else
     m_ResultEdit.Format("Cannot find matched fingerprint data");
}
else
{
   m_ResultEdit.Format("MatchIsoTemplate() failed. Error = %d ", err);
}
```

3.17. Getting Version Information of MINEX Compliant Algorithms

To obtain version information about the MINEX Compliant algorithms, use **JSGFPLib.GetMinexVersion()**. Currently, the extractor version number is 0x000A0035, and the matcher version number is 0x000A8035.

```
Long[] extractor = new long[1];
Long[]matcher = new long[1];
err = JSGFPLib.GetMinexVersion(extractor, matcher);

System.out.println("(Extractor:” +  extractor [0] + “Matcher:” + matcher);
```

Chapter 4. JSGFPLib Function Reference

4.1. JSGFPLib Creation and Termination

public JSGFPLib(android.hardware.usb.UsbManager manager)
Instantiates the JSGFPLib object.

- **Parameters**
  manager
  A fully initialized Android USB manager object. USB services must be initialized before the JSGFPLib object is instantiated.
- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_CREATION_FAILED = Failed to instantiate object

public long Open()
Opens the SecuGen native library.

- **Return values**
  SGFDX_ERROR_NONE = No error

public long Close()
Closes the SecuGen native library.

- **Return values**
  SGFDX_ERROR_NONE = No error

4.2. Initialization

public long Init(long devName)
Initializes JSGFPLib with device name information. The JSGFPLib object loads appropriate drivers with device name (devName) and initializes fingerprint algorithm module based on the device information.

- **Parameters**
  devName
  Specifies the device name
  SG_DEV_FDU03: device name for USB SDU03-based readers
  SG_DEV_FDU04: device name for USB SDU04-based readers
  SG_DEV_FDU05: device name for USB U-20-based readers
  SG_DEV_AUTO: automatically determines the device name

- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_CREATION_FAILED = Failed to create JSGFPLib object
  SGFDX_ERROR_INVALID_PARAM = Invalid parameter used
  SGFDX_ERROR_DRVLOAD_FAILED = Failed to load driver

public long SetTemplateFormat(short format)
Sets template format. Default format is SecuGen proprietary format (TEMPLATE_FORMAT_SG400).

- **Parameters**

format
Specifies template format
TEMPLATE_FORMAT_ANSI378: ANSI INCITS 378-2004 format
TEMPLATE_FORMAT_ISO19794: ISO/IEC 19794-2:2005 format
TEMPLATE_FORMAT_SG400: SecuGen proprietary format

- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_CREATION_FAILED = Failed to create JSGFPLib object
  SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template format

4.3. Device and Capturing Functions

public long OpenDevice(long devId)
Initializes the fingerprint reader.

- **Parameters**
  devId
  Specifies the device ID for USB readers. The value can be from 0 to 9. The maximum number of supported readers attached at the same time is 10.
- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_INVALID_PARAM = Invalid parameter used
  SGFDX_ERROR_SYSLOAD_FAILED = Failed to loading system files
  SGFDX_ERROR_INITIALIZE_FAILED = Failed to initialize chip
  SGFDX_ERROR_DEVICE_NOT_FOUND = Device not found

public long CloseDevice()
Closes the opened device. **OpenDevice()** must be called before this function is used.

- **Parameters**
- **Return values**
  SGFDX_ERROR_NONE = No error

public long GetDeviceInfo(SGDeviceInfoParam Info)
Gets device information from the driver (before device initialization)

- **Parameters**
  info
  An instantiated SGDeviceInfoParam object.
- **Return values**
  SGFDX_ERROR_NONE = No error

public long SetBrightness(int brightness)
Controls brightness of image sensor. This function will only work if Smart Capture is disabled.

- **Parameters**
  brightness
  Must be set to a value from 0 to 100
- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_INVALID_PARAM = Invalid parameter used

public long WriteData(byte index, byte value)

Write key value pairs to control SDK functionality.

- **Parameters**
  index
  Key being changed.
  value
  New value for selected key.

- **Allowed values**
  index=5,data=0 - Disable Smart Capture
  index=5,data=1 - Enable Smart Capture

- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_INVALID_PARAM = Invalid parameter used

public long SetLedOn(boolean on)
Turns optic unit LED on/off.

- **Parameters**
  on
  True: Turns on LED
  False: Turns off LED
- **Return values**
  SGFDX_ERROR_NONE = No error

public long GetImage(byte[] buffer)
Captures a 256 gray-level fingerprint image from the reader. The image size can be retrieved by calling **GetDeviceInfo()**. **JSGFPLib.GetImage()** does not check for image quality. To get image quality of a captured image, use **GetImageQuality()**.

- **Parameters**
  pFPM
  The handle of the JSGFPLib object
  buffer
  A pointer to the buffer containing a fingerprint image. The image size can be retrieved by calling **GetDeviceInfo()**.

- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_WRONG_IMAGE = Capture image is not a real fingerprint image
  SGFDX_ERROR_INVALID_PARAM = Invalid parameter used
  SGFDX_ERROR_LINE_DROPPED = Image data lost

public long GetImageEx(byte[] buffer, long timeout, long quality)
Captures fingerprint images from the reader until the quality of the image is greater than the value of the quality parameter. The captured fingerprint is a 256 gray-level image; image size can be retrieved by calling the **SGFPM_GetDeviceInfo()** function. A quality value of 50 or higher is recommended for registration. A quality value of 40 or higher is recommended for verification.

Note: The returned quality value is different from the value used in **SGFPM_GetImage().** The quality value in **GetImageEx()** represents only the ratio of the fingerprint image area to the whole scanned area.
- **Parameters**
  buffer:
  A byte array containing a fingerprint image. The image size can be retrieved by calling **GetDeviceInfo().**
  timeout:

The timeout value (in milliseconds) used to specify the amount of time the function will wait for a valid fingerprint to be input on the fingerprint reader

quality:

The minimum quality value of an image, used to determine whether to accept the captured image (0 – 100)

- **Return values**

  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_INVALID_PARAM = Invalid parameter used
  SGFDX_ERROR_LINE_DROPPED = Image data lost
  SGFDX_ERROR_TIME_OUT = No valid fingerprint captured in the given time

public long GetImageQuality(long width, long height, byte[] imgBuf, int[] quality)

Gets the quality of a captured (scanned) image. The value is determined by two factors. One is the ratio of the fingerprint image area to the whole scanned area, and the other is the ridge quality of the fingerprint image area. A quality value of 50 or higher is recommended for registration. A quality value of 40 or higher is recommended for verification.

- **Parameters**

  width

  Image width in pixels

  height

  Image height in pixels

  imgBuf

  Fingerprint image data

  quality

  The single element array to contain image quality
- **Return values**

  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_INVALID_PARAM = Invalid parameter used

public long ComputeNFIQ(byte[] imgBuf, long width, long height)

Compute NIST Fingerprint Image Quality score for an 8 bit grayscale fingerprint image.

- **Parameters**

  imgBuf

  Fingerprint image data

  width

  Image width in pixels

  height

  Image height in pixels
- **Return values**

  NFIQ score for the image that was processed
  1 = highest quality fingerprint image
  2 = high quality fingerprint
  3 = medium quality fingerprint image
  4 = low quality fingerprint ima
  5 = lowest quality fingerprint image
  -1 = An error occurred

public long ComputeNFIQEx(byte[] imgBuf, long width, long height, long dpi)

Compute NIST Fingerprint Image Quality score for an 8 bit grayscale fingerprint image.

- **Parameters**

  imgBuf

  Fingerprint image data

  width

Image width in pixels

height

Image height in pixels

dpi

Image resolution in dots (pixels) per inch

- **Return values**

  NFIQ score for the image that was processed
  1 = highest quality fingerprint image
  2 = high quality fingerprint
  3 = medium quality fingerprint image
  4 = low quality fingerprint ima
  5 = lowest quality fingerprint image
  -1 = An error occurred

4.4. Extraction Functions

public long GetMaxTemplateSize(int[] size)

Gets the maximum size of a fingerprint template (view or sample). Use this function before using **CreateTemplate()** to obtain an appropriate buffer size. If the template format is SG400, it returns fixed length size 400.

Note: The returned template size means the maximum size of one view or sample.

- **Parameters**

  size

  The single element array to contain template size

- **Return values**

  SGFDX_ERROR_NONE = No error

public long CreateTemplate(SGFingerInfo fpInfo, byte[] rawImage, byte[] minTemplate)

Extracts minutiae from a fingerprint image to form a template having the default format.

- **Parameters**

  fpInfo

  Fingerprint information stored in a template. For **ANSI378** templates, this information can be retrieved from the template using **GetAnsiTemplateInfo()**. For **ISO19794** templates, this information can be retrieved from the template using **GetIsoTemplateInfo()**. For **SG400** templates, this information cannot be seen in the template.

  rawImg

  A byte array containing 256 Gray-level fingerprint image data

  minTemplate

  A byte array containing minutiae data extracted from a fingerprint image

- **Return values**

  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_FEAT_NUMBER = Inadequate number of minutia
  SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
  SGFDX_ERROR_INVALID_TEMPLATE1 = 103 = Error while decoding template 1
  SGFDX_ERROR_INVALID_TEMPLATE2 = 104 = Error while decoding template 2

public long GetTemplateSize(byte[] minTemplate, int[] size)

Gets template size. If the template format is SG400, it will return 400. If the template format is ANSI378 or ISO19794, template size may vary.

- **Parameters**
  minTemplate
  A byte array containing minutiae data extracted from a fingerprint image
  size
  A byte array that will contain template size
- **Return values**
  SGFDX_ERROR_NONE = No error

4.5. Matching Functions

public long MatchTemplate(byte[] minTemplate1, byte[] minTemplate2, long secuLevel, Boolean[] matched)

Compares two sets of minutiae data of the **same** template format. The template format should be the same as that set by **SetTemplateFormat()** and should include only one sample. To match templates that have more than one sample, use **MatchTemplateEx()** or **MatchAnsiTemplate()**.

It returns TRUE or FALSE as a matching result (**matched**). Security level (**secuLevel**) affects matching result. The security level may be adjusted according to the security policy required by the user or organization.

- **Parameters**
  minTemplate1
  A byte array containing minutiae data extracted from a fingerprint image
  minTempate2
  A byte array containing minutiae data extracted from a fingerprint image
  secuLevel
  A security level as specified in “SGFDxSecurityLevel” by one the following nine security levels: SL_LOWEST, SL_LOWER, SL_LOW, SL_BELOW_NORMAL, SL_NORMAL, SL_ABOVE_NORMAL, SL_HIGH, SL_HIGHER and SL_HIGHEST. SL_NORMAL is recommended in usual case.
  matched
  A byte array that contains matching result. If passed templates are matching templates, **TRUE** is returned. If not, **FALSE** is returned.
- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
  SGFDX_ERROR_INVALID_TEMPLATE1 = Error in minTemplate1
  SGFDX_ERROR_INVALID_TEMPLATE2 = Error in minTemplate2

public long MatchTemplateEx(byte[] minTemplate1, short tempateType1, long sampleNum1, byte[] minTemplate2, short tempateType2, long sampleNum2, long secuLevel, boolean[] matched)

Compares two sets of minutiae data, which can be of different template formats (SG400 or ANSI378). It returns TRUE or FALSE as a matching result (**matched**). Security level (**secuLevel**) affects matching result. The security level may be adjusted according to the security policy required by the user or organization.

- **Parameters**
  minTemplate1
  A byte array containing minutiae data extracted from a fingerprint image
  templateType1
  Specifies format of minTemplate1. Should be either TEMPLATE_FORMAT_SG400 or TEMPLATE_FORMAT_ANSI378.
  sampleNum1
  Position of a sample to be matched in minTemplate1. If templateType1 is TEMPLATE_FORMAT_ANSI378, it can have a value from 0 to (number of samples -1) in minTemplate1. If templateType1 is TEMPLATE_FORMAT_SG400, this value is ignored.

minTemplate2
A byte array containing minutiae data extracted from a fingerprint image
templateType2
Specifies format of minTemplate2. Should be either TEMPLATE_FORMAT_SG400 or TEMPLATE_FORMAT_ANSI378.
sampleNum2
Position of a sample to be matched in minTemplate2. If templateType2 is TEMPLATE_FORMAT_ANSI378, it can have a value from 0 to (number of samples -1) in minTemplate2. If templateType2 is TEMPLATE_FORMAT_SG400, this value is ignored.
secuLevel
A security level as specified in “fplibnew.h” by one the following nine security levels: SL_LOWEST, SL_LOWER, SL_LOW, SL_BELOW_NORMAL, SL_NORMAL, SL_ABOVE_NORMAL, SL_HIGH, SL_HIGHER, and SL_HIGHEST. SL_NORMAL is recommended in usual case.
matched
TRUE: Same template
FALSE: Not same template

- **Return values**
SGFDX_ERROR_NONE = No error
SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
SGFDX_ERROR_INVALID_TEMPLATE1 = Error in minTemplate1
SGFDX_ERROR_INVALID_TEMPLATE2 = Error in minTemplate2

public long JSGFPLib.GetMatchingScore(byte[] minTemplate1, byte[] minTemplate2, int[] score)
Gets matching score of two sets of minutiae data of the **same** template format.

- **Parameters**
minTemplate1
A pointer to the buffer containing minutiae data extracted from a fingerprint image
minTemplate2
A pointer to the buffer containing minutiae data extracted from a fingerprint image
score
Matching score. Returned score has a value from 0 to 199.
- **Returned values**
SGFDX_ERROR_NONE = No error
SGFDX_ERROR_INVALID_TEMPLATE1 = Error in minTemplate1
SGFDX_ERROR_INVALID_TEMPLATE2 = Error in minTemplate2

public long GetMatchingScoreEx(byte[] minTemplate1, short tempateType1, long sampleNum1, byte[] minTemplate2, short tempateType2, long sampleNum2, int[] score);
Gets matching score of two sets of minutiae data, which can be of different template formats (SG400 or ANSI378).

- **Parameters**
minTemplate1
A byte array containing minutiae data extracted from a fingerprint image

templateType1
Specifies format of minTemplate1. Should be either TEMPLATE_FORMAT_SG400 or TEMPLATE_FORMAT_ANSI378.

sampleNum1
Position of a sample to be matched in minTemplate1. If templateType1 is TEMPLATE_FORMAT_ANSI378, it can have a value from 0 to (number of samples -1) in minTemplate1. If templateType1 is TEMPLATE_FORMAT_SG400, this value is ignored.
minTemplate2
A byte array containing minutiae data extracted from a fingerprint image

templateType2
Specifies format of minTemplate2. Should be either TEMPLATE_FORMAT_SG400 or TEMPLATE_FORMAT_ANSI378.
sampleNum2
Position of a sample to be matched in minTemplate2. If templateType2 is TEMPLATE_FORMAT_ANSI378, it can have a value from 0 to (number of samples -1) in minTemplate2. If templateType2 is TEMPLATE_FORMAT_SG400, this value is ignored.
score
Matching score. Returned score has a value from 0 to 199.

- **Returned values**
SGFDX_ERROR_NONE = No error
SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
SGFDX_ERROR_INVALID_TEMPLATE1 = Error in minTemplate1
SGFDX_ERROR_INVALID_TEMPLATE2 = Error in minTemplate2

4.6. Functions for ANSI378 Templates

public long GetTemplateSizeAfterMerge(byte[] ansiTemplate1,byte[] ansiTemplate2, int[] size)
Calculates template size if two templates – ansiTemplate1 and ansiTemplate2 – are merged. Use this function to determine exact buffer size before using **MergeAnsiTemplate()**.

- **Parameters**
ansiTemplate1
A byte array containing minutiae data. A template can have more than one sample.
ansiTempate2
A byte array containing minutiae data. A template can have more than one sample.
size
Template size if two templates are merged
- **Return values**
SGFDX_ERROR_NONE = No error
SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
SGFDX_ERROR_INVALID_TEMPLATE1 = Error in minTemplate1
SGFDX_ERROR_INVALID_TEMPLATE2 = Error in minTemplate2

public long MergeAnsiTemplate(byte[] ansiTemplate1,byte[] ansiTemplate2, byte[] outTemplate)
Merges two ANSI378 templates and returns a new merged template. The merged template (**outTemplate**) size will be less than sum of the sizes of the two input templates (size of ansiTemplate1 + size of ansiTemplate2). Call **GetTemplateSizeAfterMerge**() to determine the exact buffer size for **outTemplate** before calling **MergeAnsiTemplate()**.

- **Parameters**
ansiTemplate1
A byte array containing minutiae data. A template can have more than one sample.
asniTempate2
A byte array containing minutiae data. A template can have more than one sample.
outTempate
The byte array containing merged data. The buffer should be assigned by the application. To determine the exact buffer size, call **JSGFPLib.GetTemplateSizeAfterMerge()**.
- **Return values**
SGFDX_ERROR_NONE = No error
SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
SGFDX_ERROR_INVALID_TEMPLATE1 = Error in minTemplate1
SGFDX_ERROR_INVALID_TEMPLATE2 = Error in minTemplate2

public long GetAnsiTemplateInfo(byte[] ansiTemplate, SGANSITemplateInfo templateInfo)

Gets information of an ANSI378 template. Call this function before **MatchAnsiTemplate()** to obtain information about a template.

- **Parameters**
  anisiTemplate
  ANSI378 template
  templateInfo
  The object that contains template information. For more information see **SGANSITemplateInfo** structure.
- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_INVALID_PARAM = Invalid parameter used
  SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type

public long MatchAnsiTemplate(byte[] ansiTemplate1, long sampleNum1, byte[] ansiTemplate2, long sampleNum2, long secuLevel, Boolean[] matched)

Compares two sets of ANSI378 templates. It returns TRUE or FALSE as a matching result (**matched**). Security level (**secuLevel**) affects matching result. The security level may be adjusted according to the security policy required by the user or organization.

- **Parameters**
  ansiTemplate1
  A byte array containing minutiae data. A template can have more than one sample.
  sampleNum1
  Position of sample to be matched in **ansiTemplate1**. It can be from 0 to (number of samples -1) in **ansiTemplate1**
  ansiTempate2
  A byte array containing minutiae data. A template can have more than one sample.
  sampleNum2
  Position of sample to be matched in **ansiTemplate2**. It can be from 0 to (number of samples -1) in **ansiTemplate2**
  secuLevel
  A security level as specified in **SGFDxSecurityLevel** by one the following nine security levels: SL_LOWEST, SL_LOWER, SL_LOW, SL_BELOW_NORMAL, SL_NORMAL, SL_ABOVE_NORMAL, SL_HIGH, SL_HIGHER and SL_HIGHEST. SL_NORMAL is recommended in usual case.
  matched
  TRUE: Same template
  FALSE: Not same template
- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
  SGFDX_ERROR_INVALID_TEMPLATE1 = Error in ansiTemplate1
  SGFDX_ERROR_INVALID_TEMPLATE2 = Error in ansiTemplate2

public long GetAnsiMatchingScore(byte[] ansiTemplate1, long sampleNum1, byte[] ansiTemplate2, long sampleNum2, int[] score)

Gets matching score.

- **Parameters**
  ansiTemplate1
  A byte array containing minutiae data. A template can have more than one sample.
  sampleNum1
  Position of sample to be matched in **ansiTemplate1**. It can be from 0 to (number of samples -1) in **ansiTemplate1**

ansiTempate2
A byte array containing minutiae data. A template can have more than one sample.
sampleNum2
Position of sample to be matched in **ansiTemplate2**. It can be from 0 to (number of samples -1) in **ansiTemplate2**
score
Matching score. Returned score has a value from 0 to 199.

- **Return values**
SGFDX_ERROR_NONE = No error
SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
SGFDX_ERROR_INVALID_TEMPLATE1 = Error in ansiTemplate1
SGFDX_ERROR_INVALID_TEMPLATE2 = Error in ansiTemplate2

4.7. Functions for ISO19794-2 Templates

public long GetIsoTemplateSizeAfterMerge(byte[] isoTemplate1, byte[] isoTemplate2, int[] size)
Calculates template size if two templates – isoTemplate1 and isoTemplate2 – are merged. Use this function to determine exact buffer size before using **MergeIsoTemplate()**.

- **Parameters**
isoTemplate1
A byte array containing minutiae data. A template can have more than one sample.
isoTempate2
A byte array containing minutiae data. A template can have more than one sample.
size
Template size if two templates are merged
- **Return values**
SGFDX_ERROR_NONE = No error
SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
SGFDX_ERROR_INVALID_TEMPLATE1 = Error in minTemplate1
SGFDX_ERROR_INVALID_TEMPLATE2 = Error in minTemplate2

public long MergeIsoTemplate(byte[] isoTemplate1, byte[] isoTemplate2,byte[] outTemplate)
Merges two ISO19794-2 templates and returns a new merged template. The merged template (**outTemplate**) size will be less than sum of the sizes of the two input templates (size of isoTemplate1 + size of isoTemplate2). Call **GetTIsoemplateSizeAfterMerge**() to determine the exact buffer size for **outTemplate** before calling **MergeIsoTemplate()**.

- **Parameters**
isoTemplate1
A byte array containing minutiae data. A template can have more than one sample.
isoTempate2
A byte array containing minutiae data. A template can have more than one sample.
outTempate
The byte array containing merged data. The buffer should be assigned by the application. To determine the exact buffer size, call **GetIsoTemplateSizeAfterMerge()**.

- **Return values**
SGFDX_ERROR_NONE = No error
SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
SGFDX_ERROR_INVALID_TEMPLATE1 = Error in minTemplate1
SGFDX_ERROR_INVALID_TEMPLATE2 = Error in minTemplate2

public long GetIsoTemplateInfo(byte[] isoTemplate, SGISOTemplateInfo templateInfo)

Gets information of an ISO19794-2 template. Call this function before **MatchIsoTemplate()** to obtain information about a template.

- **Parameters**
  isoTemplate
  ISO19794-2 template
  templateInfo
  The object that contains template information. For more information see **SGISOTemplateInfo** structure.
- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_INVALID_PARAM = Invalid parameter used
  SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type

public long MatchIsoTemplate(byte[] isoTemplate1, long sampleNum1, byte[] isoTemplate2, long sampleNum2, long secuLevel, boolean[] matched)

Compares two sets of ISO19794-2 templates. It returns TRUE or FALSE as a matching result (**matched**). Security level (**secuLevel**) affects matching result. The security level may be adjusted according to the security policy required by the user or organization.

- **Parameters**
  isoTemplate1
  A byte array containing minutiae data. A template can have more than one sample.
  sampleNum1
  Position of sample to be matched in **isoTemplate1**. It can be from 0 to (number of samples -1) in **isoTemplate1**
  isoTempate2
  A byte array containing minutiae data. A template can have more than one sample.
  sampleNum2
  Position of sample to be matched in **isoTemplate2**. It can be from 0 to (number of samples -1) in **isoTemplate2**
  secuLevel
  A security level as specified in **SGFDxSecurityLevel** by one the following nine security levels: SL_LOWEST, SL_LOWER, SL_LOW, SL_BELOW_NORMAL, SL_NORMAL, SL_ABOVE_NORMAL, SL_HIGH, SL_HIGHER and SL_HIGHEST. SL_NORMAL is recommended in usual case.
  matched
  TRUE: Same template
  FALSE: Not same template
- **Return values**
  SGFDX_ERROR_NONE = No error
  SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
  SGFDX_ERROR_INVALID_TEMPLATE1 = Error in isoTemplate1
  SGFDX_ERROR_INVALID_TEMPLATE2 = Error in isoTemplate2

public long GetIsoMatchingScore(byte[] isoTemplate1, long sampleNum1, byte[] isoTemplate2, long sampleNum2, int[] score)

Gets matching score.

- **Parameters**
  isoTemplate1
  A byte array containing minutiae data. A template can have more than one sample.
  sampleNum1
  Position of sample to be matched in **isoTemplate1**. It can be from 0 to (number of samples -1) in **isoTemplate1**

isoTempate2
A byte array containing minutiae data. A template can have more than one sample.
sampleNum2
Position of sample to be matched in **isoTemplate2**. It can be from 0 to (number of samples -1) in **isoTemplate2**
score
Matching score. Returned score has a value from 0 to 199.

- **Return values**
SGFDX_ERROR_NONE = No error
SGFDX_ERROR_INVALID_TEMPLATE_TYPE = Wrong template type
SGFDX_ERROR_INVALID_TEMPLATE1 = Error in isoTemplate1
SGFDX_ERROR_INVALID_TEMPLATE2 = Error in isoTemplate2

4.8. Other

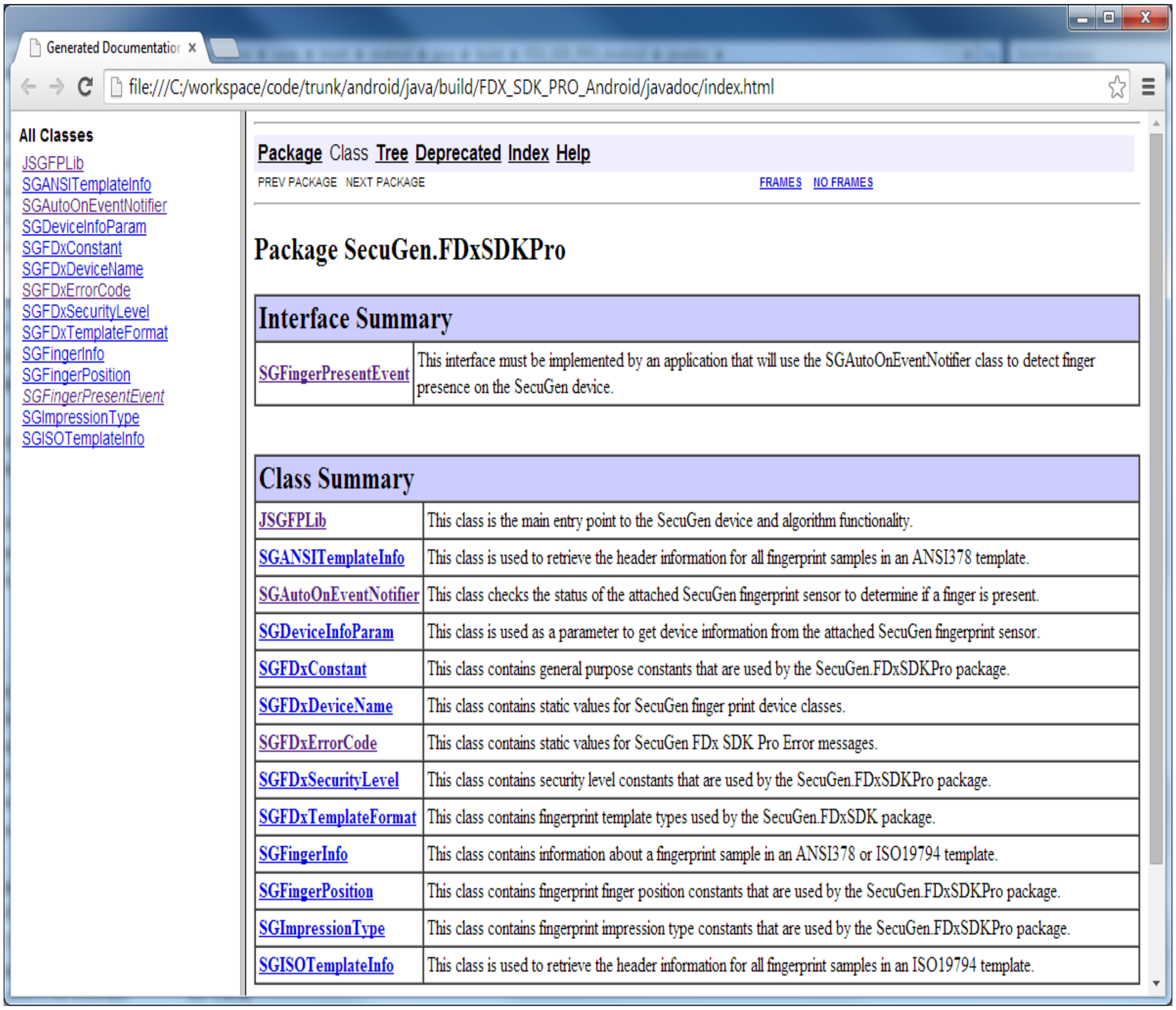
public long GetMinexVersion(long[] extractor, long[] matcher)
Gets version of MINEX Compliant algorithms used in this SDK.

- **Parameters**
extractor
Version of MINEX Compliant extractor (template generator)
matcher
Version of MINEX Compliant matcher (template matcher)
- **Return values**
SGFDX_ERROR_NONE = No error

Chapter 5. Class Reference

5.1. Java Documentation

Refer to the “javadoc” folder in this SDK release for a complete class reference.


Generated Documentation
file:///C:/workspace/code/trunk/android/java/build/FDX_SDK_PRO_Android/javadoc/index.html
Package Class Tree Deprecated Index Help
PREV PACKAGE NEXT PACKAGE
FRAMES NO FRAMES

All Classes

JSGFPLib
SGANSITemplateInfo
SGAutoOnEventNotifier
SGDeviceInfoParam
SGFDxConstant
SGFDxDeviceName
SGFDxErrorCode
SGFDxSecurityLevel
SGFDxTemplateFormat
SGFingerInfo
SGFingerPosition
SGFingerPresentEvent
SGImpressionType
SGISOTemplateInfo

Package SecuGen.FDxSDKPro

Interface Summary	
SGFingerPresentEvent	This interface must be implemented by an application that will use the SGAutoOnEventNotifier class to detect finger presence on the SecuGen device.

Class Summary	
JSGFPLib	This class is the main entry point to the SecuGen device and algorithm functionality.
SGANSITemplateInfo	This class is used to retrieve the header information for all fingerprint samples in an ANSI378 template.
SGAutoOnEventNotifier	This class checks the status of the attached SecuGen fingerprint sensor to determine if a finger is present.
SGDeviceInfoParam	This class is used as a parameter to get device information from the attached SecuGen fingerprint sensor.
SGFDxConstant	This class contains general purpose constants that are used by the SecuGen.FDxSDKPro package.
SGFDxDeviceName	This class contains static values for SecuGen finger print device classes.
SGFDxErrorCode	This class contains static values for SecuGen FDx SDK Pro Error messages.
SGFDxSecurityLevel	This class contains security level constants that are used by the SecuGen.FDxSDKPro package.
SGFDxTemplateFormat	This class contains fingerprint template types used by the SecuGen.FDxSDK package.
SGFingerInfo	This class contains information about a fingerprint sample in an ANSI378 or ISO19794 template.
SGFingerPosition	This class contains fingerprint finger position constants that are used by the SecuGen.FDxSDKPro package.
SGImpressionType	This class contains fingerprint impression type constants that are used by the SecuGen.FDxSDKPro package.
SGISOTemplateInfo	This class is used to retrieve the header information for all fingerprint samples in an ISO19794 template.

Chapter 6. Constants

6.1. SGFDxDeviceName

Device Name	Value	Description
SG_DEV_UNKNOWN	0	Not determined
SG_DEV_FDU03	0x04	SDU03P-based reader
SG_DEV_FDU04	0x05	SDU04P-based reader
SG_DEV_FDU05	0x06	U20-based reader
SG_DEV_FDU06	0x07	UPx-based reader
SG_DEV_FDU07	0x08	U10-based reader
SG_DEV_AUTO	0xFF	Auto Detect

6.2. SGFDxSecurityLevel

Security Level	Value	Description
SL_NONE	0	No Security
SL_LOWEST	1	Lowest
SL_LOWER	2	Lower
SL_LOW	3	Low
SL_BELOW_NORMAL	4	Below normal
SL_NORMAL	5	Normal
SL_ABOVE_NORMAL	6	Above normal
SL_HIGH	7	High
SL_HIGHER	8	Higher
SL_HIGHEST	9	Highest

6.3. SGFDxTemplateFormat

Template Format	Value	Description
TEMPLATE_FORMAT_ANSI378	0x0100	ANSI INCITS 378-2004 format
TEMPLATE_FORMAT_SG400	0x0200	SecuGen proprietary format
TEMPLATE_FORMAT_ISO19794	0x0300	ISO/IEC 19794-2:2005 format

6.4. SGImpressionType

Security Level	Value	Description
SG_IMPTYPE_LP	0x00	Live-scan plain
SG_IMPTYPE_LR	0x01	Live-scan rolled
SG_IMPTYPE_NP	0x02	Non-live-scan plain
SG_IMPTYPE_NR	0x03	Non-live-scan rolled

6.5. SGFingerPosition

Security Level	Value	Description
SG_FINGPOS_UK	0x00	Unknown finger
SG_FINGPOS_RT	0x01	Right thumb
SG_FINGPOS_RI	0x02	Right index finger
SG_FINGPOS_RM	0x03	Right middle finger
SG_FINGPOS_RR	0x04	Right ring finger
SG_FINGPOS_RL	0x05	Right little finger
SG_FINGPOS_LT	0x06	Left thumb
SG_FINGPOS_LI	0x07	Left index finger
SG_FINGPOS_LM	0x08	Left middle finger
SG_FINGPOS_LR	0x09	Left ring finger
SG_FINGPOS_LL	0x0A	Left little finger

6.6. SGFDxErrorCode

Error Code	Value	Description
General Error Codes		
SGFDX_ERROR_NONE	0	No error
SGFDX_ERROR_CREATION_FAILED	1	JSGFPLib object creation failed
SGFDX_ERROR_FUNCTION_FAILED	2	Function call failed
SGFDX_ERROR_INVALID_PARAM	3	Invalid parameter used
SGFDX_ERROR_NOT_USED	4	Not used function
SGFDX_ERROR_DLLLOAD_FAILED	5	DLL loading failed
SGFDX_ERROR_DLLLOAD_FAILED_DRV	6	Device driver loading failed
SGFDX_ERROR_DLLLOAD_FAILED_ALGO	7	Algorithm DLL loading failed
Device Driver Error Codes		
SGFDX_ERROR_SYSLOAD_FAILED	51	Cannot find driver sys file
SGFDX_ERROR_INITIALIZE_FAILED	52	Chip initialization failed
SGFDX_ERROR_LINE_DROPPED	53	Image data lost
SGFDX_ERROR_TIME_OUT	54	GetImageEx() timeout
SGFDX_ERROR_DEVICE_NOT_FOUND	55	Device not found
SGFDX_ERROR_DRVLOAD_FAILED	56	Driver file load failed
SGFDX_ERROR_WRONG_IMAGE	57	Wrong image
SGFDX_ERROR_LACK_OF_BANDWIDTH	58	Lack of USB bandwidth
SGFDX_ERROR_DEV_ALREADY_OPEN	59	Device is already opened
SGFDX_ERROR_GETSN_FAILED	60	Serial number does not exist
SGFDX_ERROR_UNSUPPORTED_DEV	61	Unsupported device

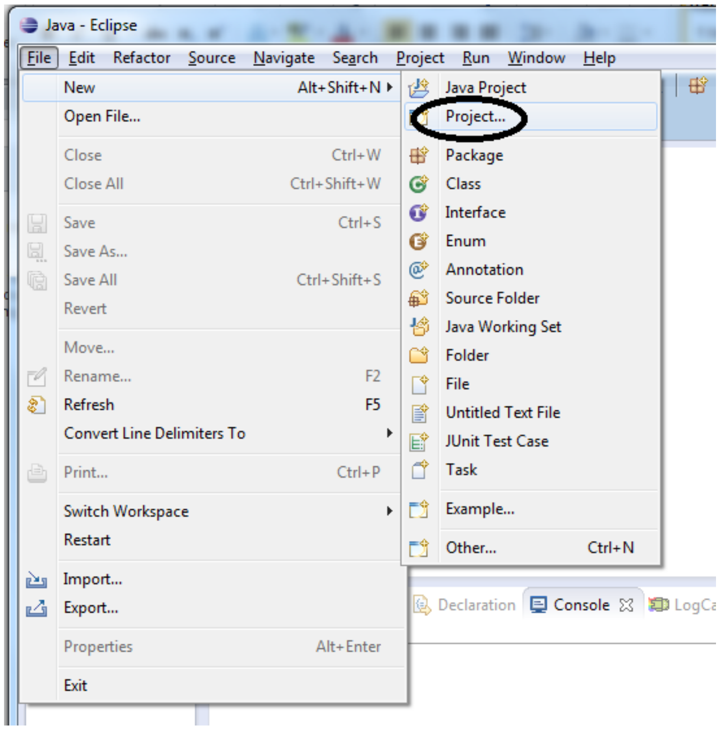
Extract & Matching Error Codes		
SGFDX_ERROR_FEAT_NUMBER	101	Inadequate number of minutiae
SGFDX_ERROR_INVALID_TEMPLATE_TYPE	102	Wrong template type
SGFDX_ERROR_INVALID_TEMPLATE1	103	Error in decoding template 1
SGFDX_ERROR_INVALID_TEMPLATE2	104	Error in decoding template 2
SGFDX_ERROR_EXTRACT_FAIL	105	Extraction failed
SGFDX_ERROR_MATCH_FAIL	106	Matching failed

6.7. SGFDxConstant

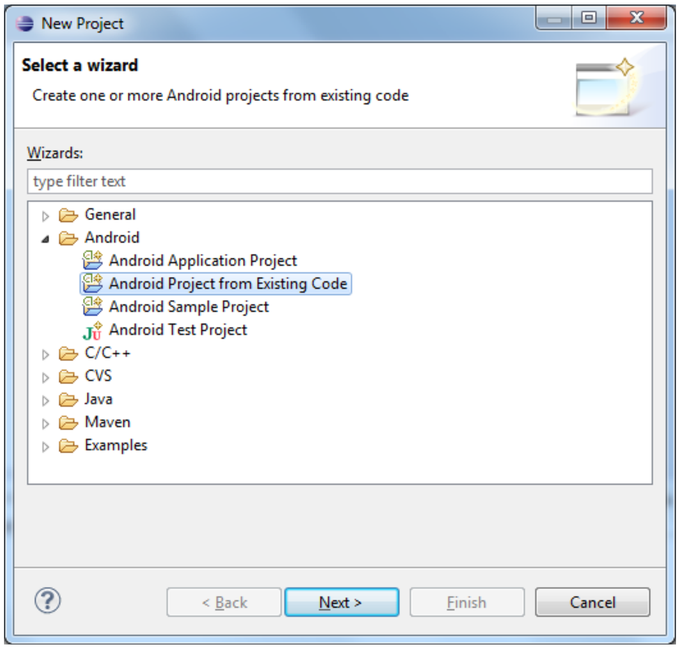
- DEV_SN_LEN 15 // Device serial number length

Appendix A. Importing the Sample App

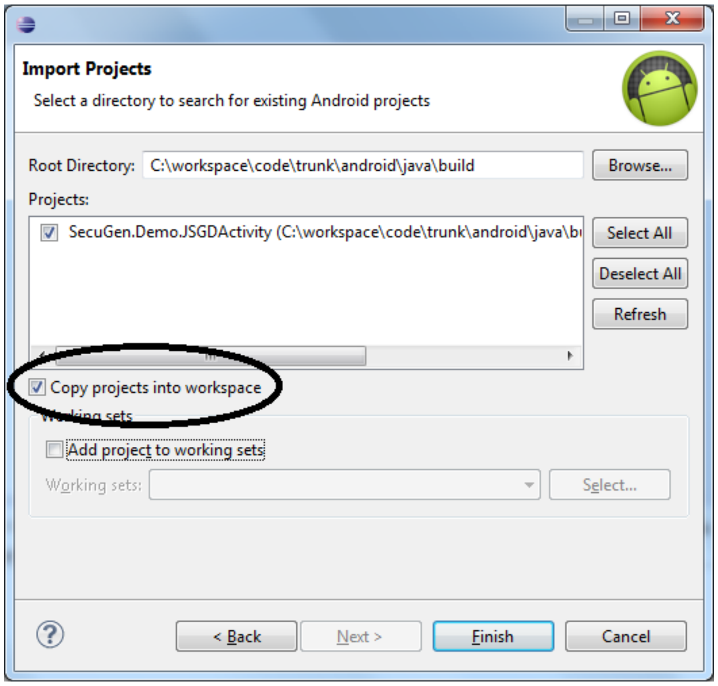
1. Select File->New->Project.



2. Select the “Android Project from Existing Code” wizard.



3. Navigate to the root directory containing the “SecGenUSBDist” sub-directory and select the SecuGen demo application. Check “Copy projects into workspace.”



Import Projects

Select a directory to search for existing Android projects

Root Directory: C:\workspace\code\trunk\android\java\build

Projects:

☑ SecuGen.Demo.JSGDActivity (C:\workspace\code\trunk\android\java\bu

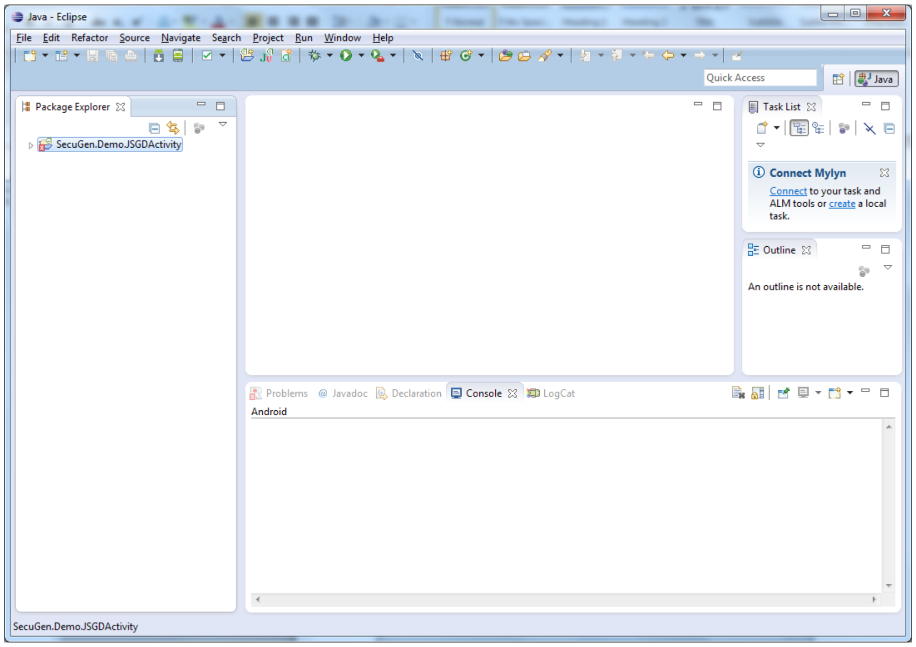
☑ Copy projects into workspace

Working sets

☐ Add project to working sets

Working sets:

4. The project will then be copied to your workspace and be visible in the Package Explorer.



Java - Eclipse

Package Explorer

SecuGen.Demo.JSGDActivity

Task List

Connect Mylyn

Connect to your task and ALM tools or create a local task.

Outline

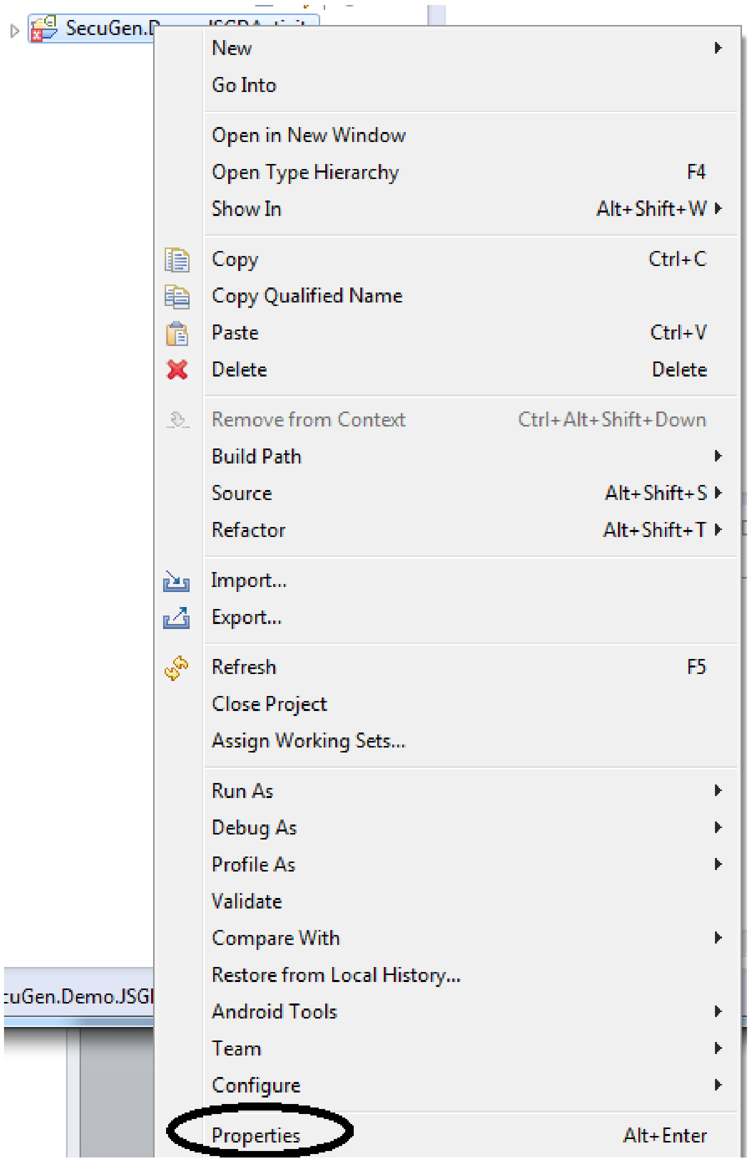
An outline is not available.

Console

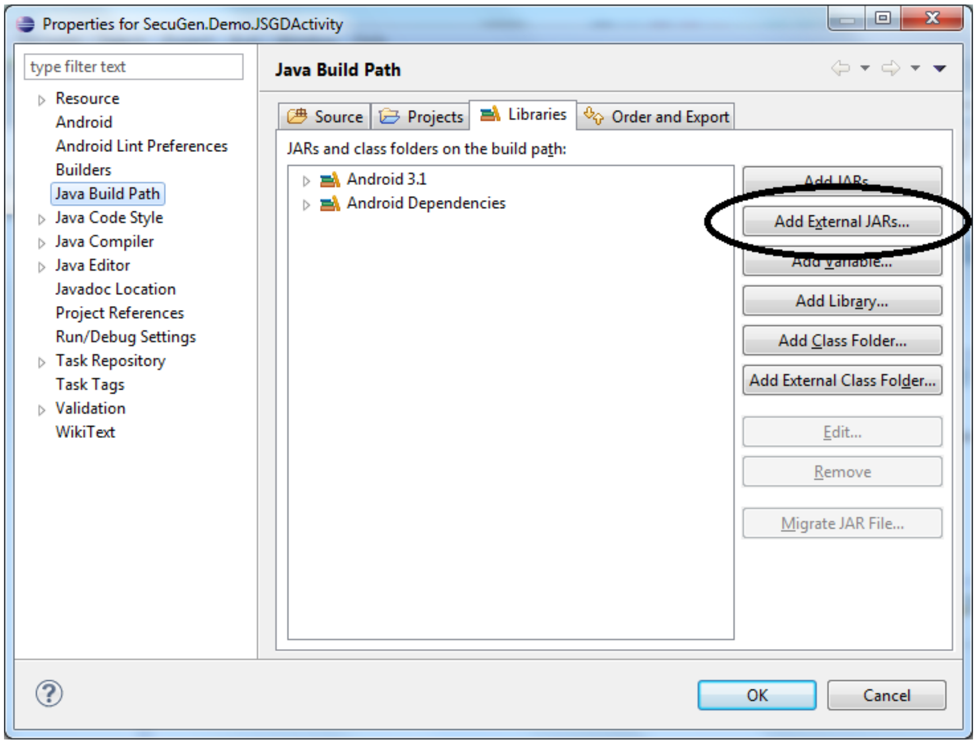
Android

SecuGen.Demo.JSGDActivity

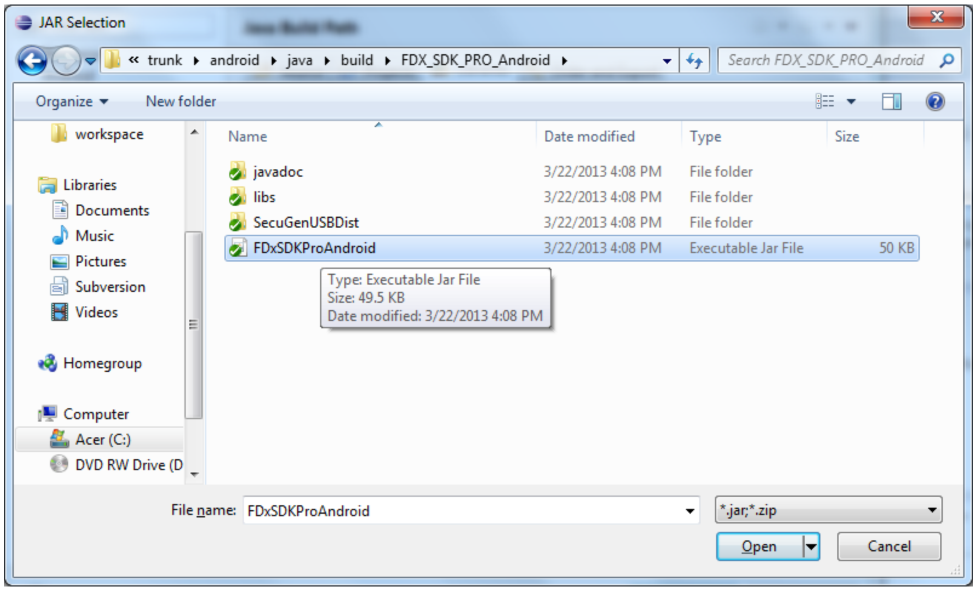
5. A red “X” will be displayed indicating that there are issues that need to be resolved in order to build the project. Right-click the project and select Properties.



6. Select Java Build Path and click “Add External Jars.”

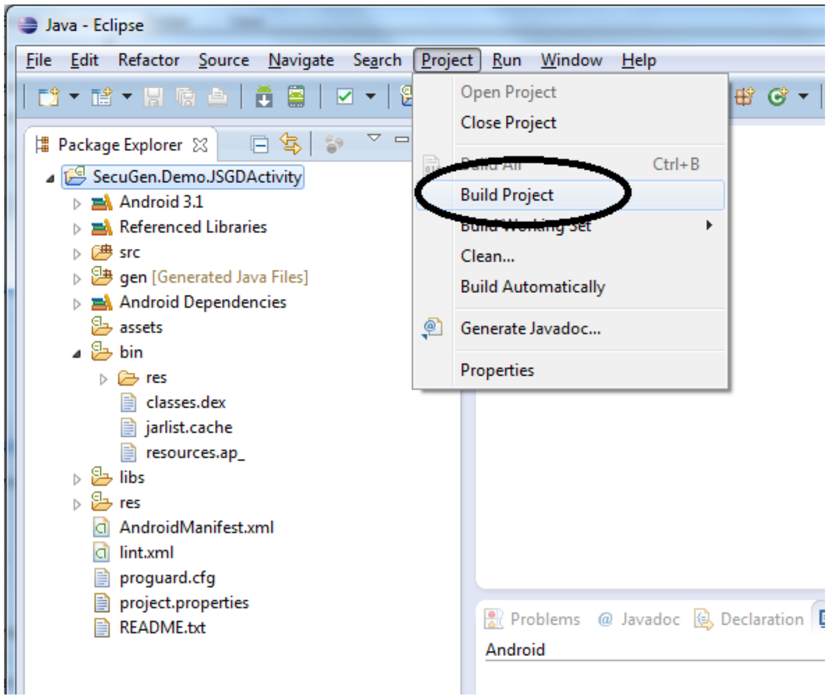


7. Select FDxSDKProAndroid.jar and click “Open”

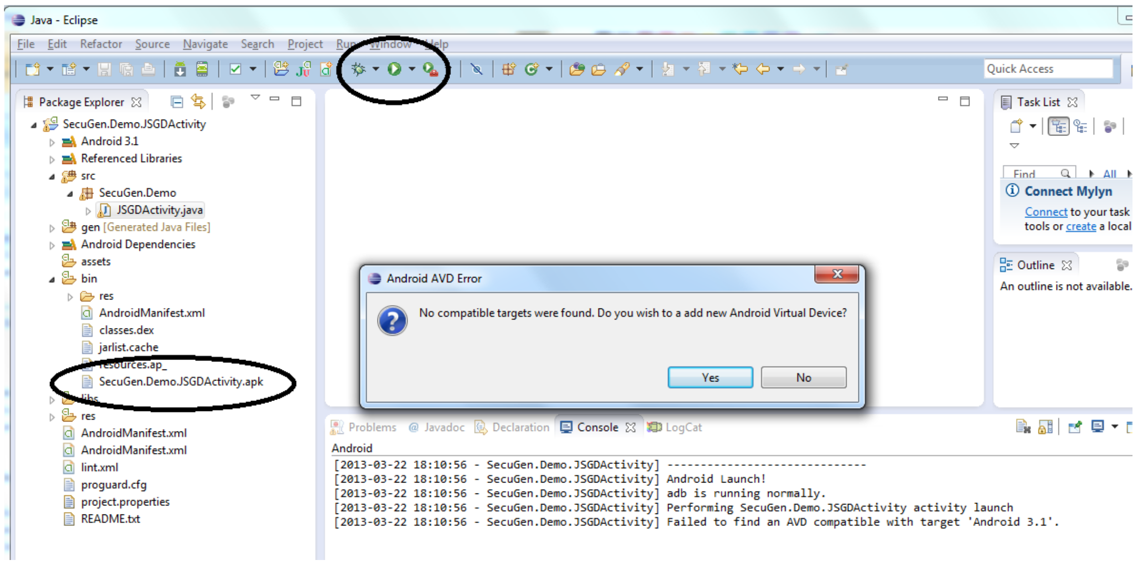

JAR Selection
« trunk ▸ android ▸ java ▸ build ▸ FDX_SDK_PRO_Android ▸
Search FDX_SDK_PRO_Android
Organize
New folder
workspace
Libraries
Documents
Music
Pictures
Subversion
Videos
Homegroup
Computer
Acer (C:)
DVD RW Drive (D
Type: Executable Jar File
Size: 49.5 KB
Date modified: 3/22/2013 4:08 PM
File name: FDxSDKProAndroid
.jar;.zip
Open
Cancel

Name	Date modified	Type	Size
javadoc	3/22/2013 4:08 PM	File folder	
libs	3/22/2013 4:08 PM	File folder	
SecuGenUSBDist	3/22/2013 4:08 PM	File folder	
FDxSDKProAndroid	3/22/2013 4:08 PM	Executable Jar File	50 KB

8. Select SecuGen.Demo.JSGDActivity in the Package Explorer and then select Project->Build Project.



9. Click the green Run button. This will generate the distributable Android packaged application (SecuGen.Demo.JSGDActivity.apk" and install the application on your selected Android device.



No compatible targets were found. Do you wish to a add new Android Virtual Device?

```
[2013-03-22 18:10:56 - SecuGen.Demo.JSGDActivity] ------------------------------
[2013-03-22 18:10:56 - SecuGen.Demo.JSGDActivity] Android Launch!
[2013-03-22 18:10:56 - SecuGen.Demo.JSGDActivity] adb is running normally.
[2013-03-22 18:10:56 - SecuGen.Demo.JSGDActivity] Performing SecuGen.Demo.JSGDActivity activity launch
[2013-03-22 18:10:56 - SecuGen.Demo.JSGDActivity] Failed to find an AVD compatible with target 'Android 3.1'.
```

Appendix B. Using Auto-On

1. Auto-On is supported from SDK v1.3 and onwards. Using this technology, an ***SGAutoOnEventNotifier*** object can be instantiated that checks the attached SecuGen fingerprint device for the presence of a finger. When a finger is detected, the ***SGAutoEventNotifier*** object calls the ***SGFingerPresentCallback()*** method implemented by the calling application.

2. To use AUTO ON, the host application must implement the **SGFingerPresentEvent** interface. The **SGFingerPresentCallback()** method must be implemented.

3. It is good practice to stop the **SGAutoOnEventNotifier** thread when the **SGFingerPresentCallback()** method is called.

Example:

```
public class JSGDActivity extends Activity
        implements View.OnClickListener, java.lang.Runnable, SGFingerPresentEvent {
        …
        …
        private SGAutoOnEventNotifier autoOn;
        private boolean mAutoOnEnabled;
        private JSGFPLib sgfplib;
        …
        …
    //This message handler is used to access local resources not
    //accessible by SGFingerPresentCallback() because it is called by
    //a separate thread.
    public Handler fingerDetectedHandler = new Handler(){
        // @Override
            public void handleMessage(Message msg) {
               //Handle the message
               CaptureFingerPrint();
            }
    };

    public void onCreate(Bundle savedInstanceState) {
                …
                …
        //USB Permissions
        mPermissionIntent = PendingIntent.getBroadcast(this, 0,
                            new  Intent(ACTION_USB_PERMISSION), 0);
        filter = new IntentFilter(ACTION_USB_PERMISSION);
        registerReceiver(mUsbReceiver, filter);
        sgfplib = new JSGFPLib((UsbManager)getSystemService(Context.USB_SERVICE));
        autoOn = new SGAutoOnEventNotifier (sgfplib, this);
        autoOn.start();
                …
                …
    }
        …
        …

    public void SGFingerPresentCallback (){
                autoOn.stop();
                fingerDetectedHandler.sendMessage(new Message());
    }

    public void CaptureFingerPrint(){
            byte[] buffer = new byte[mImageWidth*mImageHeight];
            long result = sgfplib.GetImage(buffer);

    }
}
```