# SecuGen

# Programming Manual for

# *FDx SDK Pro* for Java

For applications using SecuGen® fingerprint modules

## About SecuGen

SecuGen (www.secugen.com) provides biometric products and development tools for development organizations that are creating physical and network security systems 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 developing 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 are renowned for their accuracy, reliability, ruggedness, and affordability. Based in Silicon Valley, SecuGen has been serving the global biometric community since 1998.

## *SecuGen Sensor Qualities*

- Excellent Image Quality: Clear, distortion-free fingerprint images are generated using advanced, patented 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 SecuGen sensors ideal for a broad range of applications.

- Low Cost: Products are developed to deliver high performance, with zero maintenance at very affordable prices for general and industrial use.

## *Advantages of SecuGen Sensors over Other Fingerprint Sensors*

- Unique optical method captures fine details, even from dry skin

- Extremely low image-distortion for greater accuracy

- Resistance to damaging electrostatic discharge, moisture or corrosion

- Superior mechanical strength, wear-resistance and durability with no need for costly coatings

- Broad range of applicability, especially for use in extreme conditions and climates

- Low cost, long life, and no maintenance requirements – suitable for mass deployments

# Contents

# 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 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.so in this SDK, support the ANSI-INCITS 378-2004 standard and have been tested by NIST and proven to be MINEX Compliant:
    - 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
- Supports WSQ Image encoding and decoding

## 1.2. System Requirements

The SecuGen fingerprint reader captures a fingerprint image that is digitized into an 8-bit gray-scale image at 500 DPI resolution. The host system then retrieves the image through its USB port for subsequent processing. All SecuGen USB fingerprint readers, except for those based on FDU01 sensors, are supported in this SDK.

The following are the system requirements:

- IBM-compatible PC Pentium III or later
- 1 USB port (1.1 or higher) for the SecuGen USB fingerprint reader
- 64 MB RAM
- 80 MB available hard disk space
- Windows 10/8/7
- Java SDK v1.8.0_51 or later
- Java JRE v1.8.0_51 or later

## 1.3. Development Environment

**Requires: SecuGen FDx SDK Pro for Windows v3.54 or later**

### *1.3.1. Install the FDx SDK Pro for Windows*
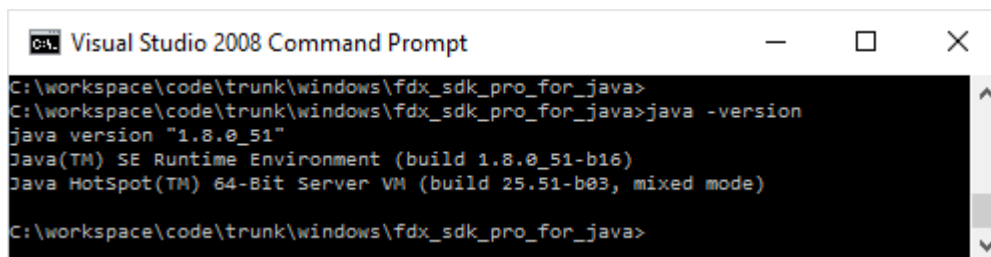
FDx SDK Pro for Java is a Java Native Interface (JNI) wrapper for FDx SDK Pro for Windows. The JNISGFPLIB runtime library included in this SDK passes calls to the SGFPLIB runtime library included in FDx SDK Pro for Windows. Refer to the FDx SDK Pro for Windows Installation Manual for detailed installation instructions.

### *1.3.2. Install the Java 2 SDK v1.8.0_51*

The Java SDK can be downloaded at www.oracle.com/technetwork/java. Refer to the Java documentation for detailed installation instructions.

After installing the Java SDK, verify that you have installed it correctly by launching a command prompt and running the following commands

- java –version
- javac –version



### *1.3.3. Copy the FDx SDK Pro for Java directory to your target location*

FDx SDK Pro for Java is distributed as a directory structure containing all required Jar files, the JNI library and various batch files that can be used to compile and run the included sample applications. As long as the Java SDK and FDx SDK Pro for Windows are correctly installed, the FDx SDK Pro for Java can be installed in any convenient location.

# Chapter 2. Installation

## 2.1. Installation

Copy the FDx SDK Pro for Java distribution into a new directory on the development machine.

## 2.2. Included Files

**Library Files**

>   **FDxSDKPRO.jar** –FDx SDK Pro for Java jar file
>   **jnisgfplib.dll** – SecuGen JNI library. Wrapper for sgfplib.dll
>   **jnisgwsqlib.dll** – SecuGen JNI library. wrapper for sgwsqlib.dll
>   **jnisgnfiqlib.dll** – SecuGen JNI library. Wrapper for sgnfiqlib.dll
>   **sgwsqlib.dll** – SecuGen WSQ library
>   **sgnfiqlib.dll** – SecuGen NFIQ library
>   **Absolutelayout.jar** – NetBeans 4.x Swing layout runtime

**Sample Program Files**

>   **extract_samples.bat** – Extracts sample source code
>   **build_samples.bat** – Builds sample applications
>   **run_JSGD.bat** – Runs the JSGD sample application
>   **run_JSGFPLibTest.bat** – Runs the JFPLibTest sample application
>   **run_JSGMultiDeviceTest.bat** – Runs the JSGMultiDeviceTest sample application

**Sample Signed Applet Files**

>   **applet/** – Directory containing signed applet demo
>   **genkey.bat** – Generates certificate & key pair
>   **buildsignedapplet.bat** – Builds signed jar file
>   **exportcert.bat** – Exports certificate
>   **JSGDAppletDemo.html** – Html file to load signed applet

**Documentation**

>   **readme.txt** – Latest release information for FDx SDK Pro for Java
>   **doc/** – Directory containing JavaDoc for FDx SDK Pro for Java
>   **FDx SDK Pro Programming Manual (Java).pdf** – This document

## 2.3. Run-time Distribution

Please follow the distribution instructions for FDx SDK Pro for Windows. Additionally, copy the FDx SDK Pro for Java runtime files as follows:

**Windows 7 32bit**

    Copy the following files to C:\windows\system32
        jnifplib\win32\jnisgfplib.dll
        jnifplib\win32\jnisgwsqlib.dll
        jnifplib\win32\sgwsqlib.dll
        jnifplib\win32\jnisgnfiqlib.dll
        jnifplib\win32\sgnfiqlib.dll

**Windows 7 64bit**

    Copy the following files to C:\windows\SysWOW64
        jnifplib\win32\jnisgfplib.dll
        jnifplib\win32\jnisgwsqlib.dll
        jnifplib\win32\sgwsqlib.dll
        jnifplib\win32\jnisgnfiqlib.dll
        jnifplib\win32\sgnfiqlib.dll

    Copy the following files to C:\windows\system32
        jnifplib\x64\jnisgfplib.dll
        jnifplib\x64\jnisgwsqlib.dll
        jnifplib\x64\sgwsqlib.dll
        jnifplib\x64\jnisgnfiqlib.dll
        jnifplib\x64\sgnfiqlib.dll

# Chapter 3. Programming in Java

SecuGen's FDx SDK *Pro* was designed for ease in programming and the most flexibility for developers. All SDK functions are integrated into the **JSGFPLib** class. The JSGFPLib class includes Device Initialization, Fingerprint Capture, and Minutiae Extraction and Matching functions.

## 3.1. Create JSGFPLib

To use JSGFPLib, call **JSGFPLib()**, which instantiates a JSGFPLib object.

```
JSGFPLib sgfplib = new
JSGFPLib((UsbManager)getSystemService(Context.USB_SERVICE));
```

## 3.2. Initialize JSGFPLib

After the JSGFPLib object is created, it should be initialized using **JSGFPLiB,Init()** or **JSGFPLib.InitEx()**. **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. **JSGFPLib.InitEx()** takes image width, image height and resolution as parameters. Call **JSGFPLib.InitEx()** when using the fingerprint algorithm module without a SecuGen reader.

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) |
|---|---|---|---|
| SG_DEV_UNKNOWN | 0 | Default | Based on Attached Device |
| SGDEV_FDP02 | 1 | Parallel device driver | 260*300 |
| SGDEV_FDU02 | 3 | USB FDU02 driver | 260*300 |
| SGDEV_FDU03 | 4 | USB FDU03 / SDU03 driver | 260*300 |
| SGDEV_FDU04 | 5 | USB FDU04 / SDU04 driver | 258*336 |
| SGDEV_FDU05 | 6 | USB U20 driver | 300*400 |
| SGDEV_FDU06 | 7 | USB UPx driver | 260*300 |

**JSGFPLib.Init()**

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

## 3.3. Terminate 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. Open 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 **USB_AUTO_DETECT** is recommended.

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

## 3.5. Get 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. Capture a Fingerprint Image

After the reader is initialized, a fingerprint image can be captured. The SGFPM 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 **SGFPM_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.

**[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. If a window handle is provided by the application, the drivers will draw a fingerprint image in the provided window using the handle value.

**[Example]**

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

## 3.7. Get 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. Use Smart Capture™ or Control Brightness Manually

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.

**JSGFPLib. SetBrightness()**

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

## 3.9. Create 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. Match 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 function is the same as JSGFPLib.MatchTemplateEx() except that it supports only ISO19794-2 templates.

| Function | Template Format | Can match templates with different formats? |
|---|---|---|
| SGFPM_MatchTemplate | SG400 (System default) | No |
| SGFPM_MatchTemplateEx | Specified template format | Yes |
| SGFPM_MatchAnsiTemplate | ANSI378 | No |
| SGFPM_MatchIsoTemplate | ISO19794-2 | No |

**JSGFPLib.MatchTemplate()**

```
byte[]RegTemplate1= new byte[maxTemplateSize[0]];
byte[]RegTemplate2= new byte[maxTemplateSize[0]];

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

// Get second fingerprint image and create template from 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_SG
400);
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. Register a Fingerprint

To register a fingerprint, a fingerprint image is first captured, and then feature data (minutiae) is extracted from the image to create 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 of two fingerprints is analogous to a password confirmation routine that is commonly required for entering a new password.

**Fingerprint Registration Process**

1. Capture fingerprint images: **JSGFPLib.GetImage()**
2. Extract minutiae from captured fingerprint to create a template: **JSGFPLib.CreateTemplate()**
3. Match newly made template to determine if it is 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. Verify a Fingerprint

The process of verifying a fingerprint 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 and the creation of a template.

**Fingerprint Verification Process**

1. Capture fingerprint image: **JSGFPLib.GetImage()**
2. Extract minutiae data from captured fingerprint to create a template: **JSGFPLib.CreateTemplate()**
3. Match newly made template against registered template(s): **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 the chances for 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. Get 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 score correlates 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 |

**Note**: Starting from version 3.53 of FDx SDK Pro for Windows, the Corresponding Matching Scores have changed.

## 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()

**Set template format to ANSI378**

```
JSGFPLib.SetTemplateFormat(SGFDxTemplateFormat
TEMPLATE_FORMAT_ANSI378);
```

**Set template format to SG400**

```
JSGFPLib.SetTemplateFormat(SGFDxTemplateFormat
TEMPLATE_FORMAT_SG400);
```

**Set template format to ISO19794**

```
JSGFPLib.SetTemplateFormat(SGFDxTemplateFormat
TEMPLATE_FORMAT_ISO19794);
```

## 3.15. Manipulate 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()

**Merge 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);
```

**Get 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. Manipulate 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()

**Merge 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);
```

**Get 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. Get 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()**

Instantiates the JSGFPLib object.

> **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 FDU03 and SDU03-based readers
>>>
>>> SG_DEV_FDU04: device name for USB FDU04 and SDU04-based readers

SG_DEV_FDU05: device name for USB U20-based readers

SG_DEV_FDU06: device name for USB UPx-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 InitEx(long width, long height, long dpi)

Initializes JSGFPLib with image information. Use when running fingerprint algorithm module without a SecuGen reader.

**Parameters**

*width*: Image width in pixels

*height*: Image height in pixels

*dpi*: Image resolution in DPI

**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_DLLLOAD_FAILED = Failed to load algorithm DLL

## 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 Capture Functions

**public long EnumerateDevice(int[] ndevs, SGDeviceList[] devList)**

Enumerates currently attached reader to the system.

**Parameters**

*ndevs*: The number of attached USB readers

*devList*: Buffer that contains device ID and device serial number.

**Return values**

SGFDX_ERROR_NONE = No error

SGFDX_ERROR_FUNCTION_FAILED = General function fail error

SGFDX_ERROR_INVALID_PARAM = Invalid parameter used

**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 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**
>
> > **buffer**: A byte array 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 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 GetImageEx(byte[] buffer, long timeout, long dispWnd , 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

>>   ***dispWnd***: null. Not used in Java

>>   ***quality***: The minimum quality value of an image, used to determine whether to accept the captured image

>   **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

## 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 Functions

**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

**public long WSQGetDecodedImageSize (int[] fingerImageOutSize, byte[] wsqImage, int wsqImageSize)**

Get the size of the RAW image contained in the WSQ image file. This function must be called before WSQDecode is called to allocate the size needed for the fingerimageOut buffer that will be populated with the raw image.

**Parameters**

*fingerImageOutSize*: Integer array that will be populated with the size of the WSQ image

*wsqImage*: Byte array containing the WSQ image

*wsqImageSize*: The size of the WSQ image file

**Return values**

SGFDX_ERROR_NONE = No error

**public long WSQDecode (byte[] fingerImageOut, int[] width, int[] height, int[] pixelDepth, int[] ppi, int[] lossyFlag, byte[] wsqImage, int wsqImageSize)**

Decode the WSQ image and return the RAW image. WSQGetDecodedImageSize() must be called first to allocate the size needed for the fingerimageOut buffer that will be populated with the raw image.

**Parameters**

*fingerImageOut*: Integer array that will be populated with the RAW image

*width*: Integer array that will be populated with the RAW image width.

*height*: Integer array that will be populated with the RAW image height.

*pixelDepth*: Integer array that will be populated with the RAW image pixelDepth. Example 8 bits per pixel.

*ppi*: Integer array that will be populated with the RAW image resolution. Example 500 ppi.

*lossyFlag*: Integer array that will be populated with the RAW image width.

*wsqImage*: Byte array containing the WSQ image

*wsqImageSize*: The size of the WSQ image file

**Return values**

SGFDX_ERROR_NONE = No error

**public long WSQGetEncodedImageSize (int[] wsqImageOutSize, float wsqBitRate, byte[] fingerImage, int width, int height, int pixelDepth, int ppi)**

Get the size of the compressed WSQ image that will be returned when the RAW image file is compressed. This function must be called before WSQEncode() is called to allocate the size needed for the wsqImageOut buffer that will be populated with the raw image..

**Parameters**

*wsqImageOutSize*: Integer array that will be populated with the size of the WSQ image

*wsqBitRate*:Compression bitrate to be used. Either BITRATE_5_TO_1 or BITRATE_15_TO_1.

*fingerImage*: Byte array containing a RAW fingerprint image

*width*: Width of the RAW image in pixels.

*height*: Height of the RAW image in pixels.

*pixelDepth*:Pixel depth of the image. Example - 8 bits per pixel.

*ppi*: Image resolution.

**Return values**

SGFDX_ERROR_NONE = No error

**public long WSQEncode (byte[] wsqImageOut, float wsqBitRate, byte[] fingerImage, int width, int height, int pixelDepth, int ppi)**

Get the size of the compressed WSQ image that will be returned when the RAW image file is compressed. This function must be called before WSQEncode() is called to allocate the size needed for the wsqImageOut buffer that will be populated with the raw image.

**Parameters**

*wsqImageOut*: Integer array that will be populated with the WSQ image

*wsqBitRate*: Compression bitrate to be used. Either BITRATE_5_TO_1 or BITRATE_15_TO_1.

*fingerImage*: Byte array containing a RAW fingerprint image

*width*:Width of the RAW image in pixels.

*height*: Height of the RAW image in pixels.

*pixelDepth*: Pixel depth of the image. Example - 8 bits per pixel.

*ppi*:Image resolution.

**Return values**

SGFDX_ERROR_NONE = No error

**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

# Chapter 5. Class Reference

## 5.1. Java Documentation

Refer to the "doc" folder in this SDK release for the complete JavaDoc class reference.

# Chapter 6. Constants

## 6.1. SGFDxDeviceName

| Device Name | Value | Description |
|---|---|---|
| SG_DEV_UNKNOWN | 0 | Not determined |
| SG_DEV_FDU03 | 0x04 | FDU03 or SDU03-based reader |
| SG_DEV_FDU04 | 0x05 | FDU04 or SDU04-based reader |
| SG_DEV_FDU05 | 0x06 | FDU05/U20-based reader |
| SG_DEV_FDU06 | 0x07 | FDU06/UPx-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.

# Chapter 7. Sample Applications

After installing the hardware and software, it is recommended that all components be checked to verify that they are working properly. The included sample applications can be used for this purpose

## 7.1. JSGD - Hardware Test Program

The **SecuGen Device Diagnostic Utility** program (JSGD.class) is located in the `FDxSDK.jar` archive. This program scans fingerprint images and also performs fingerprint registration and verification. If this program fails to capture a fingerprint image, the system is not configured correctly.

1. Launch a command prompt.

   ```
   cd <FDx_SDK_FOR_JAVA_INSTALL_DIR>
   ```

2. Type *run_JSGD.bat* and then **Enter**. The following command can also be used:

   ```
   java -cp ".;AbsoluteLayout.jar;FDxSDKPro.jar"
   SecuGen.FDxSDKPro.samples.JSGD
   ```

3. Click **Initialize** to initialize the reader. The result of initialization (success or failure) will be displayed in the status bar at the bottom left of the screen. If initialization fails, check the device connection and repeat the above steps.

If initialization is successful, place your finger on the fingerprint reader, and click **Capture**. The fingerprint image should be displayed if your reader is working properly.

## 7.2. JFPLib Test Program

The **JSGFPLibTest** program demonstrates all of the functionality included in FDx SDK Pro for Java.

1. Launch a command prompt.

```
cd <FDx_SDK_FOR_JAVA_INSTALL_DIR>
```

2. Type ***run_jsgfplibtest.bat*** and then **Enter.**

# Chapter 8. Signed Applet and Web Browser Integration

## 8.1. Install the FDx SDK Pro for Java Runtime Library

Windows 7 32bit:

> Copy jnifplib\win32\jnisgfplib.dll to C:\windows\system32
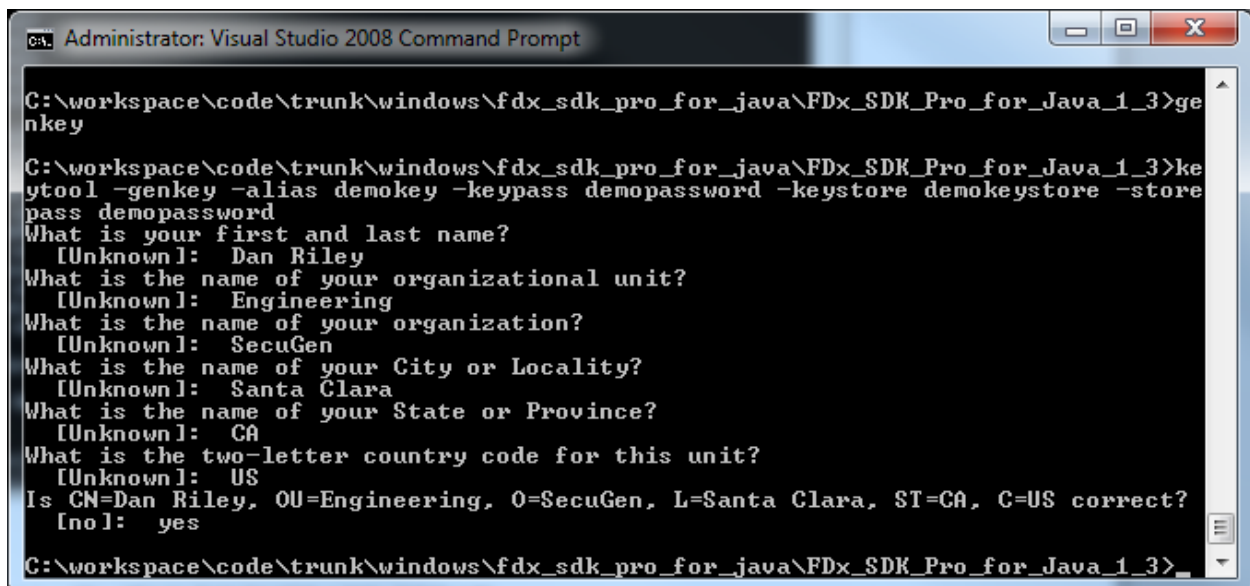
Windows 7 64bit:

> Copy jnifplib\win32\jnisgfplib.dll to C:\Windows\SysWOW64

> Copy jnifplib\x64\jnisgfplib.dll to C:\Windows\system32

## 8.2. Generate a Digital Certificate and Key Pair (GENKEY.BAT)

```
keytool -genkey -alias demokey -keypass demopassword -keystore
demokeystore -storepass demopassword
```

## 8.3. Build a Signed JAR File (BUILDSIGNEDAPPLET.BAT)

```
del AppletDemo.jar
jar xvf FDxSDKPro.jar
jar xvf AbsoluteLayout.jar
javac -deprecation applet\*.java
jar cvf AppletDemo.jar SecuGen\FDxSDKPro\jni\*.class
applet\*.class org\netbeans\lib\awtextra\*.*
jarsigner -keystore demokeystore -storepass demopassword -keypass
demopassword AppletDemo.jar demokey
```

## 8.4. Export the Certificate (EXPORTCERT.BAT)

```
keytool -export -alias demokey -storepass demopassword -keystore
demokeystore -file democert.csr
```



## 8.5. Import the Certificate into Java Plug-In
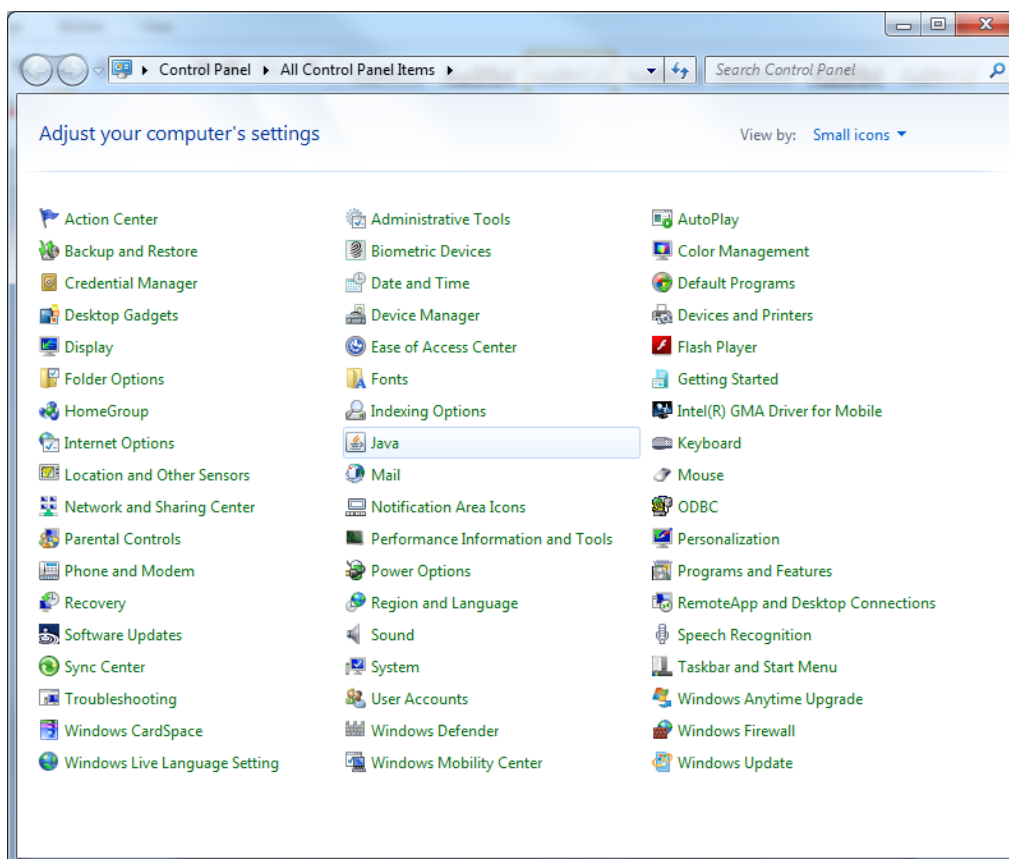
1.  Launch the Java Control Panel applet and select the Security tab.
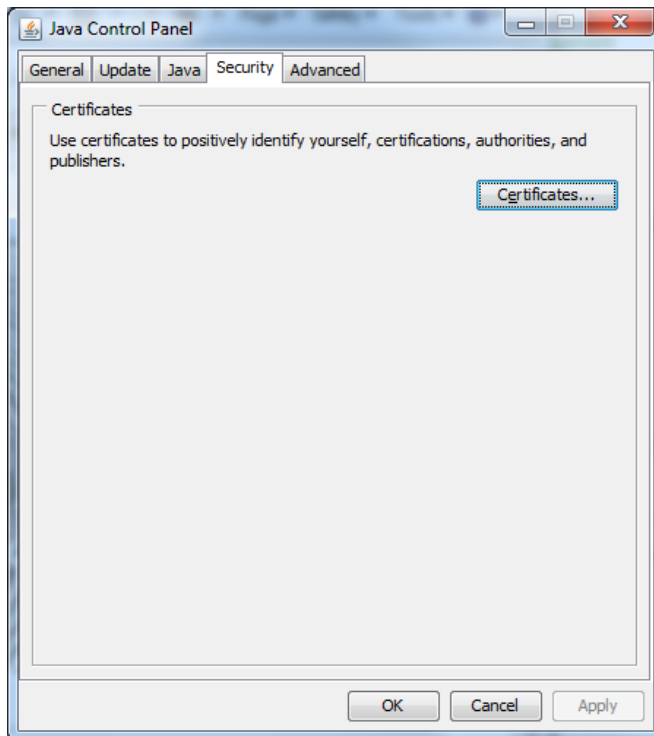


2.  Click the "Certificates" button and then click "Import."
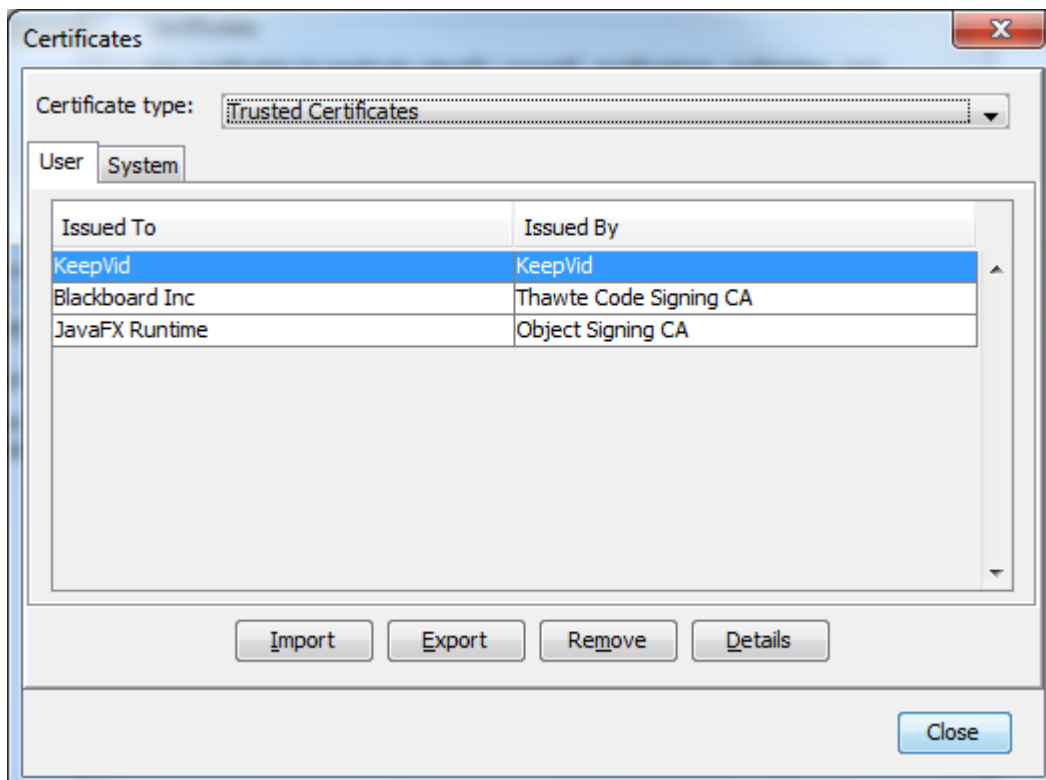
3.  Browse to the DEMOCERT.CSR certificate exported above and click "Open."
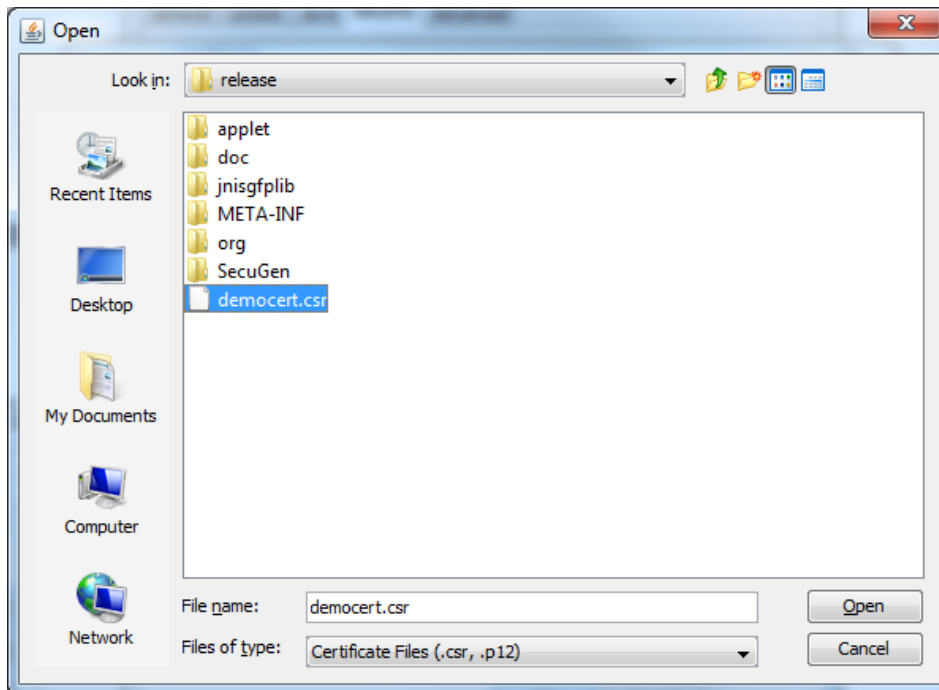


4.  The certificate is now installed in the JRE.
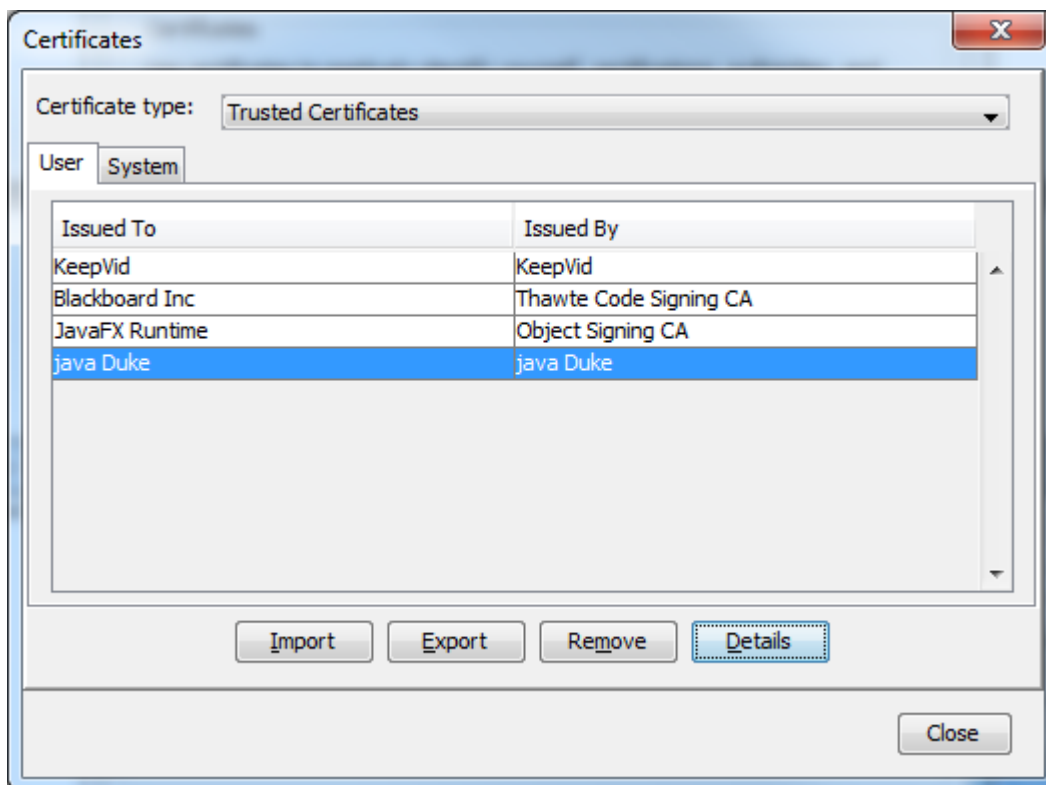
## 8.6. Open the Applet Demo HTML File

**JSGDAppletDemo.html**

_____

```
<html>
<head>
<title>
JSGD Applet Demo
</title>
</head>
<body>
<center>
<applet
  code     = "applet.JSGDApplet.class"
  name     = "JSGDApplet"
  archive  = "AppletDemo.jar"
  width    = "550"
  height   = "550"
  hspace   = "0"
  vspace   = "0"
  align    = "middle"
>
</applet>
</center>
</body>
</html>
```

_____

## 8.7. Mozilla FireFox Browser

1. Verify that the latest version of Java is installed. (JRE v8u65 and JDK v8u65)



2. Ensure that SecuGen devices drivers are correctly installed.
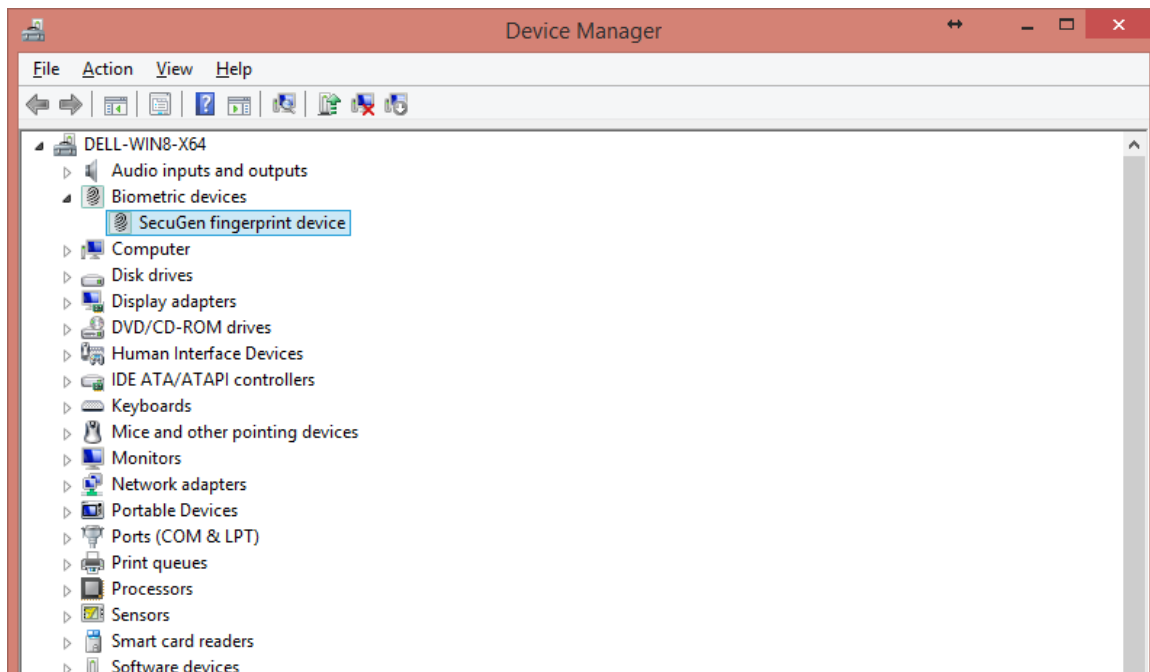
3.  Download and install the latest version of Mozilla Firefox. (Version 42)
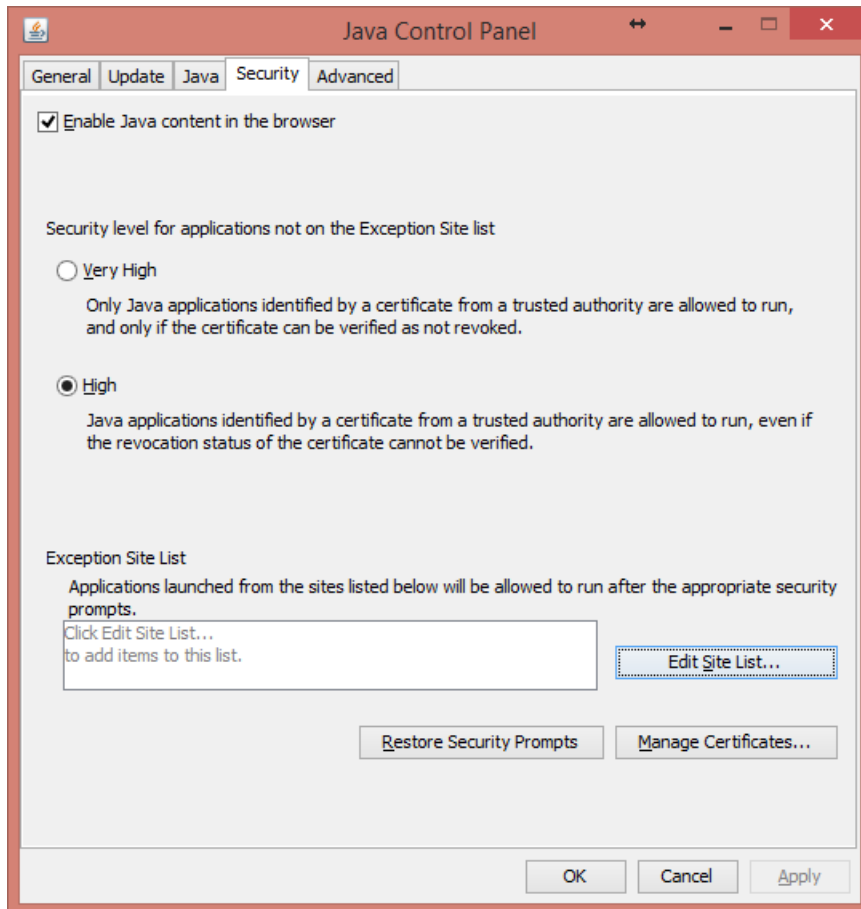
4.  Open JSGDAppetDemo.html with Mozilla Firefox
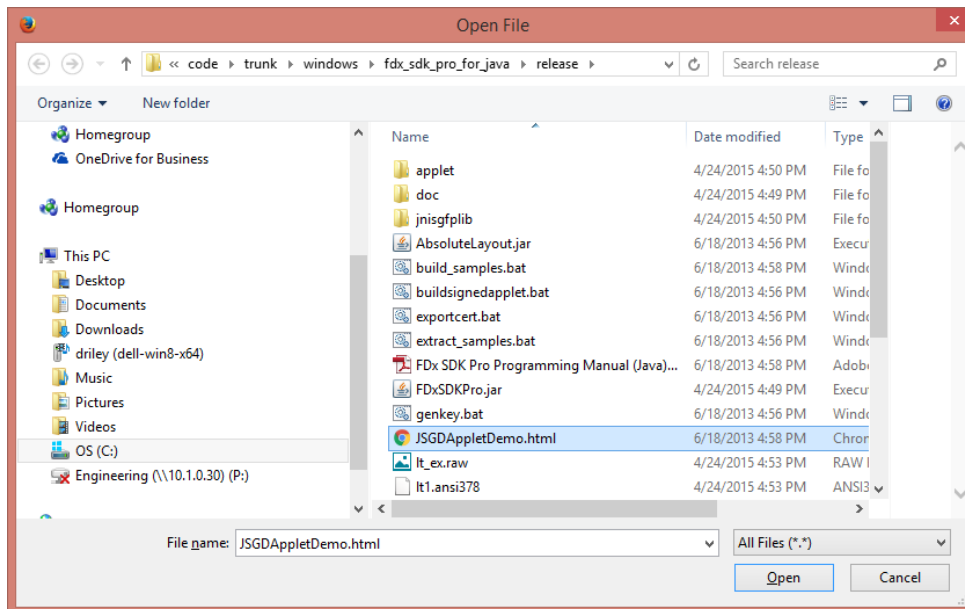


5.  If you get the following error, go to Control panel and open the Java control panel app.
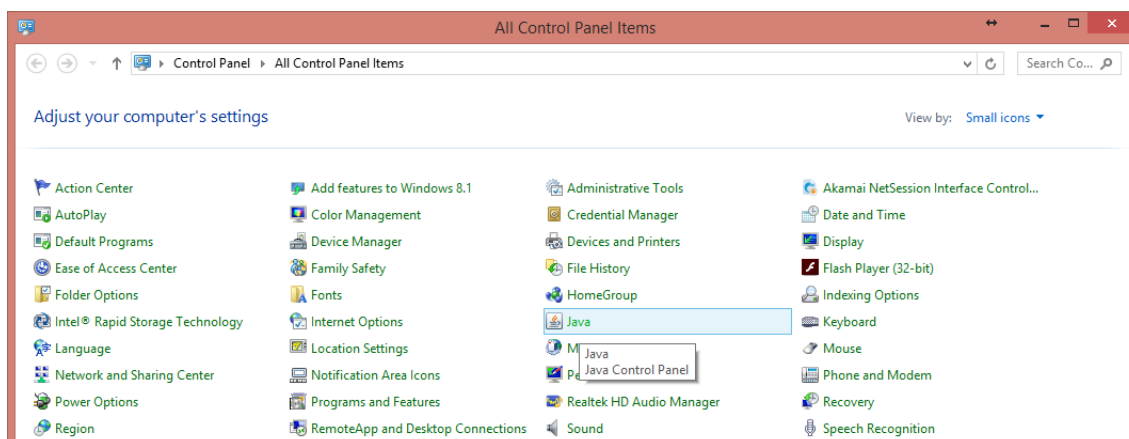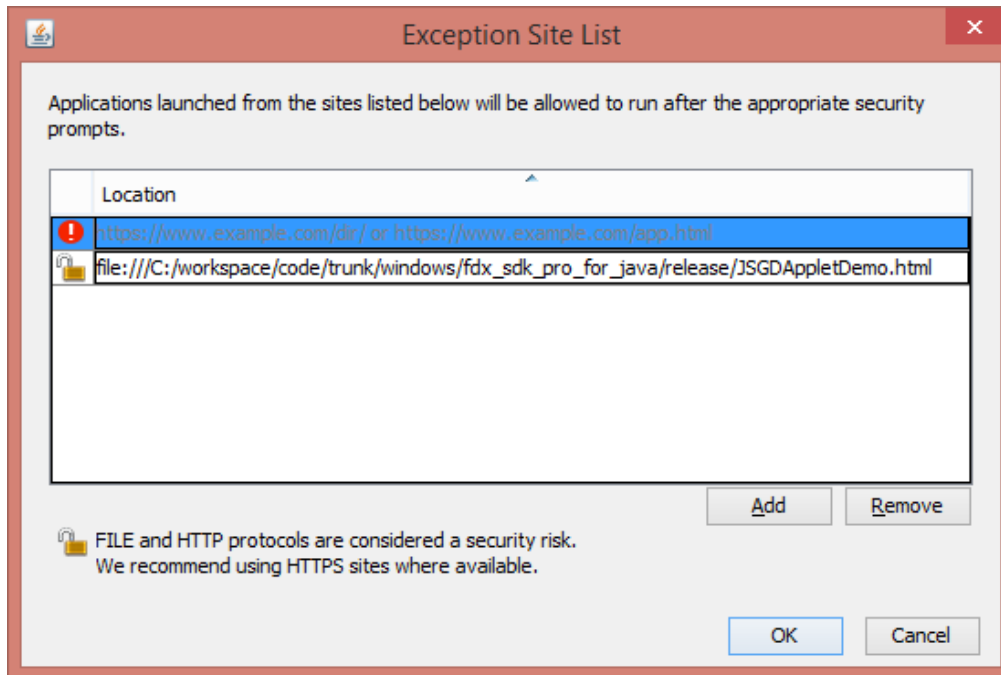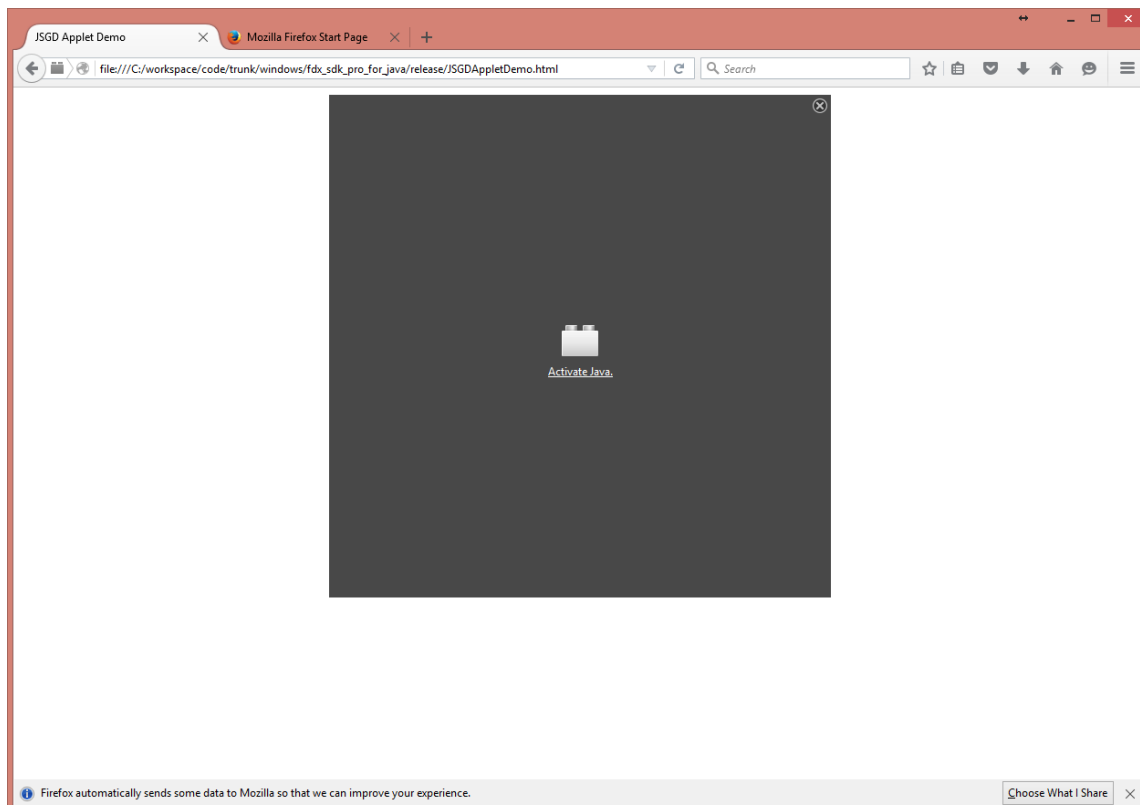
6.   Click on the Security tab.
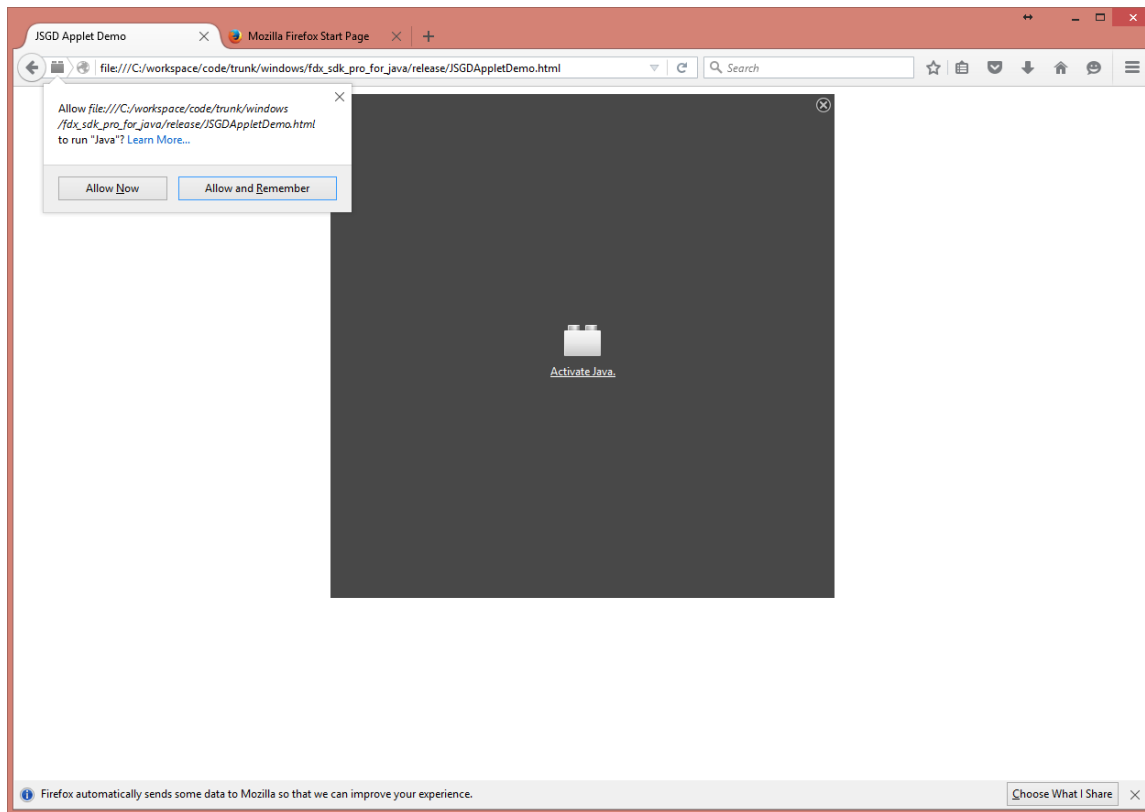
7.   Click Edit Site List.

8. Add the FILE URL path for your local test HTML file. You may need to do this also for the URL of the server hosted version.



9. Close Firefox and then relaunch. You will see a greyed applet area with a link to "Activate Java."

10. Click "Activate Java" and authorize the applet.



11. The Java applet is now running on Mozilla Firefox.