SM-93M API Manual

1.SM93MApiFactory.class

1.1 Get Instance

```
/**
 * This func return the instance of SM93MAPI
 *
 * @param context application context
 * @return an instance of SM93MAPI
 */
public static SM93MApi getInstance(Context context)
```

2.SM93MAPI.class

2.1 Open Device

```
/**
 * Turn on the fingerprint sensor .
 * NOTE : Don't call this method on main thread !
 *
 * @return file description number (positive), negative number for error code .
 */
@WorkerThread
int openDevice();
```

2.2 Close Device

```
/**
 * Turn off the fingerprint sensor .
 * NOTE : Don't call this method on main thread !
 *
 * @return 0 for successfully , negative number for error code .
 */
@WorkerThread
int closeDevice();
```

2.3 Get Device Info

```
/**
 * This function returns version of Fingerprint Collection Module.
 *
 * @return version
 */
 MxResult<String> getDeviceInfo();
```

2.4 Get Device SN

```
/**
 * This function returns serial number of Fingerprint Collection Module.
 *
 * @return serial number
 */
 MxResult<String> getDeviceSerialNumber();
```

2.4 Detect Finger

```
/**
 * This function returns whether the finger is on the sensor
 *
 * @return if true  on the sensor , otherwise not on the sensor
 */
 MxResult<Boolean> detectFinger();
```

2.5 Capture Image

```
/**
 * This function returns image captured from Fingerprint Collection Module.
 *
 * @return image ({@link Result#data})  captured from this device .
 * @see #getImage(CaptureConfig)
 */
 MxResult<MxImage> getImage();

/**
 * This function returns image captured from Fingerprint Collection Module.
 *
 * @param config capture config
 * @return image ({@link Result#data})  captured from this device .
 * @see #getImage()
 * @see CaptureConfig
 */
 MxResult<MxImage> getImage(CaptureConfig config);
```

2.6 Enroll

```
/**
 * This function returns the result of extracting registration fingerprint.
 *
 * @param flag            0 indicates the memory cache (BufferID is 0 to 2 feature
area, 3 template area). 1 indicates that flash bufferids range from 0 to 999.
 * @param bufferId        the serial number of the feature storage buffer, which is the
memory area or flash area specified by Flag.
 * @param checkDuplicates Duplicate Check Flag: 0 no duplicate check, 1 duplicate check
 * @return True is success, FALSE is failure
 */
MxResult<Boolean> enroll(int flag, int bufferId, boolean checkDuplicates);

/**
 * This function returns the result of extracting registration fingerprint.
 *
 * @param flag            0 indicates the memory cache (BufferID is 0 to 2 feature
area, 3 template area). 1 indicates that flash bufferids range from 0 to 999.
 * @param bufferId        the serial number of the feature storage buffer, which is the
memory area or flash area specified by Flag.
 * @param checkDuplicates Duplicate Check Flag: 0 no duplicate check, 1 duplicate check
 * @param config          capture config
 * @return True is success, FALSE is failure
 */
MxResult<Boolean> enroll(int flag, int bufferId, boolean checkDuplicates, CaptureConfig
config);
```

2.7 Verify

```
/**
 * This function returns the result of the fingerprint comparison.
 *
 * @param flag     0 indicates the memory cache (BufferID is 0 to 2 feature area, 3
template area). 1 indicates that flash bufferIds range from 0 to 999.
 * @param bufferId template buffer number, memory area or flash area specified by Flag.
 * @return True is success, FALSE is failure
 */
MxResult<Boolean> verify(int flag, int bufferId);
```

2.8 Search

```
/**
 * This function returns the result of a fingerprint search.
 *
 * @param startPage specifying the start page of the Flash fingerprint library (serial number)
 * @param pageNum   the number of pages searched from the start page.
 * @return the corresponding Flash page number (serial number).
 */
MxResult<Integer> search(int startPage, int pageNum);
```

2.9 Remove

```
/**
 * This function returns the result of delete the feature or template specified in memory buffer or flash.
 *
 * @param flag     0: for memory buffer, 1: for flash
 * @param bufferId if flag is 0, bufferId is 0, 1, 2, if flag is 1, flash page Id is 0~999
 * @return True is success, False is failure
 */
MxResult<Boolean> fingerDelete(int flag, int bufferId);
```

2.10 Clear

```
/**
 * This function returns the result of emptying the fingerprint repository.
 *
 * @return True is success, False is failure
 */
MxResult<Boolean> fingerErase();
```

2.11 Upload

```
/**
 * This function is to return fingerprint features to upload data.
 *
 * @param flag        0: for memory buffer, 1: for flash
 * @param bufferId    if flag is 0, bufferId is 0, 1, 2, if flag is 1, flash page Id is 0~999
 * @param featureType 0: Miaxis, 1: ISO2005, 2: ISO20011, 3: ANSI
 * @return the uploaded feature data
 */
MxResult<byte[]> uploadFeature(int flag, int bufferId, int featureType);
```

2.12 Download

```
/**
 * This function returns the result of downloading the fingerprint template
 *
 * @param flag        0: for memory buffer, 1: for flash
 * @param bufferId    if flag is 0, bufferId is 0, 1, 2, if flag is 1, flash page Id is
0~999
 * @param featureType 0: Miaxis, 1: ISO2005, 2: ISO20011, 3: ANSI
 * @param feature     feature data
 * @return True is success, False is failure
 */
MxResult<Boolean> downloadFeature(int flag, int bufferId, int featureType, byte[]
feature);
```

2.13 Set Feature Type

```
/**
 * This function returns the result of formatting the Mod feature
 *
 * @param featureType Mod feature : 0: Miaxis, 1: ISO2005, 2: ISO20011, 3: ANSI
 * @return True is success, False is failure
 */
MxResult<Boolean> setFeatureType(int featureType);
```

2.14 Get Feature Type

```
/**
 * This function returns the result of getting the Mod feature format
 *
 * @return Mod feature : 0: Miaxis, 1: ISO2005, 2: ISO20011, 3: ANSI
 */
MxResult<Integer> getFeatureType();
```

2.15 Get Feature Type

```
/**
 * This function returns the result of getting the Mod feature format
 *
 * @return Mod feature : 0: Miaxis, 1: ISO2005, 2: ISO20011, 3: ANSI
 */
MxResult<Integer> getFeatureType();
```

2.16 MOSIP Get Device Info

```
/**
 * This function returns the result of MOSIP Get device info
 *
 * @param deviceInfoParams Device Info
 * @param digitalIDParams  Digital Id
 * @return MOSIP device info
 */
MxResult<byte[]> mxGetDeviceInfo(MXDeviceInfo deviceInfoParams, MXDigitalid
digitalIDParams);
```

2.17 MOSIP Capture Compress Encrypt And Sign

```
/**
 * This function returns the result of MOSIP Capture Compress Encrypt And Sign info
 *
 * @param context        context
 * @param captureRequest Capture Request info
 * @param captureDetail  Capture Detail
 * @param digitalid      Digital Id
 * @return MOSIP Capture Compress Encrypt And Sign info
 */
MxResult<byte[]> mxCaptureCompressEncryptAndSign(Context context, CaptureRequest
captureRequest, CaptureDetail captureDetail, MXDigitalid digitalid);
```

2.18 Write File To Device

```
/**
 * This function returns the result of write digital certificate to device
 *
 * @param certificateNumber certificate number
 * @param certificateLength certificate length
 * @param certificate       certificate info
 * @return True is success, False is failure
 */
MxResult<Boolean> writeFileToDevice(int certificateNumber, int certificateLength,
byte[] certificate);
```

2.19 Delete File From Device

```
/**
 * This function returns the result of delete digital certificate to device
 *
 * @param certificateNumber certificate number
 * @return True is success, False is failure
 */
MxResult<Boolean> deleteFileFromDevice(int certificateNumber);
```

2.20 Read File From Device

```
/**
 * This function returns the result of read digital certificate to device
 *
 * @param certificateNumber certificate number
 * @return certificate info
 */
MxResult<byte[]> readFileFromDevice(int certificateNumber);
```

2.21 Generate RSA key air

```
/**
 * This function returns the result of generate RSA key air
 *
 * @param keyNumber Key pair number
 * @return True is success, False is failure
 */
MxResult<Boolean> genRSAKeyAir(int keyNumber);
```

2.22 Out RSA key air

```
/**
 * This function returns the result of out RSA key air
 *
 * @param keyNumber Key pair number
 * @return RSA Key
 */
MxResult<byte[]> outRSAKeyAir(int keyNumber);
```

2.23 Serial Communication API

2.23.1 Set ComPath and BaudRate

```
/**
 * Set the ComPath and BaudRate before calling openDevice
 * @param comPath  comPath
 * @param baudRate baudRate (default: 57600)
 */
setComPathBaudRate(String comPath, int baudRate);

setComPath(String comPath);

setBaudRate(int baudRate);
```

2.23.2 Modify Fingerprint Device BaudRate

```
/**
 * Modify the baud rate of the fingerprint device
 *
 * @param baudRate [1: 2400] [2: 9600] [3: 19200] [4: 57600] [5: 115200] [6: 230400]
[7: 460800] [8: 921600]
 * @return True is success, False is failure
 */
MxResult<Boolean> setDeviceBaudRate(int baudRate);
```

3. CaptureConfig.class

3.1 Constant

```
/**
 * Only try once
 */
public static final int TIMEOUT_TRY_ONCE = 0;
/**
 * Wait finger press forever
 */
public static final int TIMEOUT_FOREVER = -1;

/**
 * Don't wait finger press , an image without a finger may be returned
 *
 * @deprecated If you need to get a blank image, please call {@link #setAreaScore(int)}
with value 0
 */
@Deprecated
public static final int TIMEOUT_ORIGINAL_IMAGE = -2;
```

```
/**
 * Default acceptable minimum fingerprint area fraction for capturing images
 */
public static final int DEFAULT_AREA_SCORE = 45;
/**
 * Default timeout for capturing images
 */
public static final int DEFAULT_TIMEOUT = 8000;
/**
 * Do not enable AES encryption
 */
public static final int AES_NONE = 0;
/**
 * AES encryption on the host
 */
public static final int AES_HOST = 1;
/**
 * AES encryption on the device
 */
public static final int AES_DEVICE = 2;
```

3.2 Inner Class Builder

```
public static class Builder {}
```

3.3 Set Lfd Level

```
/**
 * Set Live fingerprint detection level
 *
 * @param lfdLevel the LFD level (1~5) , 0 for disable (default)
 * @return This CaptureConfig object to allow for chaining of calls to set methods.
 */
public CaptureConfig setLfdLevel(int lfdLevel)
```

3.4 Set Latent Level

```
/**
 * Set Latent fingerprint detection level
 *
 * @param latentLevel the Latent reject level (1~5) , 0 for disable (default)
 * @return This CaptureConfig object to allow for chaining of calls to set methods.
 */
public CaptureConfig setLatentLevel(int latentLevel);
```

3.5 Set Capture Timeout

```
/**
 * Set the timeout for capturing images
 *
 * @param timeout the timeout for capturing images.
 *                There are two special cases: {@link #TIMEOUT_TRY_ONCE},{@link
#TIMEOUT_FOREVER}
 * @return This CaptureConfig object to allow for chaining of calls to set methods.
 */
public CaptureConfig setTimeout(long timeout);
```

3.6 Set Area Score

```
/**
 * Set acceptable minimum fingerprint area fraction for capturing images
 *
 * @param areaScore 1~100 , default 45
 * @return This CaptureConfig object to allow for chaining of calls to set methods.
 */
public CaptureConfig setAreaScore(int areaScore);
```

3.7 Set AES config

```
/**
 * Set AES config
 */
public CaptureConfig setAESConfig(AESConfig aesConfig)
```

3.8 Set AES Status

```
/**
 * Set AES status
 *
 * @param aesStatus 0-Non,1-AES on Host,2-AES on device.
 * @see #AES_NONE
 * @see #AES_HOST
 * @see #AES_DEVICE
 */
public CaptureConfig setAESStatus(int aesStatus)
```

3.9 Set preview callback

```
public CaptureConfig setPreviewCallBack(FpPreviewCallBack previewCallBack)
```

4. AESConfig.class

4.1 Inner class Builder

```
public static class Builder {}
```

4.2 Set AES Mode

```
/**
 * Set mode of AES .
 */
public void setMode(String mode)
```

4.3 Set AES Padding

```
/**
 * Set padding of AES .
 */
public void setPadding(String padding)
```

4.4 Set AES Key

```
/**
 * Set key of AES , 16 bytes.
 */
public void setKey(String key)
```

4.5 Set AES iv

```
/**
 * Set iv of AES , 16 bytes.
 */
public void setIv(String iv)
```

5. MxImage.class

```
public class MxImage {
```

```
    /**
     * Image width
     */
    public final int width;
    /**
     * Image width
     */
    public final int height;
    /**
     * Image data
     */
    public final byte[] data;
    /**
     * 1: Grayscale image, 3: rgb, 4:argb
     */
    public final int channels;
    /**
     * Tag
     */
    public Object tag;
}
```

6. Result.class

```
public class MxResult<T> {

    /**
     * Error code,defined in the error code table
     * 0 success , negative number indicate errors
     */
    private final int code;
    /**
     * Error message
     */
    private final String msg;
    /**
     * Data
     */
    private final T data;

    private final int sw1;
}
```

7. Error Code Table

code	desc
-524001	SDK NOT INITIALIZED
-524002	DEVICE NOT CONNECTED
-524003	READ OR WRITE TIMEOUT
-524004	CAPTURE TIMEOUT
-524005	NO DEVICE
-524006	NO PERMISSION
-524007	FAILED TO WRITE OR READ DATA
-524008	FAILED TO WRITE DATA
-524009	FAILED TO READ DATA
-524010	CSW FORMAT ERROR
-524011	SCSI COMMAND EXECUTION FAILED
-524012	SCSI ERROR OCCURS AND NEEDS TO BE RESET
-524013	INSTRUCTION EXECUTION EXCEPTION
-524014	MOD TEMPLATE FORMAT ERROR
-524015	LATENT INITIALIZATION EXCEPTION
-524016	LATENT REJECT
-524017	NFIQ REJECT
-524018	LFD INITIALIZATION EXCEPTION
-524019	LFD REJECT