Терминалы сбора данных. Mertech MovFast

version 1.1.13

Mertech

October 24, 2024

Содержание

← Вернуться к списку всех документаций

Mertech MovFast

XCScanner SDK User Guide

Change log

Version	Date	Changes
1.0.0	2023/02/03	Basic scan result callback and settings.
1.0.3	2023/02/12	Add API.
1.0.4	2023/02/27	Add suspend and resume API.
1.0.6	2023/03/09	Add version info, loopscan, multibarcodes and precise scan about API.
1.0.7	2023/03/10	Add API to support config aimer and illume light work mode.
1.0.8	2023/03/13	Fixed SDK version in docs.
1.0.9	2023/03/14	Add API to support license active and license state query.
1.1.0	2023/03/15	Add API to support get scan service status.
1.1.2	2023/04/03	Add API to support get the latest decode image.
1.1.3	2023/04/11	Add API to support set suffix2 and prefix2.
1.1.8	2024/05/16	Add API to support set custom BroadcastReceiver, Disable/Enable Scan button, Export/Import configuration file, Configure barcode output failure event notification, Configure flash brightness.
1.1.9	2024/08/26	Add API to support set/get properties for the EAN13/Matrix25/UPCA symbology.
1.1.10	2024/09/24	Add API to support set/get properties for the scan trigger mode.
1.1.11	2024/10/10	Add API to support set/get properties for the Code39/DATAMATRIX/EAN8 symbology.
1.1.12	2024/10/17	Add API to support set/get properties for the code11/coded49/code93/code128/codeabar symbology.
1.1.13	2024/10/18	Add API to support set/get properties for the GS1-128/GS1-DATABAR/ITF25/MSI/QRCode/UPCE symbology.

Config Maven

Config Maven.

```
maven {
    allowInsecureProtocol = true
    url "http://47.108.228.164:8081/nexus/service/local/repositories/releases/content/"
}
```

Note

Maven is configuared in *build.gradle* usually, but also maybe in your *settings.gradle*.

Config Dependencies

Config your project build.gradle, as following example:

```
implementation('com.xcheng:scanner:1.1.13')
```

It is recommended to use the latest version of SDK.

Basic function

SDK initialize

After init SDK, you can use scan function provided by scan service via APIs.Currently, there are two ways to initialize SDK

```
XcBarcodeScanner.init(Context context, ScannerResult scannerResult)

XcBarcodeScanner.init(Context context, ScannerSymResult scannerSymResult)
```

Callback interface:

```
public interface ScannerResult {
    void onResult(String result);
}

public interface ScannerSymResult {
    void onResult(String sym, String barCode);
}
```

After init SDK, your application will connect with scan service, and the scan result will be output via the ScannerResult/ScannerSymResult callback.

Sample code:

```
XcBarcodeScanner.init(this, new ScannerResult() {
     @Override
     public void onResult(String result) {
          runOnUiThread(new Runnable() {
               @Override
               public void run() {
                    allResult = allResult + "\n" + result;
                    mTextResult.setText(allResult);
                    scrollToBottom();
               }
          });
     }
 });

 XcBarcodeScanner.init(this, new ScannerSymResult() {
     @Override
     public void onResult(String sym, String barCode) {
          runOnUiThread(new Runnable() {
               @Override
               public void run() {
                    allResult = allResult + "\n" + sym + ":" + barCode;
                    mTextResult.setText(allResult);
                    scrollToBottom();
               }
          });
     }
 });
```

SDK deinitialize

When the activity been destroied or switched to background, we need de-initialize the SDK.

Sample code:

```
@Override
protected void onPause() {
    super.onPause();

    XcBarcodeScanner.deInit(this);
}
```

Query license status

Use the following API to query license status.

```
XcBarcodeScanner.getLicenseState();
```

All license status defined in the class *LicenseState*:

```
public class LicenseState {
        // License not actived.
        public static final int INACTIVE = 0;
        // License is activating.
        public static final int ACTIVATING = 1;
        // License already actived.
        public static final int ACTIVED = 2;
        // License invalid, contact vendor.
        public static final int INVALID = 3;
        // Network issue, connet before active license.
        public static final int NETWORK_ISSUE = 4;
        // License expired, contact vendor.
        public static final int EXPIRED = 5;
}
```

Active license

Use the following API to active license if needed.

```
XcBarcodeScanner.activateLicense();
```

After active license, it need about 1 or 2 minutes to process. We can use the API *getLicenseState* to query license status.

Note

Active license need network connection.

Start/Stop scanning

Use the following API to control scan service to start or stop scanning.

```
XcBarcodeScanner.startScan();
XcBarcodeScanner.stopScan();
```

Suspend/Resume scan service

Use the following API to suspend or resume scan service.

After suspend, any API to control scan start or stop will be bypassed, and camera resource will be released by scan service.

```
XcBarcodeScanner.suspendScanService();
XcBarcodeScanner.resumeScanService();
```

Get scan service status

Use the following API to check current scan service status.

```
boolean isScanServiceSuspending();
```

Get version of scan service

Refer to the following sample code to get version info of scan service.

```
String serviceVer = XcBarcodeScanner.getServiceVersion();
Log.d(TAG, "Service ver: " + serviceVer);
```

Get version of SDK

Refer to the following sample code to get current scanner SDK version.

```
String sdkVer = XcBarcodeScanner.getSdkVersion();
Log.d(TAG, "SDK ver: " + sdkVer);
```

Get the device serial number

```
public static String getDeviceSN(){
        String serial = null;
        try {
            Class<?> c =Class.forName("android.os.SystemProperties");
            Method get =c.getMethod("get", String.class);
            serial = (String)get.invoke(c, "ro.serialno");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return serial;
  }
```

Advanced function

Enable/disable specific barcode support

Use the following API to enable or disable specific type of barcode support.

```
XcBarcodeScanner.enableBarcodeType(String barcodeType, boolean enabled);
```

All barcode types defined in the class BarcodeType.

```
public class BarcodeType {
    public static final String AZTEC = "Aztec";
    public static final String CODE11 = "Code11";
    public static final String CODE39 = "Code39";
    public static final String CODE93 = "Code93";
    public static final String CODE128 = "Code128";
    public static final String CODABAR = "Codabar";
    public static final String DOTCODE = "Dotcode";
    public static final String CODABLOCKF = "CODABLOCK F";
    public static final String DATAMATRIX = "DATAMATRIX";
    public static final String GS1_DATAMATRIX = "GS1DATAMATRIX";
```

```
    public static final String EAN8 = "EAN-8";
    public static final String EAN13 = "EAN-13";
    public static final String GS1_DATABAR = "GS1_DATABAR";
    public static final String HANXIN = "HANXIN";
    public static final String HK25 = "HK25";
    public static final String ITF25 = "ITF25";
    public static final String MATRIX25 = "MATRIX25";
    public static final String MAXICODE = "MAXICODE";
    public static final String MSI = "MSI";
    public static final String MICROPDF = "MICROPDF";
    public static final String PDF417 = "PDF417";
    public static final String USPS4ST = "USPS4ST";
    public static final String QRCODE = "QRCODE";
    public static final String INDUSTRIAL25 = "INDUSTRIAL25";
    public static final String TELEPEN = "TELEPEN";
    public static final String UPCA = "UPC-A";
    public static final String UPCE = "UPC-E";
    public static final String GS1_128 = "GS1-128";
    public static final String GS1_DATABAR_LIMITED = "GS1_DataBarLimited";
    public static final String GS1_DATABAR_EXPANDED = "GS1_DataBarExpanded";
    public static final String IATA25 = "IATA25";
    public static final String GRIDMATRIX = "Grid_Matrix";
}
```

Sample code:

```
XcBarcodeScanner.enableBarcodeType(BarcodeType.QRCODE, true); // Enable QRCode support.
XcBarcodeScanner.enableBarcodeType(BarcodeType.QRCODE, false);// Disable QRCode support.
```

Check specific type of barcode support status

All barcode types defined in the class BarcodeType.

```
boolean isBarcodeTypeEnabled(String type)
```

Config output method

Use the following API to config scan result output method.

```
XcBarcodeScanner.setOutputMethod(String method);
```

All output method defined in the class OutputMethod.

```
public class OutputMethod {
    // No putput, application can get barcode data via callback.
    public static final String NONE = "NONE";
    // Output barcode data via broadcast.
    public static final String BROADCAST = "BROADCAST_EVENT";
    // Output barcode data via keyboard simulate.
    public static final String KEYBOARD = "KEYBOARD_EVENT";
    // Output barcode data via clipboard simulate.
    public static final String CLIPBOARD = "CLIPBOARD_EVENT";
    // Output barcode data via broadcast and keyboard simulate.
    public static final String BROADCAST_KEYBOARD = "BROADCAST_EVENT/KEYBOARD_EVENT";
    // Output barcode data via broadcast and clipboard simulate.
    public static final String BROADCAST_CLIPBOARD = "BROADCAST_EVENT/CLIPBOARD_EVENT";
}
```

Sample code:

```
XcBarcodeScanner.setOutputMethod(OutputMethod.CLIPBOARD);
```

Config barcode output event notification

Use the following API to config type of notification when scan success.

```
XcBarcodeScanner.setSuccessNotification(String notification);
```

All types of notifications defined in the class NotificationType.

```
public class NotificationType {
    public static final String MUTE = "Mute";
    public static final String SOUND = "Sound";
    public static final String VIBRATOR = "Vib";
    public static final String SOUND_VIBRATOR = "Sound/Vib";
}
```

Sample code:

```
XcBarcodeScanner.setSuccessNotification(NotificationType.SOUND);
```

Config barcode scan LED indicator

Use the following API to enable or disable scan LED indicator.

```
XcBarcodeScanner.enableSuccessIndicator(boolean enabled);
```

Sample code:

```
XcBarcodeScanner.enableSuccessIndicator(true); // Enable LED indicator.
XcBarcodeScanner.enableSuccessIndicator(false); // Disable LED indicator.
```

Config scan time limit

Use the following API to config scan time limitation.

```
XcBarcodeScanner.setTimeout(int seconds);
```

The range of time limitation: 1 ~ 9 seconds

Sample code:

```
XcBarcodeScanner.setTimeout(5); // Set scan time limit to 5 seconds.
```

Config aimer light

Use the following API to config aimer light work mode.

```
void setAimerLightsMode(int aimMode);
```

Aimer light work mode defined in the class *AimerMode* :

```
public class AimerMode {
    public static int ALWAYS_OFF; // Always Off.
    public static int TRIGGER_ON; // TurnOn while scanning.
    public static int ALWAYS_ON; // Always On.
}
```

Sample code:

```
// Set aimer mode to: Turn on while scanning.
XcBarcodeScanner.setAimerLightsMode(AimerMode.TRIGGER_ON);
// Set aimer mode to: always off.
XcBarcodeScanner.setAimerLightsMode(AimerMode.ALWAYS_OFF);
```

Warning

For laser aimer, DO NOT USE THE *ALWAYS_ON* mode.

Config illume light

Use the following API to config illume light work mode.

```
void setFlashLightsMode(int flashMode);
```

The work mode of illume light defined in class *FlashMode* :

```
public class FlashMode {
    public static int OFF;           // Always off.
    public static int ILLUME_ONLY;   // Only illume.
    public static int ILLUME_STROBE; // Illume and Strobe.
}
```

Sample code:

```
// Set illume to: illume and strobe
XcBarcodeScanner.setFlashLightsMode(FlashMode.ILLUME_STROBE);
// Set illume to: illume only
XcBarcodeScanner.setFlashLightsMode(FlashMode.ILLUME_ONLY);
```

Note

The illume light only turn on while scanning, the *STROBE* will output FLASH light which sync with image sensor shutter.

Config scan result text case switch

Use the following API to config case switch of scan result.

```
XcBarcodeScanner.setTextCase(String textCase);
```

All case options defined in the class TextCaseType.

```
public class TextCaseType {
    public static final String NONE = "NONE_CASE";
    public static final String UPPER = "UPPER_CASE";
    public static final String LOWER = "LOWER_CASE";
}
```

Sample code:

```
XcBarcodeScanner.setTextCase(TextCaseType.NONE);  // No case switch.
XcBarcodeScanner.setTextCase(TextCaseType.UPPER); // Switch to Uppercase.
XcBarcodeScanner.setTextCase(TextCaseType.LOWER); // Switch to Lowercase.
```

Config prefix of barcode result

Use the following API to config prefix of barcode result.

```
XcBarcodeScanner.setTextPrefix(String prefix);
XcBarcodeScanner.setTextPrefix2(String prefix2);
```

Sample code:

```
XcBarcodeScanner.setTextPrefix("<");  // Config prefix as "<"
XcBarcodeScanner.setTextPrefix2(":");  // Config prefix2 as ":"
XcBarcodeScanner.setTextPrefix("Empty"); // Config prefix None.
```

Config suffix of barcode result

Use the following API to config suffix of barcode result.

```
XcBarcodeScanner.setTextSuffix(String suffix);
XcBarcodeScanner.setTextSuffix2(String suffix2);
```

Sample code:

```
XcBarcodeScanner.setTextSuffix(">");  // Config suffix as ">"
XcBarcodeScanner.setTextSuffix2(":");  // Config suffix2 as ":"
XcBarcodeScanner.setTextSuffix("Empty"); // Config suffix None.
```

Config interval of loopscan

Use the following API to config interval of loopscan.

```
setLoopScanInterval(int ms);
```

Sample code:

```
XcBarcodeScanner.setLoopScanInterval(100); // Config interval of loopscan as 100 ms.
```

Get running status of loopscan

Use the following API to get running status of loopscan.

```
boolean isLoopScanRunning();
```

Start/Stop loopscan

Use the following API to start or stop loopscan.

```
startLoopScan();
stopLoopScan();
```

Sample code:

```
// Start loopscan.
XcBarcodeScanner.setLoopScanInterval(100); // Set loopscan interval to 100 ms
XcBarcodeScanner.startLoopScan(); // start loopscan.

// Stop loopscan
if (XcBarcodeScanner.isLoopScanRunning()) { // Check if loopscan is running.
  XcBarcodeScanner.stopLoopScan(); // stop loopscan.
}
```

Config multibarcodes

Use the following API to config multibarcodes options.

```
void setMultiBarcodes(int numberOfBarcodes, boolean fixedNumber);
```

The parameters:

```
numberOfBarcodes - Max barcodes in one shot. range: 1 ~ 20
fixedNumber - true means fixed number of barcodes. false means non-fixed.
```

Note

Fixed number of barcodes means it will not output till the numberOfBarcodes barcodes been scanned. non-Fixed means it will output barcodes been scanned, it may smaller or equal to the numberOfBarcodes.

Sample code:

```
// Config numberOfBarcodes to 3, and be fixedNumber.
XcBarcodeScanner.setMultiBarcodes(3, true);

// Config numberOfBarcodes to 1
XcBarcodeScanner.setMultiBarcodes(1, false);
```

Note

if the numberOfBarcodes been set to 1, it is single barcode mode, the fixedNumber option is meaningless.

Config region size of barcode scan

Use the following API to config the region size of barcode scanning. The typical usage is 1D barcode precise scanning.

```
void setScanRegionSize(int regionSize);
```

All supported region size defined in class RegionSizeType:

```
public class RegionSizeType {
    public static final int VIEWSIZE_100 = 0; // 100% frame region.
    public static final int VIEWSIZE_75 = 1;  // 75% frame region.
    public static final int VIEWSIZE_50 = 2;  // 50% frame region.
    public static final int VIEWSIZE_25 = 3;  // 25% frame region.
    public static final int VIEWSIZE_12 = 4;  // 12% frame region.
    public static final int VIEWSIZE_6 = 5;  // 6% frame region.
    public static final int VIEWSIZE_3 = 6;  // 3% frame region.
    public static final int VIEWSIZE_1 = 7;  // 1% frame region.
    public static final int VIEWSIZE_1D = 8;  // 1D barcode region.
}
```

Sample code:

```
// Config scan region to 1D barcode region.
XcBarcodeScanner.setScanRegionSize(RegionSizeType.VIEWSIZE_1D);

// Config barcode scan region to max region. (All 100% of framne)
XcBarcodeScanner.setScanRegionSize(RegionSizeType.VIEWSIZE_100);
```

Note

if enabled multibarcodes and numberOfBarcodes more than 1, it is recommend to use 100% region of frame.

Get the last one decode image

Use the following API to get the last image to decode.

```
XCImage getLastDecodeImage();
```

The XCImage is the image class object retured.

Sample code:

```
import com.tools.XCImage;

public void getLastImage() {
    XCImage lastImg=XcBarcodeScanner.getLastDecodeImage();

    if(lastImg!=null){
        String infoStr= "Witdh: " + lastImg.getWidth() + ", ";
        infoStr += "Height: " + lastImg.getHeight() + ", ";
        infoStr += "Stride: " + lastImg.getStride() + ", ";
        infoStr += "Size: " + lastImg.getData().length + " Bytes";
        showAlertDialog("Image Info:",infoStr,false,"OK",null);
    }else{
        showAlertDialog("Image Info:","No image!",false,"OK",null);
    }
}
```

Set custom BroadcastReceiver

Use the following API to configure the Action and Key for custom broadcasts. After successful configuration, the scan result can be received through the broadcast.

```
void setScanResultBroadcast(String action, String resultKey);
```

Sample code:

```
XcBarcodeScanner.setScanResultBroadcast("xxx.Action", "scanResultKey");
```

Disable/Enable Scan button

Due to differences in device design, the currently supported scanning buttons are: left side scanning button/right side scanning button/front scanning button/handheld handle scanning button. After disabling the scanning button function, pressing the scanning button will prevent scanning; After enabling the scanning button function, the button will restore the scanning function.

```
void setFrontScanKeyEnable(boolean isEnable); // front scanning button
void setLeftScanKeyEnable(boolean isEnable); // left side scanning button
void setRightScanKeyEnable(boolean isEnable); // right side scanning button
void setPoGoScanKeyEnable(boolean isEnable); // handheld handle scanning button
```

Sample code:

```
XcBarcodeScanner.setFrontScanKeyEnable(false); // Disable front scanning button
XcBarcodeScanner.setFrontScanKeyEnable(true); // Enable front scanning button
```

Export configuration file

Export the currently used configuration file to the specified directory.

```
XcBarcodeScanner.exportSettings(String exportPath);
```

The format of the exported file must be of XML type, and the file name can only contain letters and numbers.Sample code:

```
String exportPath=Environment.getExternalStorageDirectory().getPath()+"/Scanner.xml";
// Export the configuration file to sdcard and rename it to Scanner.xml
XcBarcodeScanner.exportSettings(exportPath);
```

Import configuration file

You can use the configuration files in the specified directory through the interface provided by the SDK.

```
XcBarcodeScanner.importSettingsByProfileName(String profileName, String importPath);
```

Note

The configuration file name can only contain letters and numbers. And the configuration file must be exported through the «XcBarcodeScanner. exportSettings» interface.

Sample code:

```
String fileName = "Scanner";   // The file name cannot contain type suffix
String importPath=Environment.getExternalStorageDirectory().getPath()+"/Scanner.xml";
XcBarcodeScanner.importSettingsByProfileName(fileName, importPath);
```

Configure barcode output failure event notification

Use the following API to config type of notification when scan failed.

```
XcBarcodeScanner.setFailNotification(String notification);
```

All types of notifications defined in the class NotificationType.

```
public class NotificationType {
        public static final String MUTE = "Mute";
        public static final String SOUND = "Sound";
        public static final String VIBRATOR = "Vib";
        public static final String SOUND_VIBRATOR = "Sound/Vib";
}
```

Sample code:

```
XcBarcodeScanner.setFailNotification(NotificationType.MUTE);
XcBarcodeScanner.setFailNotification(NotificationType.SOUND);
```

Configure flash brightness

Use the following API to config flash brightness.

```
XcBarcodeScanner.setStrobeLightBrightness(int brightness);
```

All types of brightness defined in the class StrobeLightBrightness.

```
public class StrobeLightBrightness {
        public static int FULL_BRIGHTNESS = 4;
        public static int MEDIUM_BRIGHTNESS = 7;
        public static int WEAK_BRIGHTNESS = 5;
        public static int WEAKEST_BRIGHTNESS = 6;
}
```

Sample code:

```
XcBarcodeScanner.setStrobeLightBrightness(StrobeLightBrightness.WEAK_BRIGHTNESS);
```

Get barcode properties

Use the following API to get barcode properties.

```
int getDecoderTagValue(int tag);
```

The properties that support queries are defined in the XCBarcodeTag class:

```
public class XCBarcodeTag {
        // Code39
        // Check digit options.
        // 0:Disable Check Digit
        // 1:Enable Check Digit and No Output
        // 2:Enable Check Digit and Output
        public static final int TAG_CODE39_CHECK_DIGIT_MODE      = 0x1A016004;
        // Transmit start/stop char.
        // 1:enable;0:disable
        public static final int TAG_CODE39_START_STOP_TRANSMIT   = 0x1A016007;
        // Code 39 Full ASCII.
         // 1:enable;0:disable
        public static final int TAG_CODE39_FULL_ASCII_ENABLED    = 0x1A016006;
        // Code39 Base32 decode.
        // 1:enable;0:disable
        public static final int TAG_CODE39_BASE32_ENABLED        = 0x1A016008;
        // Maximum length(1-127).
        // The range of values is integers from 1 to 127.
        public static final int TAG_CODE39_MAX_LENGTH            = 0x1A016003;
        // Minimum length(1-127).
        // The range of values is integers from 1 to 127.
        public static final int TAG_CODE39_MIN_LENGTH            = 0x1A016002;

        // DataMatrix
        // With Separators.
        // 1:display;0:hide
        public static final int TAG_DATAMATRIX_SEPARATOR_ENABLED = 0x1A029004;
        // Max out length (0: no limit).
        // An integer greater than or equal to 0, where 0 indicates no restriction.
        public static final int TAG_DATAMATRIX_OUTPUT_MAX_LENGTH = 0x1A029005;

        // EAN-8
        // Transmit check digit.
        // 1:enable;0:disable
        public static final int TAG_EAN8_CHECK_DIGIT_TRANSMIT    = 0x1A012002;
        // 2 Digit Addenda.
        // 1:enable;0:disable
        public static final int TAG_EAN8_2CHAR_ADDENDA_ENABLED   = 0x1A012003;
        // 5 Digit Addenda.
        // 1:enable;0:disable
        public static final int TAG_EAN8_5CHAR_ADDENDA_ENABLED   = 0x1A012004;
        // Addenda Required.
        // 1:enable;0:disable
        public static final int TAG_EAN8_ADDENDA_REQUIRED        = 0x1A012005;
        // Addenda add Separator.
        // 1:enable;0:disable
        public static final int TAG_EAN8_ADDENDA_SEPARATOR       = 0x1A012006;

        // EAN-13
        // Transmit check digit.
        // 1:enable;0:disable
        public static final int TAG_EAN13_CHECK_DIGIT_TRANSMIT   = 0x1A013002;
        // 2 Digit Addenda.
        // 1:enable;0:disable
```

```
        public static final int TAG_EAN13_2CHAR_ADDENDA_ENABLED  = 0x1A013003;
        // 5 Digit Addenda.
        // 1:enable;0:disable
        public static final int TAG_EAN13_5CHAR_ADDENDA_ENABLED  = 0x1A013004;
        // Addenda Required.
        // 1:enable;0:disable
        public static final int TAG_EAN13_ADDENDA_REQUIRED       = 0x1A013005;
        // Addenda add Separator.1:enable;0:disable
        public static final int TAG_EAN13_ADDENDA_SEPARATOR      = 0x1A013006;

        // Matrix 2 of 5
        // Check digit options.
        // 0:Disable Check Digit
        // 1:Enable Check Digit and Output
        // 2:Enable Check Digit and No Output
        public static final int TAG_M25_CHECK_DIGIT_MODE         = 0x1A01C004;

        // UPC-A
        // Transmit check digit.
        // 1:enable;0:disable
        public static final int TAG_UPCA_CHECK_DIGIT_TRANSMIT    = 0x1A010002;
        // Number system digit.
        // 1:enable;0:disable
        public static final int TAG_UPCA_NUMBER_SYSTEM_TRANSMIT  = 0x1A010003;
        // 2 Digit Addenda.
        // 1:enable;0:disable
        public static final int TAG_UPCA_2CHAR_ADDENDA_ENABLED   = 0x1A010004;
        // 5 Digit Addenda.
        // 1:enable;0:disable
        public static final int TAG_UPCA_5CHAR_ADDENDA_ENABLED   = 0x1A010005;
        // Addenda Required.
        // 1:enable;0:disable
        public static final int TAG_UPCA_ADDENDA_REQUIRED        = 0x1A010006;
        // Addenda add Separator.
        // 1:enable;0:disable
        public static final int TAG_UPCA_ADDENDA_SEPARATOR       = 0x1A010007;
        // Convert to EAN13.
        // 1:enable;0:disable
        public static final int TAG_UPCA_ADD_COUNTRY_CODE        = 0x1A010008;
}
```

Sample code:

```
// This method is used to check if stripping of
//EAN-13 check digit in decoded data is enabled.
// 1:enable;0:disable
int checkSumDef = XcBarcodeScanner
       .getDecoderTagValue(XCBarcodeTag.TAG_EAN13_CHECK_DIGIT_TRANSMIT);

// Returns the constant of this type with the specified name.
// 1:Enable Check Digit and Output;2:Enable Check Digit and No Output.
int checkDigitDef = XcBarcodeScanner
       .getDecoderTagValue(XCBarcodeTag.TAG_M25_CHECK_DIGIT_MODE);

// This method is used to enable/disable decoding of 2-digit supplemental code for UPC-A.
// 1:enable;0:disable
int twoAddonDef = XcBarcodeScanner
       .getDecoderTagValue(XCBarcodeTag.TAG_UPCA_2CHAR_ADDENDA_ENABLED);
```

Set barcode properties

Use the following API to set barcode properties.

```
void setDecoderTag(int tag, int value);
```

The properties that support queries are defined in the XCBarcodeTag class:

```
public class XCBarcodeTag {
        // Code39
        // Check digit options.
        // 0:Disable Check Digit
        // 1:Enable Check Digit and No Output
        // 2:Enable Check Digit and Output
        public static final int TAG_CODE39_CHECK_DIGIT_MODE      = 0x1A016004;
        // Transmit start/stop char.
        // 1:enable;0:disable
        public static final int TAG_CODE39_START_STOP_TRANSMIT   = 0x1A016007;
        // Code 39 Full ASCII.
                // 1:enable;0:disable
        public static final int TAG_CODE39_FULL_ASCII_ENABLED    = 0x1A016006;
        // Code39 Base32 decode.
        // 1:enable;0:disable
        public static final int TAG_CODE39_BASE32_ENABLED        = 0x1A016008;
        // Maximum length(1-127).
        // The range of values is integers from 1 to 127.
        public static final int TAG_CODE39_MAX_LENGTH            = 0x1A016003;
        // Minimum length(1-127).
        // The range of values is integers from 1 to 127.
        public static final int TAG_CODE39_MIN_LENGTH            = 0x1A016002;

        // DataMatrix
        // With Separators.
        // 1:display;0:hide
        public static final int TAG_DATAMATRIX_SEPARATOR_ENABLED = 0x1A029004;
        // Max out length (0: no limit).
        // An integer greater than or equal to 0, where 0 indicates no restriction.
        public static final int TAG_DATAMATRIX_OUTPUT_MAX_LENGTH = 0x1A029005;

        // EAN-8
        // Transmit check digit.
        // 1:enable;0:disable
        public static final int TAG_EAN8_CHECK_DIGIT_TRANSMIT    = 0x1A012002;
        // 2 Digit Addenda.
        // 1:enable;0:disable
        public static final int TAG_EAN8_2CHAR_ADDENDA_ENABLED   = 0x1A012003;
        // 5 Digit Addenda.
        // 1:enable;0:disable
        public static final int TAG_EAN8_5CHAR_ADDENDA_ENABLED   = 0x1A012004;
        // Addenda Required.
        // 1:enable;0:disable
        public static final int TAG_EAN8_ADDENDA_REQUIRED        = 0x1A012005;
        // Addenda add Separator.
        // 1:enable;0:disable
        public static final int TAG_EAN8_ADDENDA_SEPARATOR       = 0x1A012006;

        // EAN-13
        // Transmit check digit.
        // 1:enable;0:disable
        public static final int TAG_EAN13_CHECK_DIGIT_TRANSMIT   = 0x1A013002;
        // 2 Digit Addenda.
        // 1:enable;0:disable
```

```
        public static final int TAG_EAN13_2CHAR_ADDENDA_ENABLED  = 0x1A013003;
        // 5 Digit Addenda.
        // 1:enable;0:disable
        public static final int TAG_EAN13_5CHAR_ADDENDA_ENABLED  = 0x1A013004;
        // Addenda Required.
        // 1:enable;0:disable
        public static final int TAG_EAN13_ADDENDA_REQUIRED       = 0x1A013005;
        // Addenda add Separator.1:enable;0:disable
        public static final int TAG_EAN13_ADDENDA_SEPARATOR      = 0x1A013006;

        // Matrix 2 of 5
        // Check digit options.
        // 0:Disable Check Digit
        // 1:Enable Check Digit and Output
        // 2:Enable Check Digit and No Output
        public static final int TAG_M25_CHECK_DIGIT_MODE         = 0x1A01C004;

        // UPC-A
        // Transmit check digit.
        // 1:enable;0:disable
        public static final int TAG_UPCA_CHECK_DIGIT_TRANSMIT    = 0x1A010002;
        // Number system digit.
        // 1:enable;0:disable
        public static final int TAG_UPCA_NUMBER_SYSTEM_TRANSMIT  = 0x1A010003;
        // 2 Digit Addenda.
        // 1:enable;0:disable
        public static final int TAG_UPCA_2CHAR_ADDENDA_ENABLED   = 0x1A010004;
        // 5 Digit Addenda.
        // 1:enable;0:disable
        public static final int TAG_UPCA_5CHAR_ADDENDA_ENABLED   = 0x1A010005;
        // Addenda Required.
        // 1:enable;0:disable
        public static final int TAG_UPCA_ADDENDA_REQUIRED        = 0x1A010006;
        // Addenda add Separator.
        // 1:enable;0:disable
        public static final int TAG_UPCA_ADDENDA_SEPARATOR       = 0x1A010007;
        // Convert to EAN13.
        // 1:enable;0:disable
        public static final int TAG_UPCA_ADD_COUNTRY_CODE        = 0x1A010008;
}
```

Sample code:

```
// This method is used to enable/disable stripping of EAN-13 check digit in decoded data.
// 1:enable;0:disable
XcBarcodeScanner.setDecoderTag(XCBarcodeTag.TAG_EAN13_CHECK_DIGIT_TRANSMIT, 1);

// This method is used to set the optional checksum setting of
// Symbologies.Matrix2of5Properties to the decoder.
// 0:Disable Check Digit
// 1:Enable Check Digit and Output
// 2:Enable Check Digit and No Output.
XcBarcodeScanner.setDecoderTag(XCBarcodeTag.TAG_M25_CHECK_DIGIT_MODE, 2);

// This method is used to enable/disable decoding of 2-digit supplemental code for UPC-A.
// 1:enable;0:disable
XcBarcodeScanner.setDecoderTag(XCBarcodeTag.TAG_UPCA_2CHAR_ADDENDA_ENABLED, 0);
```

Get scan trigger mode

Use the following API to get scan trigger mode.

```
String getScanTriggerMode();
```

The properties that support queries are defined in the ScanTriggerMode class:

```
public class ScanTriggerMode {
       public static final String STOP_ON_RELEASE = "SYNC"; //Stop on release
       public static final String STOP_ON_TIMEOUT = "TIMEOUT"; //Stop on timeout
}
```

Sample code:

```
String defMode = XcBarcodeScanner.getScanTriggerMode();
```

Set scan trigger mode

Use the following API to set scan trigger mode.

```
void setScanTriggerMode(String val);
```

The properties that support queries are defined in the ScanTriggerMode class:

```
public class ScanTriggerMode {
       public static final String STOP_ON_RELEASE = "SYNC"; //Stop on release
       public static final String STOP_ON_TIMEOUT = "TIMEOUT"; //Stop on timeout
}
```

Sample code:

```
//set the scan trigger mode to stop on release
XcBarcodeScanner.setScanTriggerMode(ScanTriggerMode.STOP_ON_RELEASE);

//set the scan trigger mode to stop on timeout
XcBarcodeScanner.setScanTriggerMode(ScanTriggerMode.STOP_ON_TIMEOUT);
```

Документация PDF