OneSpan

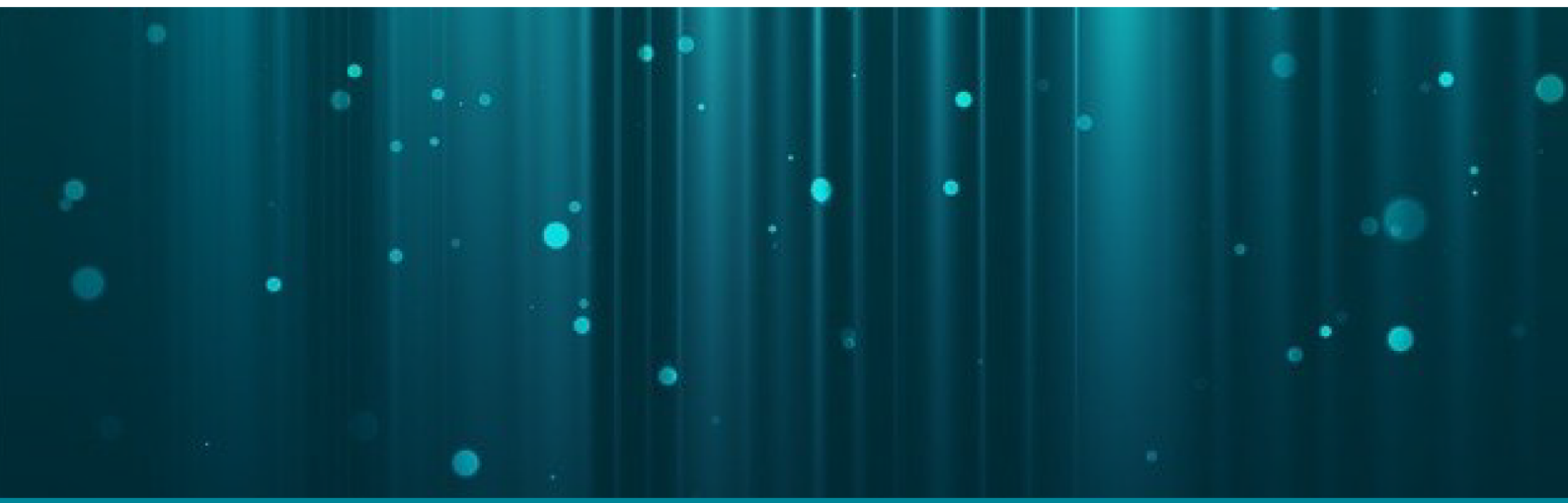


Image Scanner SDK

Integration Guide

Version: 4.31

Copyright Notice

Copyright © 2013–2024 OneSpan North America, Inc. All rights reserved.

Trademarks

OneSpan™, DIGIPASS® and CRONTO® are registered or unregistered trademarks of OneSpan North America Inc., OneSpan NV and/or OneSpan International GmbH (collectively "OneSpan") in the U.S. and other countries.

OneSpan reserves all rights to the trademarks, service marks and logos of OneSpan and its subsidiaries.

All other trademarks or trade names are the property of their respective owners.

Intellectual Property

OneSpan Software, documents and related materials (“Materials”) contain proprietary and confidential information. All title, rights and interest in OneSpan Software and Materials, updates and upgrades thereof, including software rights, copyrights, patent rights, industrial design rights, trade secret rights, sui generis database rights, and all other intellectual and industrial property rights, vest exclusively in OneSpan or its licensors. No OneSpan Software or Materials may be downloaded, copied, transferred, disclosed, reproduced, redistributed, or transmitted in any form or by any means, electronic, mechanical or otherwise, for any commercial or production purpose, except as otherwise marked or when expressly permitted by OneSpan in writing.

Disclaimer

OneSpan accepts no liability for the accuracy, completeness, or timeliness of content, or for the reliability of links to and content of external or third party websites.

OneSpan shall have no liability under any circumstances for any loss, damage, or expense incurred by you, your company, or any third party arising from the use or inability to use OneSpan Software or Materials, or any third party material made available or downloadable. OneSpan will not be liable in relation to any loss/damage caused by modification of these Legal Notices or content.

Reservation

OneSpan reserves the right to modify these Notices and the content at any time. OneSpan likewise reserves the right to withdraw or revoke consent or otherwise prohibit use of the OneSpan Software or Materials if such use does not conform to the terms of any written agreement between OneSpan and you, or other applicable terms that OneSpan publishes from time to time.

Contact us

Visit our website: **https://www.onespan.com**
Resource center: **https://www.onespan.com/resource-center**
Technical support and knowledge base: **https://www.onespan.com/support**

If there is no solution in the knowledge base, contact the company that supplied you with the OneSpan product.

Date: 2024-05-07

Contents

1 Introduction 1

2 What's new in the Image Scanner SDK 2

2.1 Version 4.31.0 3

2.2 Previous versions 4

3 Usage of the Image Scanner SDK 6

3.1 Overview 7

3.2 Image Decoding Method for Android 10

3.3 Functions for Android 11

3.4 Image Decoding Method for iOS 16

3.5 Functions for iOS 17

4 Integrate the Image Scanner SDK 21

4.1 Integrate the Image Scanner SDK with Android 22

4.2 Integrate the Image Scanner SDK with iOS 23

Index 24

Tables

Table 1: Error Codes 9

Table 2: QRCodeScannerSDKConstants Properties for Android 13

Table 3: QRCodeScannerSDKConstants values for iOS 20

Procedures

To use the Image Scanner SDK in your Android project 22

To use the Image Scanner SDK in your iOS project 23

Introduction

1

Welcome to the Image Scanner SDK Integration Guide! This guide describes how to integrate the Image Scanner SDK into any application.

This guide provides information about:

- Functions of the Image Scanner SDK, with parameter and method descriptions
- Integration steps

What's new in the Image Scanner SDK 2

This section provides an overview of changes introduced in the Image Scanner SDK to facilitate the integration of the SDK and provide information on backward compatibility.

2.1 Version 4.31.0 3

2.2 Previous versions 4

2.1 Version 4.31.0

2.1.1 Android

Target API level increased to Android 14 (API 34)

The target version of the SDK has been increased to Android 14 with API level 34.

2.2 Previous versions

2.2.1 Version 4.30.0

Android

Documented workaround for known issue with Samsung Galaxy A23

There is a known issue with the camera not recognizing Cronto images on the Samsung Galaxy A23 and a workaround was documented. For more information, see **Known Issue**.

2.2.2 Version 4.29.0

iOS

Bitcode support has been eliminated

Following the deprecation of Bitcode by Apple, we no longer support and have removed all Bitcode from the SDK framework.

2.2.3 Version 4.28.1

Target API increased to 33

To meet new requirements for apps published on the Google Play Store, the target version of the SDK has been increased to Android 13 with API level 33. This change is needed to avoid APK rejection caused by security issues found with older API versions.

2.2.4 4.28.0

Android

Target API increased to Android 12 (API 31)

To meet new requirements for apps published on the Google Play Store, the target version of the SDK has been increased to Android 12 with API 31 or higher. This change is needed to avoid APK rejection caused by security issues found with the older API versions.

Minimum supported version increased to Android 6 (API 23)

The minimum supported version has been increased to fully support new features and devices. Deprecated code has been replaced and simplified to be compatible with API 23 or higher.

iOS

Increased the minimum supported version to iOS 13

The minimum supported version has been increased to fully support all ARM64 devices and SwiftUI features. Deprecated code has been removed and replaced with code fully supporting iOS 13 or higher.

Fixed internal error conversion issues

Error codes converted between Objective-C and Swift need to have the `NSCustomError` setting implemented to display the correct value. Without this setting, the wrong codes could be displayed from the enum. The setting has been implemented in all the remaining error handling objects.

Change of framework name to *MSSImageScanner*

The framework name has been changed from `ImageScannerSDK.xcframework` to `MSSImageScanner.xcframework`.

2.2.5 Version 4.27.1

iOS

API updates – NSErrors API support for Objective-C added

The Objective-C API points no longer throw an `NSException`. All API points that previously would throw such an `NSException` now require an `NSError` pointer. If an error occurs, it will be attached to the pointer that is provided as a parameter. Refer to the Objective-C sample included in the product package for full code examples.

Usage of the Image Scanner SDK 3

This section contains an overview of the Image Scanner SDK and information about the functions of the SDK.

3.1 Overview 7

3.2 Image Decoding Method for Android 10

3.3 Functions for Android 11

3.4 Image Decoding Method for iOS 16

3.5 Functions for iOS 17

3.1 Overview

The Image Scanner SDK facilitates the Digipass application development: it provides you with the image scanning functionality to capture QR codes and **Cronto**[1] images. The Image Scanner SDK package includes the following:

- `MSSImageScanner.xcframework` for iOS
- `QRCodeScannerSDK.aar` for Android

The Image Scanner SDK can be used on a variety of devices and supports the following platforms:

Android devices:

- Minimum Android 6 (API level 23)
- Target Android 14 (API level 34)

> **NOTE:** The Image Scanner SDK does not support some Doogee devices.

iOS devices:

- iOS 13 or higher
- Swift 5.0 or higher
- Xcode 14 or higher

> **NOTE:** As of OneSpan Mobile Security Suite 4.20.0, the integration of the Image Scanner SDK has changed for iOS. For more information, see **4.2 Integrate the Image Scanner SDK with iOS**.

[1] Specific colorful cryptogram, similar to a QR code; used for visual transaction signing.

3.1.1 Image Requirements

The Image Scanner SDK requires the target image to meet the following requirements:

- The image containing the QR code must be clear and well-focused.
- The QR code should be in the center of the image.
- The photo should not be taken too closely.
- The maximum width and height is 2048 pixels (applies to iOS only).

3.1.2 Exposed APIs

The Image Scanner SDK APIs are exposed as follows:

- Android: The APIs are exposed in the `com.vasco.digipass.sdk.utils.qrcodescanner` package.

3.1.3 Exception Management on Android

When an error occurs on Android, a `QRCodeScannerSDKException` is thrown. This exception consists of an error code and, if the exception is an internal error, the cause of the exception.

3.1.4 Error Codes

Table 1 lists the constants which can be retrieved from the `QRCodeScannerSDKErrorCodes` class.

Table 1: Error Codes

Name	Value	Description
`INTERNAL_ERROR`	–4500	Internal error.
`INPUT_PARAMETER_NULL`	–4501	Call-back / delegate is null.
`CAMERA_NOT_AVAILABLE`	–4502	Camera not present on the device.
`PERMISSION_DENIED`	–4503	Permission to access the camera is not set.

Table 1: Error Codes (continued)

Name	Value	Description
NATIVE_LIBRARY_NOT_LOADED	–4504	Native library has not been loaded (Android only).
INVALID_IMAGE	–4505	Provided Image is invalid and cannot be decoded.
IMAGE_TOO_BIG	–4506	Provided image is too big to process (iOS only)
INVALID_CODE_TYPE	–4507	Provided code type is invalid and cannot be processed.

CAUTION: To be able to access the camera of the device, you need to set the appropriate permissions in your application configuration.

For *Android*, you need to add the *CAMERA* and *VIBRATE* permissions to the `AndroidManifest.xml` file.

3.2 Image Decoding Method for Android

The Android API contains a method to decode and view the results of the static images in the Image Scanner SDK.

3.2.1 Image Decoding Method

The following method decodes the QR code in the image once it is detected:

```
public static QRCodeScannerSDKDecodingResultData decodeImage(Image image, int codeType) throws QRCodeScannerSDKException, LinkageError
```

3.2.2 Obtain Decoding Results

After decoding the QR code in the image, the `QRCodeScannerSDKDecodingResultData` obtains the results of the scan with the following methods:

```
public int getScannedImageFormat()
```

```
public String getScannedImageData()
```

3.3 Functions for Android

The Android API consists of the `QRCodeScannerSDKActivity` class, which manages the behavior of the Image Scanner SDK.

The Android API returns a result code to indicate if a scan has been successful.

3.3.1 Display the scan result

When a code is flashed, the `QRcodeScannerSDKActivity` class ends the scan process, and the activity is closed with the following code:

```
resultCode == RESULT_OK
```

The scan result will be transmitted with the `OUTPUT_RESULT` key.

3.3.2 Cancel the scan process

When the back button is pressed, the `QRCodeScannerSDKActivity` instance is closed with the following code:

```
resultCode == RESULT_CANCELED
```

3.3.3 Exception

If an error occurs, the caught exception is converted to a `QRCodeScannerSDKException` object. The `QRCodeScannerSDKActivity` ends the process and returns the following code:

```
resultCode == RESULT_ERROR
```

The result of the scan will be transmitted with the `OUTPUT_EXCEPTION` key.

3.3.4 `QRCodeScannerSDKConstants for Android`

The `QRCodeScannerSDKConstants` class contains the constants. The following table lists the available constants and parameters.

Properties

Table 2: QRCodeScannerSDKConstants Properties for Android

Property Name	Data Type	Description
Android Extra Parameters		
`EXTRA_VIBRATE`	Boolean	Indicates whether the device must vibrate when a QR code is scanned. Key value: `extra_vibrate` Default value: `true`
`EXTRA_CODE_TYPE`	int	Indicates which type of code can be parsed. Key value: `extra_code_type` Possible values are: • `QRCodeScannerSDKConstants.QR_CODE` • `QRCodeScannerSDKConstants.CRONTO_CODE` • `QRCodeScannerSDKConstants.QR_CODE + QRCodeScannerSDKConstants.CRONTO_CODE` Default value: `QRCodeScannerSDKConstants.QR_CODE + QRCodeScannerSDKConstants.CRONTO_CODE`
`QR_CODE`	int	Indicates that the Image Scanner SDK must parse the image to search QR codes. Key value: `0x01`
`EXTRA_SCANNER_OVERLAY`	Boolean	Indicates that a scanner overlay will be displayed in scan view. Key value: `extra_scanner_overlay` Default value: `false`
`EXTRA_SCANNER_OVERLAY_COLOR`	int	Indicates the color that will be applied in the scanner overlay. Key value: `extra_scanner_overlay_color` Default value: `0x99000000`
`CRONTO_CODE`	int	Indicates that the Image Scanner SDK must parse the image to search Cronto images. Key value: `0x02`
Android Return Parameters		

Table 2: QRCodeScannerSDKConstants Properties for Android (continued)

Property Name	Data Type	Description
OUTPUT_RESULT	String	Result of the scan. Key value: output_result
OUTPUT_EXCEPTION	Exception	Caught exception. Key value: output_exception
OUTPUT_CODE_TYPE	int	Code type result. Key value: output_code_type Possible values are: • QR_CODE • CRONTO_CODE
RESULT_ERROR	int	Indicates whether an exception has been thrown during the scanning process. Key value: RESULT_FIRST_USER + 1

Example

```
01. public void OnScanClicked(View v)
02. {
03.   ..
04.   // We want a vibration feedback after scanning
05.   // Note that the vibration feedback is activated by default
06.   intent.putExtra(QRCodeScannerSDKConstants.EXTRA_VIBRATE, true);
07.
08.   // Indicate which sort of image we want to scan
09.   intent.putExtra(QRCodeScannerSDKConstants.EXTRA_CODE_TYPE,
                      QRCodeScannerSDKConstants.QR_CODE +
                      QRCodeScannerSDKConstants.CRONTO_CODE);
10.
11.   // Launch Image Scanner activity
12.   startActivityForResult(intent, 1);
13. }
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
  switch (resultCode)
  {
    case RESULT_OK:
      ..
      String result = data.getStringExtra
                      (QRCodeScannerSDKConstants.OUTPUT_RESULT);
      int codeType = data.getIntExtra
                      (QRCodeScannerSDKConstants.OUTPUT_CODE_TYPE, 0);

      // Convert the result
      if (codeType == QRCodeScannerSDKConstants.CRONTO_CODE)
      {
        // we have scan a cronto code => convert the hexa string to string
        byte[] tmp = hexaToBytes(result);
        ..
      }
      else if (codeType == QRCodeScannerSDKConstants.QR_CODE)
      {
        // we have scanned a QR code => display directly the result
        resultTextView.setText(result);
        ..
      }
  }
}
```

Known Issue

CAUTION: Known issue with Samsung Galaxy A23

Due to a known issue with the camera on Samsung Galaxy A23, Cronto codes are not decoded properly with the default resolution. The solution to this issue is to increase the resolution using the following code snippet:

```
var imageAnalysisBuilder = ImageAnalysis.Builder().setTargetRotation(rotation)
// Set Resolution property only for Samsung Galaxy A23 model
if(Build.MANUFACTURER.equals("Samsung", true) && Build.MODEL.contains("A23")) {
        imageAnalysisBuilder.setTargetResolution(Size(1080, 1920))
}
imageAnalysisBuilder.build().setAnalyzer(cameraExecutor, QrCodeAnalyzer())
```

3.4 Image Decoding Method for iOS

The iOS API contains a method to decode and view the results of the static images in the Image Scanner SDK.

3.4.1 Image Decoding Method

The following method decodes the QR code in the image once it is detected.

In Objective-C:

```
+ (QRCodeScannerSDKDecodingResultData * _Nullable)decodeImage:(UIImage *)image
ofType:(int)codeType error:(NSError ** _Nullable)error;
```

In Swift:

```
public static func decodeImage(_ image: UIImage, codeType: CodeType) throws ->
QRCodeScannerSDKDecodingResultData
```

3.4.2 Obtain Decoding Results

After decoding the QR code in the image, the `QRCodeScannerSDKDecodingResultData` obtains the results of the scan. The result contains the following fields.

In Objective-C:

```
int codeType;
```

```
NSString *result;
```

In Swift:

```
var codeType: CodeType
```

```
var result: String
```

3.5 Functions for iOS

The main entry is a `UIViewController` returned by the SDK. The following API is available to obtain the instance:

Objective-C:

```
@interface QRCodeScannerSDK : NSObject
+ (UIViewController * _Nullable)getQRCodeScannerSDKViewControllerWithDelegate:
(id<QRCodeScannerSDKDelegate>)delegate
        vibrate:(BOOL)vibrate
        codeType:(int)codeType
        image:(UIImage *_Nullable)image
        error:(NSError **_Nullable)error;
+ (UIViewController * _Nullable)getQRCodeScannerSDKViewControllerWithDelegate:
(id<QRCodeScannerSDKDelegate>)delegate
        vibrate:(BOOL)vibrate
        codeType:(int)codeType
        image:(UIImage *_Nullable)image
        scannerOverlay:(BOOL)scannerOverlay
        scannerOverlayColor:(UIColor *_Nullable)color
        error:(NSError **_Nullable)error;
+ (const NSString *)version;
@end
```

Swift:

```
01. public class QRCodeScannerSDK {
02. public static func getQRCodeScannerSDKViewController(delegate: ScannerDelegate, vibrate: Bool, codeType: CodeType, image: UIImage?) throws -> UIViewController
03. public static func getQRCodeScannerSDKViewController(delegate: ScannerDelegate, vibrate: Bool, codeType: CodeType, image: UIImage?, scannerOverlay: Bool, scannerOverlayColor: UIColor?) throws -> UIViewController
04. public static var version: String
05. }
```

Parameter details:

- `vibrate`: input parameter. The phone will vibrate when a QR code or a **Cronto**[1] image has been flashed, if the corresponding option is configured. This option is available when the `vibrate` input parameter is used.
- `codeType`: input parameter. The Image Scanner SDK can scan either the content of the QR code or the Cronto image, or both. This option is available when the `codeType` input parameter is used. Possible values are:

  Objective-C:

  ```
  QRCodeScannerSDKConstants.QR_CODE
  ```

  ```
  QRCodeScannerSDKConstants.CRONTO_CODE
  ```

  ```
  QRCodeScannerSDKConstants.QR_CODE +
  QRCodeScannerSDKConstants.CRONTO_CODE
  ```

  The default value enables scanning for all types.

  Swift:

  ```
  CodeType.qrCode
  ```

[1]Specific colorful cryptogram, similar to a QR code; used for visual transaction signing.

```
CodeType.crontoCode
```

```
CodeType.all
```

The default value enables scanning for all types.

- `image`: to customize the icon for canceling the scan process.
- `delegate`: protocol to manage the callbacks of the view controller instance returned by the SDK. It contains three methods to manage the QR code scanning process and must be implemented in your application.
- `scannerOverlay`: if set to **YES**, a scanner overlay is added to the scan view. By default this overlay is disabled.
- `scannerOverlayColor`: To customize the color of the overlay. If set to null, the default color (**0x99000000**) is used.

3.5.1 Display the scan result

This method is called when a code has been scanned, with the result of the scan as the input parameter:

Objective-C:

```
(void)qrCodeScannerSDKController:(QRCodeScannerSDKViewController*) controller
didScanResult:(NSString *)result withCodeType:(int)codeType;
```

Swift:

```
func qrCodeScannerSDKController(_ controller: UIViewController, didScan result:
String, codeType: CodeType)
```

3.5.2 Cancel the scan process

When the end user presses the relevant icon, `qrCodeScannerSDKControllerDidCancel` is called, which closes the image scanner:

Objective-C:

```
(void)qrCodeScannerSDKControllerDidCancel:
(QRCodeScannerSDKViewController*)controller;
```

Swift:

```
func qrCodeScannerSDKControllerDidCancel(_ controller: UIViewController)
```

3.5.3 Exception

When an exception is thrown during the scan process, the callback `threwException` is called:

Objective-C:

```
(void)qrCodeScannerSDKController:(QRCodeScannerSDKViewController*) controller
threwException:(QRCodeScannerSDKException *)exception;
```

Swift:

```
func qrCodeScannerSDKController(_ controller: UIViewController, didReceive
error: ScannerError)
```

3.5.4 Example

Sample Applications in Objective-C and Swift are provided along with this package.

3.5.5 QRCodeScannerSDKConstants for iOS

The `QRCodeScannerSDKConstants` file contains the constant, fixed values for the Image Scanner SDK in iOS.

Table 3: `QRCodeScannerSDKConstants` **values for iOS**

Name	Value	Description
`QRCodeScannerSDKConstants_IMAGE_MAX_SIZE`	2048	Indicates the maximum width and height of a decoded image, in pixels.

Integrate the Image Scanner SDK 4

4.1 Integrate the Image Scanner SDK with Android 22

4.2 Integrate the Image Scanner SDK with iOS 23

4.1 Integrate the Image Scanner SDK with Android

This procedure allows you to integrate and use the Image Scanner SDK in your Android project.

▸ To use the Image Scanner SDK in your Android project

1. Copy `QRCodeScannerSDK.aar` from the OneSpan Mobile Security Suite to the `libs` folder.
2. Declare `QrScannerSDKActivity` in your application manifest (refer to the Image Scanner SDK sample for details).

> **NOTE:** To be able to access the camera, you need to set the appropriate permissions in your application configuration. The *CAMERA* and *VIBRATE* permissions must be added to the `AndroidManifest.xml` file.
>
> For applications that target API 23 and run on Android 6.0+, the *CAMERA* permission must be explicitly requested at runtime. Refer to the Android sample for more details.
>
> For applications that target an API earlier than 23 and run on Android 6.0+: If the user first grants permission upon installation and then denies the permission in the device settings, the Image Scanner SDK may not work as expected (as indicated by the Android system).

You are now ready to use the Image Scanner SDK. For more information how to integrate the functions, see **3.3 Functions for Android**.

4.2 Integrate the Image Scanner SDK with iOS

This procedure allows you to integrate and use the Image Scanner SDK in your iOS project.

▸ To use the Image Scanner SDK in your iOS project

1. Link `MSSImageScanner.xcframework` from the OneSpan Mobile Security Suite package to your Xcode project (linked framework and libraries).
2. Link the following libraries from the iOS SDK to your Xcode project with the *Link Binary With Libraries* build phase:
   - `Libiconv`
   - `libc++` (only required if the project is compiled in Objective-C and not in Objective-C++)

NOTE: `XCFramework` is compatible with Xcode 11 and later. We strongly recommend using `XCFramework`, as it facilitates the archiving of projects and it works with simulators.

CAUTION: Your project must be compiled with the `libc++` standard library.

CAUTION: In iOS 10, `NSCameraUsageDescription` is mandatory and must be added to the `.plist` file of the project.

CAUTION: If you use Swift, your project must be linked with the `-ObjC` flag. This should be set in `Other Linker Flags` in the target's build settings. In a pbxproj project file, the raw key is `OTHER_LDFLAGS`.

You are now ready to use the Image Scanner SDK. For more information how to integrate the functions, see **3.5 Functions for iOS**.

Index

A

Android
- decoding results 10
- exceptions 11
- exposed APIs 8
- functions 11
- image decoding 10
- integrate the SDK 22
- QRCodeScannerSDKConstants 11, 20
- QRCodeScannerSDKConstants, example 13
- required permissions 9
- scan process, cancel 11
- scan process, display 11
- supported versions 7

API 17

C

cancel scan process, Android 11
cancel scan process, iOS 19

D

display scan process, Android 11
display scan process, iOS 19

E

error codes, QRCodeScannerSDKErrorCodes class 8
exception management, QRCodeScannerSDKException 8

F

functions
- Android 11
- iOS 17

I

image decoding
- Android 10
- iOS 16

Image requirements 8
integration steps
- Android 22
- iOS 23

iOS
- exceptions 20
- functions 17
- image decoding 16
- integrate the SDK 23
- integration, caution notice 23
- qrCodeScannerSDKController, example 20
- scan process, cancel 19
- scan process, display 19
- supported versions 7
- version 10 integration, caution notice 23

M

methods
Android 10
iOS 16

O

OUTPUT_EXCEPTION key, Android 11
OUTPUT_RESULT key, Android 11

P

package content 7

Q

QRCodeScannerSDK Constants
class 11, 20
QRCodeScannerSDKActivity,
Android 11
QRCodeScan-
nerSDKControllerDidCancel
method, iOS 19
QRCodeScannerSDKErrorCodes
class 8
QRCodeScannerSDKException 8
QRCodeScannerSDKException object,
Android 11

R

results
image decoding 16

S

supported platforms 7

T

threwException callback, iOS 20