

# **ZEBRA SCANNER SDK for iOS DEVELOPER GUIDE**



**ZEBRA SCANNER SDK  
for iOS  
DEVELOPER GUIDE**

MN001834A02

Revision A

December 2015

No part of this publication may be reproduced or used in any form, or by any electrical or mechanical means, without permission in writing from Zebra. This includes electronic or mechanical means, such as photocopying, recording, or information storage and retrieval systems. The material in this manual is subject to change without notice.

The software is provided strictly on an “as is” basis. All software, including firmware, furnished to the user is on a licensed basis. Zebra grants to the user a non-transferable and non-exclusive license to use each software or firmware program delivered hereunder (licensed program). Except as noted below, such license may not be assigned, sublicensed, or otherwise transferred by the user without prior written consent of Zebra. No right to copy a licensed program in whole or in part is granted, except as permitted under copyright law. The user shall not modify, merge, or incorporate any form or portion of a licensed program with other program material, create a derivative work from a licensed program, or use a licensed program in a network without written permission from Zebra. The user agrees to maintain Zebra’s copyright notice on the licensed programs delivered hereunder, and to include the same on any authorized copies it makes, in whole or in part. The user agrees not to decompile, disassemble, decode, or reverse engineer any licensed program delivered to the user or any portion thereof.

Zebra reserves the right to make changes to any software or product to improve reliability, function, or design. Zebra does not assume any product liability arising out of, or in connection with, the application or use of any product, circuit, or application described herein.

No license is granted, either expressly or by implication, estoppel, or otherwise under any Zebra Technologies Corporation, intellectual property rights. An implied license only exists for equipment, circuits, and subsystems contained in Zebra products.

---

## Warranty

For the complete Zebra hardware product warranty statement, go to:

<http://www.zebra.com/warranty>.

## Revision History

Changes to the original manual are listed below:

Change	Date	Description
-01 Rev A	10/2015	Initial Release
-02 Rev A	12/2015	<ul style="list-style-type: none"> <li>- Removed all references to sbtEstablishCommunicationSession:aRawPipeMode and all uses of rawPipeMode;</li> <li>- Removed all references to sbtEventRawData</li> <li>- Removed Raw Data Received Event section</li> <li>- Removed the following line in ISbtSdkApiDelegate - - (void) sbtEventRawData:(NSData*)rawData fromScanner:(int)scannerID;</li> <li>- Removed sbtEventRawData section and all references to sbtReadRawData</li> <li>- Removed the following line: <ul style="list-style-type: none"> <li>- (SBT_RESULT) sbtReadRawData:(unsigned char*)buffer maxLength:(int)length bytesRead:(int*)bytesRead forScanner:(int)scannerID;</li> </ul> </li> <li>- Removed the sbtReadRawData section</li> <li>- Removed all references to sbtWriteRawData</li> <li>- Remove the following line: <ul style="list-style-type: none"> <li>- (SBT_RESULT) sbtWriteRawData:(unsigned char*)buffer maxLength:(int)length bytesWritten:(int*)bytesWritten forScanner:(int)scannerID;</li> </ul> </li> <li>- Removed sbtWriteRawData section.</li> <li>- Updated Beep Control section, table, and code sample</li> <li>- Removed the Turn Off The LED section.</li> <li>- Replaced the entire Turn On the LED section</li> <li>- Added a deprecation statement to sbtBeepControl</li> <li>- Added a deprecation statement to sbtLedControl</li> <li>- Added code to SbtScannerInfo</li> </ul>



# TABLE OF CONTENTS

Warranty .....	ii
Revision History .....	iii

## About This Guide

Introduction .....	ix
Chapter Descriptions .....	ix
Related Documents .....	ix
Notational Conventions .....	x
Service Information .....	xi

## Chapter 1: INTRODUCTION to the ZEBRA SCANNER SDK for iOS

Overview .....	1-1
Supported Scanners .....	1-1
System Requirements .....	1-2

## Chapter 2: ZEBRA SCANNER CONTROL APPLICATION INSTALLATION and CONFIGURATION

Introduction .....	2-1
Installing the Zebra Scanner Control Application .....	2-2
Using the Source Code .....	2-2
Prerequisites .....	2-2
Launching the Zebra Scanner Control Application .....	2-2
From the Apple App Store .....	2-2
Prerequisites .....	2-2
Launching the Zebra Scanner Control Application .....	2-2
Launching the Zebra Scanner Control Application .....	2-3
Configuring the Xcode Project .....	2-7

**Chapter 3: SAMPLE SOURCE CODE**

Introduction .....	3-1
Accessing the SDK .....	3-2
Event Handling .....	3-2
Connectivity Management .....	3-5
Performing Operations .....	3-8

**Chapter 4: ZEBRA SCANNER SDK for iOS API**

Introduction .....	4-1
Application Programming Interface Definition .....	4-2
Constants .....	4-2
Result Codes .....	4-2
Operating Modes .....	4-2
Scanner Modes .....	4-3
Notifications .....	4-3
Opcodes .....	4-3
Scanner Models .....	4-5
LED Codes .....	4-5
Beep Codes .....	4-5
Types .....	4-7
SBT_RESULT .....	4-7
SbtScannerInfo .....	4-7
ISbtSdkApi .....	4-8
SbtSdkFactory .....	4-8
ISbtSdkApiDelegate .....	4-8
Functions .....	4-9
sbtGetVersion .....	4-9
sbtSetDelegate .....	4-9
sbtSetOperationalMode .....	4-9
sbtSubscribeForEvents .....	4-10
sbtUnsubscribeForEvents .....	4-10
sbtGetAvailableScannersList .....	4-10
sbtGetActiveScannersList .....	4-11
sbtEstablishCommunicationSession .....	4-12
sbtTerminateCommunicationSession .....	4-12
sbtEnableAvailableScannersDetection .....	4-12
sbtEnableAutomaticSessionReestablishment .....	4-13
sbtExecuteCommand .....	4-14
sbtLedControl .....	4-15
sbtBeepControl .....	4-16
Notifications .....	4-17
sbtEventScannerAppeared .....	4-17
sbtEventScannerDisappeared .....	4-17
sbtEventCommunicationSessionEstablished .....	4-17
sbtEventCommunicationSessionTerminated .....	4-17
sbtEventBarcode (deprecated) .....	4-18
sbtEventBarcodeData .....	4-18
sbtEventImage .....	4-19
sbtEventVideo .....	4-19
Methods of the "Execute Command" API .....	4-20



DEVICE_RELEASE_TRIGGER .....	4-20
SCAN_DISABLE .....	4-21
SCAN_ENABLE .....	4-21
DEVICE_CAPTURE_BARCODE .....	4-22
DEVICE_CAPTURE_IMAGE .....	4-22
DEVICE_CAPTURE_VIDEO .....	4-23
SET_ACTION .....	4-23
ATTR_GETALL .....	4-24
ATTR_GET .....	4-25
ATTR_GET_OFFSET .....	4-27
ATTR_SET .....	4-29
ATTR_STORE .....	4-30

## Index



# ABOUT THIS GUIDE

---

## Introduction

The *Zebra Scanner SDK for iOS Developer Guide* provides installation and programming information for the Software Developer Kit (SDK) that allows Software Decode based applications for iOS based devices.

---

## Chapter Descriptions

This guide includes the following topics:

- [Chapter 1, INTRODUCTION to the ZEBRA SCANNER SDK for iOS](#) provides an overview of the Zebra Scanner SDK for iOS.
- [Chapter 2, ZEBRA SCANNER CONTROL APPLICATION INSTALLATION and CONFIGURATION](#) describes the process required to install and execute the scanner Zebra Scanner Control application.
- [Chapter 3, SAMPLE SOURCE CODE](#) provides detailed examples that demonstrate how to develop iOS applications using the Zebra Scanner Software Development Kit (SDK).
- [Chapter 4, ZEBRA SCANNER SDK for iOS API](#) defines the API that can be used by external applications to connect remote scanners to a specific iOS device and control connected scanners.

---

## Related Documents

- Zebra Scanner SDK for Android Developer Guide, p/n MN002223Axx.
- RFD8500 RFID Developer Guide, p/n MN002222Axx.
- RFD8500 Quick Start Guide, p/n MN002225Axx.
- RFD8500 Regulatory Guide, p/n MN002062Axx.
- CRDUNIV-RFD8500-1R Three Slot Universal Cradle Charge Only Regulatory Guide, p/n MN002224Axx.
- RFD8500 User Guide, p/n MN002065Axx.

For the latest version of this guide and all guides, go to: [www.zebra.com/support](http://www.zebra.com/support).

---

## Notational Conventions

This document uses the following conventions:

- The prefix SBT is used to reference Zebra Scanner SDK for iOS APIs.
- The abbreviation for Bluetooth is BT.
- SDK refers to the Zebra Scanner SDK for iOS.
- *Italics* are used to highlight chapters, screen names, fields, and sections in this and related documents
- bullets (•) indicate:
  - Action items
  - Lists of alternatives
  - Lists of required steps that are not necessarily sequential
- Sequential lists (e.g., those that describe step-by-step procedures) appear as numbered lists.



**NOTE** This symbol indicates something of special interest or importance to the reader. Failure to read the note does not result in physical harm to the reader, equipment or data.



**CAUTION** This symbol indicates that if this information is ignored, the possibility of data or material damage may occur.



**WARNING!** This symbol indicates that if this information is ignored the possibility that serious personal injury may occur.

---

## Service Information

If you have a problem using the equipment, contact your facility's technical or systems support. If there is a problem with the equipment, they contact the Zebra Technologies Global Customer Support Center at: <http://www.zebra.com/support>.

When contacting Zebra support, please have the following information available:

- Product name
- Version number

Zebra responds to calls by e-mail, telephone or fax within the time limits set forth in support agreements.

If your problem cannot be solved by Zebra support, you may need to return your equipment for servicing and will be given specific directions. Zebra is not responsible for any damages incurred during shipment if the approved shipping container is not used. Shipping the units improperly can possibly void the warranty.

If you purchased your business product from a Zebra business partner, contact that business partner for support.



# Chapter 1 INTRODUCTION to the ZEBRA SCANNER SDK for iOS

---

## Overview

The Zebra Scanner SDK for iOS is provided as a static library that can be linked to a custom iOS application. The static library provides an Application Programming Interface (API) to provide the communication facilities required to operate a supported Zebra scanner on the iOS platform. The SDK provides the capability to manage connections, configure connected scanners, perform various operations with connected scanners, and to obtain information about connected scanners.

## Supported Scanners

The following MFi (Made for iPhone/iPod/iPad) Zebra scanners are currently supported by the iOS SDK.

**Table 1-1** *Supported Zebra Scanners*

Zebra Scanner	Configuration
CS4070	Bluetooth SSI Profile mode
RFD8500	Bluetooth MFi Server mode, with primary mode set to MFI SSI



**NOTE** To configure a device in the mode specified, refer to the appropriate Product Reference Guide, User Guide, or Integration Guide.

---

## System Requirements

The following system requirements are necessary in order to develop, test, and execute applications using the Zebra Scanner SDK for iOS:

- Xcode 6, or later
- iOS SDK 8.0, or later
- Zebra Scanner SDK for iOS
- iPhone or iPod Touch running iOS 8.0 or later.



# Chapter 2 ZEBRA SCANNER CONTROL APPLICATION INSTALLATION and CONFIGURATION

---

## Introduction

This chapter describes the process required to install and execute the Zebra Scanner Control application that is provided with the Zebra Scanner SDK for iOS.

✓ **NOTE** The purpose of the scanner Zebra Scanner Control application is to demonstrate the various capabilities of the SDK library. It is not intended for production use purposes.

---

## Installing the Zebra Scanner Control Application

### Using the Source Code

#### Prerequisites

- OS X.
- Xcode. Download Xcode from the [Mac App Store](#).
- iOS 8, or higher, installed.
- An iOS Developer account with Apple.
- A provisioned iOS device.

#### Launching the Zebra Scanner Control Application

To launch the application on the device:

1. Connect a provisioned iOS device to your Mac.
2. Open the *ScannerSDKApp.xcodeproj* project file.
3. In the project navigator, choose your device from the *Scheme* toolbar menu.

✓ **NOTE** If your iOS device is listed as an ineligible device, fix the issue before continuing. The version of iOS that is installed on the device must match the deployment target set in the project.

4. Click the **Run** button. Xcode installs the application on the device.
5. If a warning message appears that states *No matching provisioning profile found*, click the **Fix Issue** button. Xcode needs to add the device to the team provisioning profile before it can launch the application on the device.
6. If a prompt appears that asks whether or not *codesign* may sign the application using a key in your *keychain*, click **Always Allow**.
7. The application launches on your device.

### From the Apple App Store

#### Prerequisites

- An iOS device with iOS 8, or higher, installed.

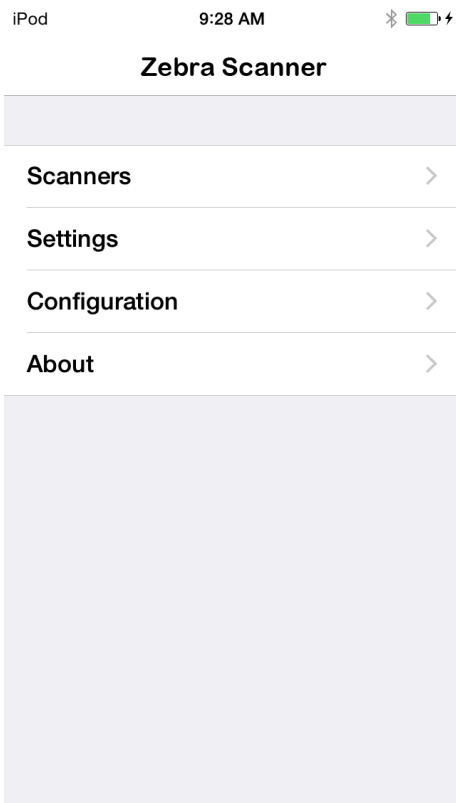
#### Launching the Zebra Scanner Control Application

To launch the scanner application on the device from the Apple App Store:

1. Launch the App Store application on your iOS device.
2. Search for the *Zebra Scanner* application, or *Zebra Scanner Control* application by Zebra Technologies.
3. Tap the **Get** button next to the application, then tap the **Install** button.

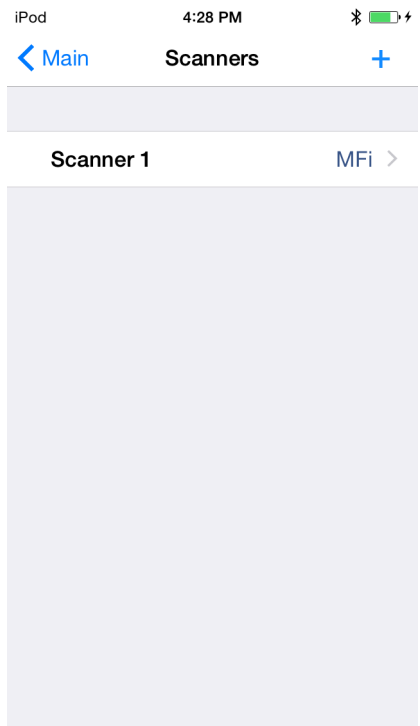
## Launching the Zebra Scanner Control Application

1. Before launching the application, set the Zebra scanner to the appropriate configuration. See [Supported Scanners on page 1-1](#) to determine the required configuration needed for your Zebra scanner.
2. Enable Bluetooth on your iOS device and pair it with the Bluetooth Zebra scanner, if it is not paired already. Refer to the relevant Product Reference Guide, User Guide, or Integration Guide for instructions on how to pair the device.
3. Launch the application by selecting the Zebra Scanner Control application from your device's home screen. The following screen displays:



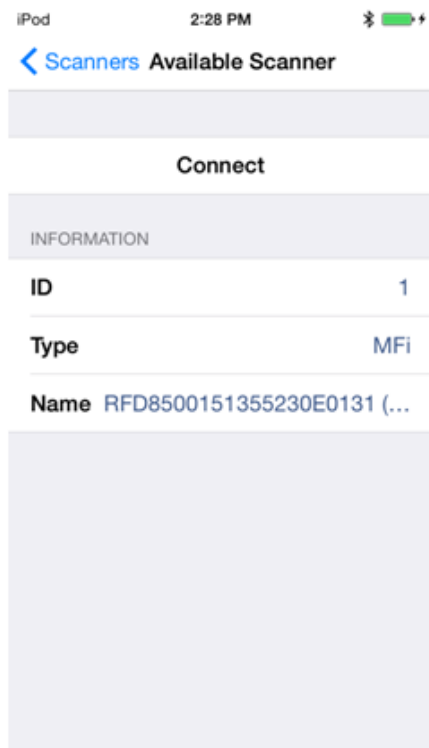
**Figure 2-1** *Scanner Application - Initial Screen*

4. Select *Scanners* from the menu. The following screen displays:



**Figure 2-2** *Figure 2 - Scanner Application - Scanners Screen*

5. Select the device from the left panel, and then tap the **Connect** button from the *Available Scanner* menu. The following screen displays:



**Figure 2-3** *Scanner Application - Available Scanner Screen*

6. Once connected, the *Active Scanner* menu displays. This menu is used to select various options including *Info*, *Barcode*, and *Settings*.

a. Select the *Info* tab to view the *Scanner ID*, *Type*, *Name*, and *Auto Reconnection* state. This tab also contains functionality to control the *Beeper* and *LED*. The following screen displays:

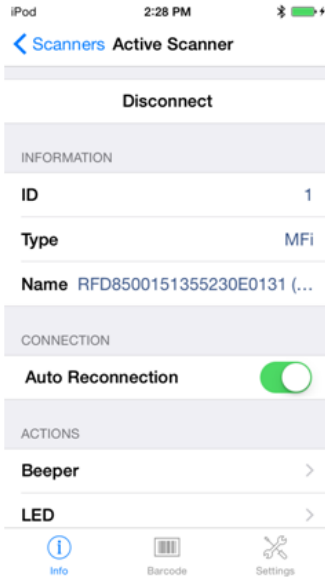


Figure 2-4 Scanner Application - Active Scanner Screen

b. Select the *Barcode* tab to view scanned bar codes, and to pull/release the scanner trigger. The following screen displays:

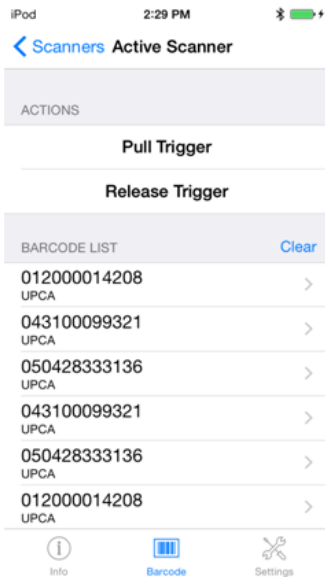


Figure 2-5 Scanner Application - Barcode screen

- c. Select the *Settings* tab to select supported Symbologies, configure the beeper volume and frequency, and to enable/disable scanning. The following screen displays:

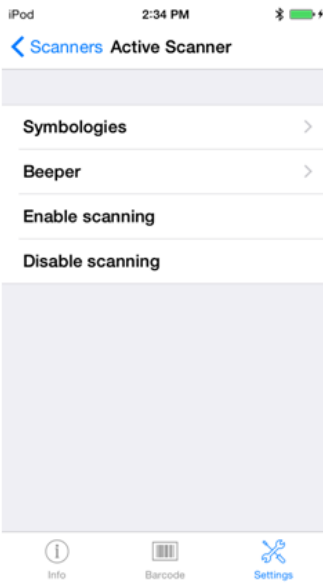


Figure 2-6 Scanner Application - Settings Screen

- i. To control which symbologies the scanner reads, select the *Symbologies* option. An on/off control switch appears next to each bar code type displayed in the menu.

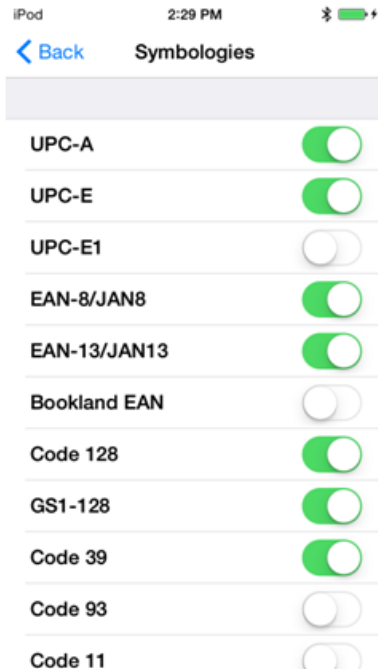


Figure 2-7 Scanner Application - Symbologies Screen

- ✓ **NOTE** Custom symbology settings can be configured via 123Scan<sup>2</sup>. When setting the Inter Character Delay for HID only, set the delay in milliseconds between emulated keystrokes. When pairing with an Android device, set the Inter Character Delay to 70 ms to avoid data loss.

## Configuring the Xcode Project

The Zebra Scanner SDK for iOS consists of a static library that can be linked with an external iOS application and a set of necessary header files.

To add the SDK:

1. Create a new Xcode project and save it to a new project folder.
2. Copy the `symbolbt-sdk` folder provided by Zebra Technologies into the new project folder. This folder contains the `libsymbolbt-sdk.a` file and an `include` folder containing SDK header files.
3. Open the new project in Xcode, and select your project in the file navigator sidebar.
4. Configure the Xcode project to support one or more external accessory communication protocols through the `UISupportedExternalAccessoryProtocols` key in your application *Info.plist* file or via the Info tab of your project settings.

**Table 2-1** *Communication Protocols*

Communication Protocol	Zebra Scanner
<code>com.motorolasolutions.CS4070_ssi</code>	CS4070
<code>com.zebra.scanner.SSI</code> <code>com.motorolasolutions.scanner</code>	RFD8500

5. In order to configure your application to communicate with Bluetooth scanners in a background mode, configure your Xcode project to specify the background modes that your application supports using the `UIBackgroundModes` key in your application *Info.plist* file or via the Info tab of your project settings.
6. Select your *Target*, and then its *Build Phases* tab.
7. Expand the *Link Binary With Libraries* item.
8. Add the following frameworks by clicking the **+** button:
  - `libsymbolbt-sdk.a`
  - `ExternalAccessory`
  - `CoreBluetooth`.
9. Select the *Build Settings* tab.
10. Navigate to the *Search Paths* section of the *Build Settings*.
11. Set *User Header Search Paths* to `$(SRCROOT)/symbolbt-sdk/include/`.
12. Your project is now configured to use the Zebra Scanner SDK for iOS.





# Chapter 3 SAMPLE SOURCE CODE

---

## Introduction

This chapter provides detailed examples that demonstrate how to develop iOS applications using the Zebra Scanner Software Development Kit (SDK).

---

## Accessing the SDK

Create an instance of the Zebra Scanner SDK for iOS API inside of a class that conforms to the `ISbtSdkApiDelegate`. All available API functions are defined by the `ISbtSdkApi` Objective-C protocol. A single shared instance of an API object that implements the `ISbtSdkApi` can be obtained via `createSbtSdkApiInstance` method of the `SbtSdkFactory` class.

### Code Snippet - Obtain Instance of API

```
// ....
#import "SbtSdkFactory.h"

// ....

// Get instance to the Zebra Scanner SDK API
id <ISbtSdkApi> apiInstance = [SbtSdkFactory createSbtSdkApiInstance];

// Get the SDK version string
NSString *version = [apiInstance sbtGetVersion];

NSLog(@"Zebra SDK version: %@\n", version);
```

---

## Event Handling

The SDK supports a set of asynchronous notifications to inform the application about scanner related events. This includes connectivity related events (i.e., appearance of a scanner), and scanner action events (i.e., received bar code data). All supported callbacks are defined by the `ISbtSdkApiDelegate` Objective-C protocol.

In order to receive asynchronous notifications from the SDK, the application performs the following steps:

1. Create an object that implements the `ISbtSdkApiDelegate`.

### Code Snippet - Create A Class That Implements `ISbtSdkApiDelegate`

```
// Definition of a class that implements the ISbtSdkApiDelegate protocol
@interface EventReceiver : NSObject <ISbtSdkApiDelegate> {

// TODO: variables

}

// TODO: methods definition
```

2. Register the created object as notification receiver via the `sbtSetDelegate` function.

### Code Snippet - Set Delegate Of API Instance To Object That Implements `ISbtSdkApiDelegate`

```
// Registration of the callback interface with SDK
EventReceiver *eventListener = [[EventReceiver alloc] init];

[apiInstance sbtSetDelegate:eventListener];
```

3. Subscribe for specific asynchronous events using the `sbtSubscribeForEvents` method to specify which events should be reported by the SDK.

Valid notification/event mask values include:

- `SBT_EVENT_BARCODE`
- `SBT_EVENT_IMAGE`
- `SBT_EVENT_VIDEO`
- `SBT_EVENT_SCANNER_APPEARANCE`
- `SBT_EVENT_SCANNER_DISAPPEARANCE`
- `SBT_EVENT_SESSION_ESTABLISHMENT`
- `SBT_EVENT_SESSION_TERMINATION`
- `SBT_EVENT_RAW_DATA`.

#### Code Snippet - Subscribe To Scanner Events

```
// Subscribe to scanner appearance/disappearance, session establishment/termination,
// barcode, and image & video event notifications.
[apiInstance sbtSubscribeForEvents:SBT_EVENT_SCANNER_APPEARANCE |
SBT_EVENT_SCANNER_DISAPPEARANCE | SBT_EVENT_SESSION_ESTABLISHMENT |
SBT_EVENT_SESSION_TERMINATION | SBT_EVENT_BARCODE | SBT_EVENT_IMAGE |
SBT_EVENT_VIDEO];
```

#### Events

If an object is registered as a notification receiver, the SDK calls the corresponding method of the registered object when a particular event occurs and the application is subscribed for events of this type.

#### ***Scanner Appeared Event***

This event occurs when the presence of a scanner appears.

#### Code Snippet - Scanner Appeared Event

```
- (void)sbtEventScannerAppeared:(SbtScannerInfo*)availableScanner
{
    // TODO: Handle event
}
```

#### ***Scanner Disappeared Event***

This event occurs when a scanner is no longer present.

#### Code Snippet - Scanner Disappeared Event

```
- (void) sbtEventScannerDisappeared:(int)scannerID
{
    // TODO: Handle event
}
```

***Communication Session Established Event***

This event occurs when communication is established with a scanner.

**Code Snippet - Scanner Session Established Event**

```
- (void)
sbtEventCommunicationSessionEstablished:(SbtScannerInfo*)activeScanner
{
    // TODO: Handle event
}

```

***Communication Session Terminated Event***

This event occurs when communication with a scanner is terminated.

**Code Snippet - Scanner Session Terminated Event**

```
- (void) sbtEventCommunicationSessionTerminated:(int)scannerID
{
    // TODO: Handle event
}

```

***Bar Code Information Received Event***

This event occurs when bar code data is read and received.

**Code Snippet - Bar Code Data Received Event**

```
- (void) sbtEventBarcodeData:(NSData *)barcodeData
barcodeType:(int)barcodeType fromScanner:(int)scannerID
{
    // TODO: Handle event
}

```

***Image Data Received Event***

This event occurs when image data is received.

**Code Snippet - Image Data Received Event**

```
- (void) sbtEventImage:(NSData*)imageData fromScanner:(int)scannerID
{
    // TODO: Handle event
}

```

***Video Data Received Event***

This event occurs when video data is received.

**Code Snippet - Video Data Received Event**

```
- (void) sbtEventVideo:(NSData*)videoFrame fromScanner:(int)scannerID
{
    // TODO: Handle event
}

```

## Connectivity Management

The SDK identifies scanners as "available" and "active". An "available" scanner is a scanner that is connected to the iOS device via Bluetooth, but has not yet established a communication session with the SDK. An "active" scanner is a scanner that is both connected to the iOS device via Bluetooth, and has established a communication session with the SDK.

The SDK supports simultaneous interaction with multiple scanners. To distinguish between various scanners, the SDK assigns a unique integer identifier for each scanner when it becomes available for the first time.

### Set Operation Mode

Zebra Scanner SDK for iOS is designed to support interaction with scanners operating in BT MFi or BT LE mode. The SDK shall be configured to enable communication with a particular type of scanner by setting the operation mode.

Use the `sbtSetOperationalMode` function to set the required operational mode of the scanners to interface with.

Valid inputs include:

- `SBT_OPMODE_MFI`
- `SBT_OPMODE_BTLE`
- `SBT_OPMODE_ALL`.

### Code Snippet - Set Operational Mode

```
// ....

// Set operational mode to all so that SDK can interface with scanners
// operating in MFI or BTLE mode.
[apiInstance sbtSetOperationalMode:SBT_OPMODE_ALL];
```

### Enable Scanner Detection

The SDK supports automatic detection of appearance and disappearance of available scanners. When the "Available scanners detection" option is enabled the SDK updates its internal list of available scanners and deliver a corresponding asynchronous notification once it detects connection or disconnection of a particular scanner to the iOS device via Bluetooth. If the option is disabled, the SDK updates its internal list of available scanners only when requested by the application via the `sbtGetAvailableScannersList`.

Use the `sbtEnableAvailableScannersDetection` method to actively detect appearance and disappearance of scanners.

### Code Snippet - Enable Auto Detection

```
// Actively detect appearance/disappearance of scanners
[apiInstance sbtEnableAvailableScannersDetection:YES];
```

### Get Available and Active Scanners

The SDK maintains an internal list of active and available scanners. The following example demonstrates how to receive a list of available and active scanners from the SDK.

**Code Snippet - Get Available And Active Scanners**

```

// ....

// Allocate an array for storage of a list of available scanners
NSMutableArray *availableScanners = [[NSMutableArray alloc] init];

// Allocate an array for storage of a list of active scanners
NSMutableArray *activeScanners = [[NSMutableArray alloc] init];

// Retrieve a list of available scanners
[apiInstance sbtGetAvailableScanners:&availableScanners];

// Retrieve a list of active scanners
[apiInstance sbtGetActiveScanners:&activeScanners];

// Merge the available and active scanners into a single list
NSMutableArray *allScanners = [[NSMutableArray alloc] init];
[allScanners addObjectsFromArray:availableScanners];
[allScanners addObjectsFromArray:activeScanners];

// Print information for each available and active scanner
for (SbtScannerInfo *info in allScanners)
{
    NSLog(@"Scanner is %@: ID = %d name = %@", (([info isActive] == YES) ?
@"active" : @"available"), [info getScannerId], [info getScannerName]);
}

```

**Connect to a Scanner**

You can establish a connection with an available scanner by providing the scanner ID to the SDK's `sbtEstablishCommunicationSession` method.

**Code Snippet - Connect To A Scanner**

```

// ....

// Attempt to establish a connection with the scanner device that has an ID = 3.
SBT_RESULT result = [apiInstance sbtEstablishCommunicationSession:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process successful connection
    NSLog(@"Connection to scanner ID %d successful",scannerId);
}
else
{
    // TODO: Process error (unsuccessful connection)
    NSLog(@"Failed to establish a connection with scanner ID %d",scannerId);
}

```

## Disconnect from a Scanner

You can disconnect from an active scanner using the SDK's `sbtTerminateCommunicationSession` method.

### Code Snippet - Disconnect From A Scanner

```
// ....

// Attempt to disconnect from the scanner device that has an ID = 3.
SBT_RESULT result = [apiInstance sbtTerminateCommunicationSession:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process successful disconnection
    NSLog(@"Disconnect from scanner ID %d successful", scannerId);
}
else
{
    // TODO: Process error (unsuccessful connection)
    NSLog(@"Failed to disconnect from scanner ID %d", scannerId);
}
```

## Automatically Reconnect to Scanner

The SDK supports "Automatic Session Reestablishment" option is used to automatically reconnect to a scanner that unexpectedly disappeared. If the "Automatic Session Reestablishment" option is enabled for a specific scanner, the SDK automatically attempts to reestablish a connection with the scanner when it becomes available again. The option has no effect if the application has intentionally terminated a communication session with the scanner via the `sbtTerminateCommunicationSession` function.

Use the `sbtEnableAutomaticSessionReestablishment` function to enable or disable automatic reconnection for a specific scanner.

### Code Snippet - Enable Automatic Reconnection

```
// ....

// Set the automatic session reestablishment option for scanner ID 3
SBT_RESULT result = [apiInstance sbtEnableAutomaticSessionReestablishment:YES
forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Option successfully set
    NSLog(@"Automatic Session Reestablishment for scanner ID %d has been set
successfully", scannerId);
}
else
{
    // TODO: Error (option was not set)
    NSLog(@"Automatic Session Reestablishment for scanner ID %d could not be
set", scannerId);
}
```

## Performing Operations

The Zebra Scanner iOS SDK provides the capability to perform various scanner operations such as:

- Adjusting beeper settings
- Turning the LED on and off
- Enabling and disabling scanning
- Enabling and disabling supported symbologies
- Pulling and releasing the trigger.

### Beeper

#### *Get Beeper Settings*

The get attribute command (SBT\_RSM\_ATTR\_GET) is used to retrieve the beeper volume and frequency attribute values. The attribute values are returned from the scanner as an XML formatted string.

**Table 3-1** *Beeper Setting Attributes*

Attribute Description	Attribute Name
Beeper Volume	RMD_ATTR_BEEPER_VOLUME
Beeper Frequency	RMD_ATTR_BEEPER_FREQUENCY

The following code example demonstrates how to retrieve the scanner's current beeper volume and frequency settings.

#### Code Snippet - Retrieve Beeper Settings

```
// ....

// Create XML string to request beeper volume and frequency settings for
scanner ID 3
NSString *xmlInput = [NSString stringWithFormat:@"
<inArgs><scannerID>%d</scannerID><cmdArgs><arg-xml><attrib_list>%d,%d</attrib_lis
t></arg-xml></cmdArgs></inArgs>", 3, RMD_ATTR_BEEPER_VOLUME,
RMD_ATTR_BEEPER_FREQUENCY];

NSMutableString *xmlResponse = [[NSMutableString] alloc] init];

// Retrieve beeper volume and frequency settings from Scanner ID 3
SBT_RESULT result = [apiInstance sbtExecuteCommand:SBT_RSM_ATTR_GET
aInXML:xmlInput aOutXML:&xmlResponse forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process xml string containing requested beeper settings
    NSLog(@"Retrieved beeper settings from scanner ID %d:
%@", scannerId, xmlResponse);
}
else
{
    // TODO: Process error
    NSLog(@"Failed to retrieve beeper settings from scanner ID %d", scannerId);
}
```



### ***Set Beeper Volume***

The set attribute command (SBT\_RSM\_ATTR\_SET) is used to set the value of the beeper volume attribute (RMD\_ATTR\_BEEPER\_VOLUME). [Table 3-2](#) includes valid volume attribute values.

**Table 3-2** *Beeper Volume Attribute Values*

<b>Beeper Volume Attribute Values</b>
RMD_ATTR_VALUE_BEEPER_VOLUME_LOW
RMD_ATTR_VALUE_BEEPER_VOLUME_MEDIUM
RMD_ATTR_VALUE_BEEPER_VOLUME_HIGH

The following code example demonstrates how to set the scanner's beeper volume;

#### **Code Snippet - Set The Beeper Volume**

```
// ....

// Create XML string to set the beeper volume on scanner ID 3 to LOW.
NSString *xmlInput = [NSString stringWithFormat:@"
<inArgs><scannerID>%d</scannerID><cmdArgs><arg-xml><attrib_list><attribute><id>%d
</id><datatype>B</datatype><value>%d</value></attribute></attrib_list></arg-xml><
/cmdArgs></inArgs>", 3, RMD_ATTR_BEEPER_VOLUME, RMD_ATTR_VALUE_BEEPER_VOLUME_LOW];

// Command Scanner ID 3 to set beeper volume to LOW.
SBT_RESULT result = [apiInstance sbtExecuteCommand:SBT_RSM_ATTR_SET
aInXML:xmlInput aOutXML:nil forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process successful operation
    NSLog(@"Successfully updated beeper settings for scanner ID %d:
%@", scannerId, xmlResponse);
}
else
{
    // TODO: Process error
    NSLog(@"Failed to update beeper settings from scanner ID %d", scannerId);
}
```

### ***Set Beeper Frequency***

The set attribute command (SBT\_RSM\_ATTR\_SET) is used to set the value of the beeper frequency attribute (RMD\_ATTR\_BEEPER\_FREQUENCY). [Table 3-3](#) includes valid frequency attribute values.

**Table 3-3** *Beeper Frequency Attribute Values*

<b>Beeper Frequency Attribute Values</b>
RMD_ATTR_VALUE_BEEPER_FREQ_LOW
RMD_ATTR_VALUE_BEEPER_FREQ_MEDIUM
RMD_ATTR_VALUE_BEEPER_FREQ_HIGH

The following code example demonstrates how to set the scanner's beeper frequency.

### Code Snippet - Set The Beeper Frequency

```
// ....

// Create XML string to set the beeper frequency on scanner ID 3 to HIGH.
NSString *xmlInput = [NSString stringWithFormat:@"
<inArgs><scannerID>%d</scannerID><cmdArgs><arg-xml><attrib_list><attribute><id>%d
</id><datatype>B</datatype><value>%d</value></attribute></attrib_list></arg-xml><
/cmdArgs></inArgs>", 3, RMD_ATTR_BEEPER_FREQUENCY,
RMD_ATTR_VALUE_BEEPER_FREQ_HIGH];

// Command Scanner ID 3 to set beeper frequency to HIGH.
SBT_RESULT result = [apiInstance sbtExecuteCommand:SBT_RSM_ATTR_SET
aInXML:xmlInput aOutXML:nil forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process successful operation
    NSLog(@"Successfully updated beeper settings for scanner ID %d:
%@", scannerId, xmlResponse);
}
else
{
    // TODO: Process error
    NSLog(@"Failed to update beeper settings from scanner ID %d", scannerId);
}
}
```

### Beep Control

Command the scanner's beeper to emit sound by using the `sbtExecuteCommand` API. Use command op code `SBT_SET_ACTION` and an attribute value shown in [Table 3-4](#) below:

**Table 3-4** *Beeper Attribute Values*

Description	Attribute Values
One high short beep	RMD_ATTR_VALUE_ACTION_HIGH_SHORT_BEEP_1
Two high short beeps	RMD_ATTR_VALUE_ACTION_HIGH_SHORT_BEEP_2
Three high short beeps	RMD_ATTR_VALUE_ACTION_HIGH_SHORT_BEEP_3
Four high short beeps	RMD_ATTR_VALUE_ACTION_HIGH_SHORT_BEEP_4
Five high short beeps	RMD_ATTR_VALUE_ACTION_HIGH_SHORT_BEEP_5
One low short beep	RMD_ATTR_VALUE_ACTION_LOW_SHORT_BEEP_1
Two low short beeps	RMD_ATTR_VALUE_ACTION_LOW_SHORT_BEEP_2
Three low short beeps	RMD_ATTR_VALUE_ACTION_LOW_SHORT_BEEP_3
Four low short beeps	RMD_ATTR_VALUE_ACTION_LOW_SHORT_BEEP_4
Five low short beeps	RMD_ATTR_VALUE_ACTION_LOW_SHORT_BEEP_5

**Table 3-4** *Beeper Attribute Values (Continued)*

Description	Attribute Values
One high long beep	RMD_ATTR_VALUE_ACTION_HIGH_LONG_BEEP_1
Two high long beeps	RMD_ATTR_VALUE_ACTION_HIGH_LONG_BEEP_2
Three high long beeps	RMD_ATTR_VALUE_ACTION_HIGH_LONG_BEEP_3
Four high long beeps	RMD_ATTR_VALUE_ACTION_HIGH_LONG_BEEP_4
Five high long beeps	RMD_ATTR_VALUE_ACTION_HIGH_LONG_BEEP_5
One low long beep	RMD_ATTR_VALUE_ACTION_LOW_LONG_BEEP_1
Two low long beeps	RMD_ATTR_VALUE_ACTION_LOW_LONG_BEEP_2
Three low long beeps	RMD_ATTR_VALUE_ACTION_LOW_LONG_BEEP_3
Four low long beeps	RMD_ATTR_VALUE_ACTION_LOW_LONG_BEEP_4
Five low long beeps	RMD_ATTR_VALUE_ACTION_LOW_LONG_BEEP_5
Fast warble beep	RMD_ATTR_VALUE_ACTION_FAST_WARBLE_BEEP
Slow warble beep	RMD_ATTR_VALUE_ACTION_SLOW_WARBLE_BEEP
High-low beep	RMD_ATTR_VALUE_ACTION_HIGH_LOW_BEEP
Low-high beep	RMD_ATTR_VALUE_ACTION_LOW_HIGH_BEEP
High-low-high beep	RMD_ATTR_VALUE_ACTION_HIGH_LOW_HIGH_BEEP
Low-high-low beep	RMD_ATTR_VALUE_ACTION_LOW_HIGH_LOW_BEEP

The following code example demonstrates how to command the scanner's beeper to emit a high short beep sound.

#### Code Snippet - Beeper Control

```
// ....
#import "RMDAttributes.h"

/ ....

// Specify the ID of the scanner to control
int scannerID = 1;

// Specify the scanner ID and attribute value of the beeper command in the XML
NSString *inXML = [NSString
stringWithFormat:@"<inArgs><scannerID>%d</scannerID><cmdArgs><arg-int>%d</arg-int>
<</cmdArgs></inArgs>", scannerID, RMD_ATTR_VALUE_ACTION_HIGH_SHORT_BEEP_1];

// Issue the beep command
[scannerSdkApi sbtExecuteCommand:SBT_SET_ACTION aInXML:inXML aOutXML:nil
forScanner:scannerID];
```

## LED

### **Control the** LED

Control the scanner's LED by using the `sbtExecuteCommand` API. Use command op code `SBT_SET_ACTION` and an attribute value shown in [Table 3-5](#) below.

**Table 3-5** LED Attribute Values

Description	Attribute Value
Turn Green LED off	RMD_ATTR_VALUE_ACTION_LED_GREEN_OFF
Turn Green LED on	RMD_ATTR_VALUE_ACTION_LED_GREEN_ON
Turn Yellow LED on	RMD_ATTR_VALUE_ACTION_LED_YELLOW_ON
Turn Yellow LED off	RMD_ATTR_VALUE_ACTION_LED_YELLOW_OFF
Turn Red LED on	RMD_ATTR_VALUE_ACTION_LED_RED_ON
Turn Red LED off	RMD_ATTR_VALUE_ACTION_LED_RED_OFF

The following code example demonstrates how to command the scanner's LED to illuminate green.

### **Code Snippet - LED Control**

```
// ....
#import "RMDAttributes.h"

// ....

// Specify the ID of the scanner to control
int scannerID = 1;

// Specify the scanner ID and attribute value the LED command in the XML
NSString *inXML = [NSString
stringWithFormat:@"<inArgs><scannerID>%d</scannerID><cmdArgs><arg-int>%d</arg-int></cmdArgs></inArgs>", scannerID, RMD_ATTR_VALUE_ACTION_LED_GREEN_ON];

// Issue the LED command
[scannerSdkApi sbtExecuteCommand:SBT_SET_ACTION aInXML:inXML aOutXML:nil
forScanner:scannerID];
```

## Scan Capability

### **Enable Scanning**

The scan enable command (SBT\_DEVICE\_SCAN\_ENABLE) is used to enable the device's scanning capability. The following code example demonstrates how to enable the scan capability.

#### **Code Snippet - Enable Scanning Capability**

```
// ....

// Create XML string to enable scanning on scanner ID 3
NSString *xmlInput = [NSString stringWithFormat:@"
<inArgs><scannerID>%d</scannerID></inArgs>", scannerId];

// Command scanner ID 3 to enable scanning
SBT_RESULT result = [apiInstance sbtExecuteCommand:SBT_DEVICE_SCAN_ENABLE
aInXML:xmlInput aOutXML:nil forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process successful operation
    NSLog(@"Successfully enabled scanning on scanner ID %d", scannerId);
}
else
{
    // TODO: Process error
    NSLog(@"Failed to enable scanning on scanner ID %d", scannerId);
}
```

### **Disable Scanning**

The scan disable command (SBT\_DEVICE\_SCAN\_DISABLE) is used to disable the device's scanning capability.

The following code example demonstrates how to disable the scan capability.

#### **Code Snippet - Disable Scanning Capability**

```
// ....

// Create XML string to disable scanning on scanner ID 3
NSString *xmlInput = [NSString stringWithFormat:@"
<inArgs><scannerID>%d</scannerID></inArgs>", scannerId];

// Command scanner ID 3 to disable scanning
SBT_RESULT result = [apiInstance sbtExecuteCommand:SBT_DEVICE_SCAN_DISABLE
aInXML:xmlInput aOutXML:nil forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process successful operation
    NSLog(@"Successfully disabled scanning on scanner ID %d", scannerId);
}
else
{
    // TODO: Process error
    NSLog(@"Failed to disable scanning on scanner ID %d", scannerId);
}
```

## Symbologies

### Get Supported Symbology IDs

The get all symbologies command (`SBT_RSM_ATTR_GETALL`) is used to retrieve the scanner's support symbologies. The supported symbology attribute values are returned from the scanner as an XML formatted string. [Table 3-6](#) includes valid symbology attribute values.

**Table 3-6** Symbology Attribute Values

Description	Attribute Values
UPC-A	RMD_ATTR_SYM_UPC_A
UPC-E	RMD_ATTR_SYM_UPC_E
UPC-E1	RMD_ATTR_SYM_UPC_E_1
EAN-8/JAN8	RMD_ATTR_SYM_EAN_8_JAN_8
EAN-13/JAN13	RMD_ATTR_SYM_EAN_13_JAN_13
Bookland EAN	RMD_ATTR_SYM_BOOKLAND_EAN
Code 128	RMD_ATTR_SYM_CODE_128
UCC/EAN-128	RMD_ATTR_SYM_UCC_EAN_128
Code 39	RMD_ATTR_SYM_CODE_39
Code 93	RMD_ATTR_SYM_CODE_93
Code 11	RMD_ATTR_SYM_CODE_11
Interleaved 2 of 5	RMD_ATTR_SYM_INTERLEAVED_2_OF_5
Discrete 2 of 5	RMD_ATTR_SYM_DISCRETE_2_OF_5
Chinese 2 of 5	RMD_ATTR_SYM_CHINESE_2_OF_5
Codabar	RMD_ATTR_SYM_CODABAR
MSI	RMD_ATTR_SYM_MSI
Code 32	RMD_ATTR_SYM_CODE_32
Data Matrix	RMD_ATTR_SYM_DATAMATRIXQR
PDF	RMD_ATTR_SYM_PDF

(continued on next page)

The following code example demonstrates how to get all supported symbologies.

#### Code Snippet - Retrieve Supported Symbologies

```
// ....

// Create XML string to request supported symbologies of scanner ID 3
NSString *xmlInput = [NSString
stringWithFormat:@"<inArgs><scannerID>%d</scannerID></inArgs>",3];

NSMutableString *xmlResponse = [[NSMutableString] alloc] init];

// Retrieve supported symbologies of scanner device that has an ID = 3.
SBT_RESULT result = [apiInstance sbtExecuteCommand:SBT_RSM_ATTR_GETALL
aInXML:xmlInput aOutXML:&xmlResponse forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process xml string containing supported symbologies
    NSLog(@"Supported symbologies from scanner ID %d:
%@", scannerId,xmlResponse);
}
else
{
    // TODO: Process error
    NSLog(@"Failed to retrieve supported symbologies from scanner ID
%d", scannerId);
}
```

**Get Symbology Values**

The get attribute command (SBT\_RSM\_ATTR\_GET) is used to retrieve the value of a symbology. The symbology value is returned from the scanner as an XML formatted string.

The following code example demonstrates how to retrieve the values of symbologies.

**Code Snippet - Retrieve Symbology Value**

```
// ....

// Create XML string to request value of supported symbology ID's 1 and 8 for
scanner ID 3
NSString *xmlInput = [NSString
stringWithFormat:@"<inArgs><scannerID>%d</scannerID><cmdArgs><arg-xml><attrib_lis
t>1,8</attrib_list></arg-xml></cmdArgs></inArgs>",3];

NSMutableString *xmlResponse = [[NSMutableString] alloc] init];

// Retrieved values of symbologies for scanner ID = 3.
SBT_RESULT result = [apiInstance sbtExecuteCommand:SBT_RSM_ATTR_GET
aInXML:xmlInput aOutXML:&xmlResponse forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process xml string containing symbology values
    NSLog(@"Supported symbology values from scanner ID %d:
%@", scannerId,xmlResponse);
}
else
{
    // TODO: Process error
    NSLog(@"Failed to retrieve symbology values from scanner ID %d",scannerId);
}
```



**Set Symbology Values**

The set attribute command (SBT\_RSM\_ATTR\_SET) is used to set the value of a symbology. The symbology value is returned from the scanner as an XML formatted string.

The following code example demonstrates how to set the value of symbology.

**Code Snippet - Set Symbology Value**

```
// ....

int scannerId = 3;
int symbologyId = 8;

// Create XML string to set value of symbology ID 8 for scanner ID 3 to "True"
NSString *xmlInput = [NSString stringWithFormat:@"
<inArgs><scannerID>%d</scannerID><cmdArgs><arg-xml><attrib_list><attribute><id>%d
</id><datatype>F</datatype><value>%@</value></attribute></attrib_list></arg-xml><
/cmdArgs></inArgs>", scannerId, symbologyId, "True"];

// Set value of symbology ID 8 for scanner ID = 3 to "True"
SBT_RESULT result = [apiInstance sbtExecuteCommand:SBT_RSM_ATTR_SET
aInXML:xmlInput aOutXML:nil forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process successful operation
    NSLog(@"Successfully updated symbology value for scanner ID %d:
%@", scannerId);
}
else
{
    // TODO: Process error
    NSLog(@"Failed to set symbology value for scanner ID %d", scannerId);
}
```

## Trigger

### *Pull Trigger*

The pull trigger command (SBT\_DEVICE\_PULL\_TRIGGER) is used to pull the device's trigger. The following code example demonstrates how to command the scanner's trigger to be pulled.

#### Code Snippet - Pull Trigger

```
// ....

// Create XML string to command scanner ID 3 to pull the trigger
NSString *xmlInput = [NSString stringWithFormat:@"
<inArgs><scannerID>%d</scannerID></inArgs>", scannerId];

// Command scanner ID 3 to pull the trigger
SBT_RESULT result = [apiInstance sbtExecuteCommand:SBT_DEVICE_PULL_TRIGGER
aInXML:xmlInput aOutXML:nil forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process successful operation
    NSLog(@"Successfully pulled trigger on scanner ID %d: %@", scannerId);
}
else
{
    // TODO: Process error
    NSLog(@"Failed to pull trigger on scanner ID %d", scannerId);
}
```

### *Release Trigger*

The release trigger command (SBT\_DEVICE\_RELEASE\_TRIGGER) is used to release the device's trigger. The following code example demonstrates how to command the scanner's trigger to be released.

#### Code Snippet - Release Trigger

```
// ....

// Create XML string to command scanner ID 3 to release the trigger
NSString *xmlInput = [NSString stringWithFormat:@"
<inArgs><scannerID>%d</scannerID></inArgs>", scannerId];

// Command scanner ID 3 to pull the trigger
SBT_RESULT result = [apiInstance sbtExecuteCommand:SBT_DEVICE_RELEASE_TRIGGER
aInXML:xmlInput aOutXML:nil forScanner:3];

if (result == SBT_RESULT_SUCCESS)
{
    // TODO: Process successful operation
    NSLog(@"Successfully released trigger on scanner ID %d: %@", scannerId);
}
else
{
    // TODO: Process error
    NSLog(@"Failed to release trigger on scanner ID %d", scannerId);
}
```

# Chapter 4 ZEBRA SCANNER SDK for iOS API

---

## Introduction

This chapter defines the API that can be used by external applications to connect remote scanners to a specific iOS device and control connected scanners.

## Application Programming Interface Definition

This section describes constants, types, functions, and notifications which are used in the Zebra Scanner SDK for iOS API.

### Constants

#### Result Codes

These constants are defined to represent result codes that can be returned by SDK API functions.

**Table 4-1** Constants - Result Codes

Result Code	Description	Value
<b>SBT_RESULT_SUCCESS</b>	A specific API function has completed successfully	0
<b>SBT_RESULT_FAILURE</b>	A specific API function has completed unsuccessfully.	1
<b>SBT_RESULT_SCANNER_NOT_AVAILABLE</b>	A specific API function has completed unsuccessfully because a specified scanner was not available.	2
<b>SBT_RESULT_SCANNER_NOT_ACTIVE</b>	A specific API function has completed unsuccessfully because a specified scanner was not active.	3
<b>SBT_RESULT_INVALID_PARAMS</b>	A specific API function has completed unsuccessfully due to invalid input and/or output parameters.	4
<b>SBT_RESULT_RESPONSE_TIMEOUT</b>	A specific API function has completed unsuccessfully due to expiration of a response timeout during communication with a specific scanner.	5
<b>SBT_RESULT_OPCODE_NOT_SUPPORTED</b>	A specific API function has completed unsuccessfully due to unsupported opcode.	6
<b>SBT_RESULT_SCANNER_NO_SUPPORT</b>	A specific API function and/or a corresponding SSI command are not supported by a specific model of scanner.	7

#### Operating Modes

These constants are defined to represent operating modes of the Zebra Scanner SDK for iOS.

**Table 4-2** Constants - Operating Modes

Operating Mode	Description	Value
<b>SBT_OPMODE_MFI</b>	The SDK is able to communicate with scanners in "iOS BT MFi" mode only.	1
<b>SBT_OPMODE_BTLE</b>	The SDK is able to communicate with scanners in "iOS BT LE" mode only.	2
<b>SBT_OPMODE_ALL</b>	The SDK is able to communicate with scanners in "iOS BT MFi" mode and with scanners in "iOS BT LE" mode.	3

## Scanner Modes

These constants are defined to represent communication modes of scanners.

**Table 4-3** Constants - Scanner Modes

Scanner Mode	Description	Value
<b>SBT_CONNTYPE_INVALID</b>	The SDK is unable to determine communication mode of a specific scanner.	0
<b>SBT_CONNTYPE_MFI</b>	A specific scanner is in "iOS BT MFi" mode.	1
<b>SBT_CONNTYPE_BTLE</b>	A specific scanner is in "iOS BT LE" mode.	2

## Notifications

These constants are defined to represent notifications provided by Zebra Scanner SDK for iOS.

**Table 4-4** Constants - Notifications

Notification	Description	Value
<b>SBT_EVENT_BARCODE</b>	"Barcode event" notification (reception of a specific bar code of specific type from a specific active scanner).	1
<b>SBT_EVENT_IMAGE</b>	"Image event" notification (triggered when an imaging scanner captures images in image mode).	2
<b>SBT_EVENT_VIDEO</b>	"Video event" notification (triggered when an imaging scanner captures video in video mode).	4
<b>SBT_EVENT_SCANNER_APPEARANCE</b>	"Device Arrival" notification (appearance of an available scanner).	8
<b>SBT_EVENT_SCANNER_DISAPPEARANCE</b>	"Device Disappeared" notification (disappearance of an available scanner).	16
<b>SBT_EVENT_SESSION_ESTABLISHMENT</b>	"Session Established" notification (appearance of a specific active scanner).	32
<b>SBT_EVENT_SESSION_TERMINATION</b>	"Session Terminated" notification (disappearance of an active scanner).	64
<b>SBT_EVENT_RAW_DATA</b>	"Raw Data Received" notification (reception of raw data from a specific active scanner in "raw data pipe" mode).	128

## Opcodes

These constants are defined to represent opcodes of methods supported by "Execute Command" API function.

Table 4-5 Constants - Opcodes

Opcode	Description	Value
<b>SBT_DEVICE_PULL_TRIGGER</b>	Opcode of <code>DEVICE_PULL_TRIGGER</code> method of the "Execute Command" API.	2011
<b>SBT_DEVICE_RELEASE_TRIGGER</b>	Opcode of <code>DEVICE_RELEASE_TRIGGER</code> method of the "Execute Command" API.	2012
<b>SBT_DEVICE_CAPTURE_IMAGE</b>	Opcode of <code>DEVICE_CAPTURE_IMAGE</code> method of the "Execute Command" API.	3000
<b>SBT_DEVICE_CAPTURE_VIDEO</b>	Opcode of <code>DEVICE_CAPTURE_VIDEO</code> method of the "Execute Command" API.	4000
<b>SBT_DEVICE_CAPTURE_BARCODE</b>	Opcode of <code>DEVICE_CAPTURE_BARCODE</code> method of the "Execute Command" API.	3500
<b>SBT_DEVICE_SCAN_ENABLE</b>	Opcode of <code>SCAN_ENABLE</code> method of the "Execute Command" API.	2014
<b>SBT_DEVICE_SCAN_DISABLE</b>	Opcode of <code>SCAN_DISABLE</code> method of the "Execute Command" API.	2013
<b>SBT_SET_ACTION</b>	Opcode of <code>SET_ACTION</code> method of the "Execute Command" API.	6000
<b>SBT_RSM_ATTR_GETALL</b>	Opcode of <code>ATTR_GETALL</code> method of the "Execute Command" API.	5000
<b>SBT_RSM_ATTR_GET</b>	Opcode of <code>ATTR_GET</code> method of the "Execute Command" API.	5001
<b>SBT_RSM_ATTR_GET_OFFSET</b>	Opcode of <code>ATTR_GET_OFFSET</code> method of the "Execute Command" API.	5003
<b>SBT_RSM_ATTR_SET</b>	Opcode of <code>ATTR_SET</code> method of the "Execute Command" API.	5004
<b>SBT_RSM_ATTR_STORE</b>	Opcode of <code>ATTR_STORE</code> method of the "Execute Command" API.	5005

## Scanner Models

These constants are defined to represent models of scanners supported by Zebra Scanner SDK for iOS.

**Table 4-6** Constants - Scanner Models

Scanner Model	Description	Value
<code>SBT_DEVMODEL_INVALID</code>	The model either unknown, not recognized or not supported.	0
<code>SBT_DEVMODEL_SSI_RFD8500</code>	RFD8500 in SSI Imager mode.	1
<code>SBT_DEVMODEL_SSI_CS4070</code>	CS4070 scanner.	2
<code>SBT_DEVMODEL_SSI_GENERIC</code>	Generic SSI imager.	3
<code>SBT_DEVMODEL_RFID_RFD8500</code>	RFD8500 in RFID reader mode.	4

## LED Codes

These constants are defined to represent LED codes supported by "LED Control" API function.

**Table 4-7** Constants - LED Codes

LED Code	Description	Value
<code>SBT_LEDCODE_RED</code>	Red LED.	0
<code>SBT_LEDCODE_GREEN</code>	Green LED.	1
<code>SBT_LEDCODE_YELLOW</code>	Yellow LED.	2
<code>SBT_LEDCODE_AMBER</code>	Amber LED.	3
<code>SBT_LEDCODE_BLUE</code>	Blue LED.	4

## Beep Codes

These constants are defined to represent beep codes supported by "Beep Control" API function.

**Table 4-8** Constants - Beep Codes

Beep Code	Description	Value
<code>SBT_BEEPCODE_SHORT_HIGH_1</code>	One high short beep.	0
<code>SBT_BEEPCODE_SHORT_HIGH_2</code>	Two high short beeps.	1
<code>SBT_BEEPCODE_SHORT_HIGH_3</code>	Three high short beeps.	2
<code>SBT_BEEPCODE_SHORT_HIGH_4</code>	Four high short beeps.	3
<code>SBT_BEEPCODE_SHORT_HIGH_5</code>	Five high short beeps.	4
<code>SBT_BEEPCODE_SHORT_LOW_1</code>	One low short beep.	5
<code>SBT_BEEPCODE_SHORT_LOW_2</code>	Two low short beeps.	6
<code>SBT_BEEPCODE_SHORT_LOW_3</code>	Three low short beeps.	7
<code>SBT_BEEPCODE_SHORT_LOW_4</code>	Four low short beeps.	8

Table 4-8 Constants - Beep Codes (Continued)

<b>Beep Code</b>	<b>Description</b>	<b>Value</b>
<code>SBT_BEEPCODE_SHORT_LOW_5</code>	Five low short beeps.	9
<code>SBT_BEEPCODE_LONG_HIGH_1</code>	One high long beep.	10
<code>SBT_BEEPCODE_LONG_HIGH_2</code>	Two high long beeps.	11
<code>SBT_BEEPCODE_LONG_HIGH_3</code>	Three high long beeps.	12
<code>SBT_BEEPCODE_LONG_HIGH_4</code>	Four high long beeps.	13
<code>SBT_BEEPCODE_LONG_HIGH_5</code>	Five high long beeps.	14
<code>SBT_BEEPCODE_LONG_LOW_1</code>	One low long beep.	15
<code>SBT_BEEPCODE_LONG_LOW_2</code>	Two low long beeps.	16
<code>SBT_BEEPCODE_LONG_LOW_3</code>	Three low long beeps.	17
<code>SBT_BEEPCODE_LONG_LOW_4</code>	Four low long beeps.	18
<code>SBT_BEEPCODE_LONG_LOW_5</code>	Five low long beeps.	19
<code>SBT_BEEPCODE_FAST_WARBLE</code>	Fast warble beep.	20
<code>SBT_BEEPCODE_SLOW_WARBLE</code>	Slow warble beep.	21
<code>SBT_BEEPCODE_MIX1_HIGH_LOW</code>	High-low mix beep.	22
<code>SBT_BEEPCODE_MIX2_LOW_HIGH</code>	Low-high mix beep.	23
<code>SBT_BEEPCODE_MIX3_HIGH_LOW_HIGH</code>	High-low-high mix beep.	24
<code>SBT_BEEPCODE_MIX4_LOW_HIGH_LOW</code>	Low-high-low mix beep.	25



## Types

### SBT\_RESULT

Type of return value of Zebra Scanner SDK for iOS API functions.

```
typedef enum {
    SBT_RESULT_SUCCESS                = 0x00,
    SBT_RESULT_FAILURE                = 0x01,
    SBT_RESULT_SCANNER_NOT_AVAILABLE = 0x02,
    SBT_RESULT_SCANNER_NOT_ACTIVE    = 0x03,
    SBT_RESULT_INVALID_PARAMS        = 0x04,
    SBT_RESULT_RESPONSE_TIMEOUT      = 0x05,
    SBT_RESULT_OPCODE_NOT_SUPPORTED  = 0x06,
    SBT_RESULT_SCANNER_NO_SUPPORT    = 0x07
} SBT_RESULT;
```

### SbtScannerInfo

Used to represent a specific scanner.

```
@interface SbtScannerInfo : NSObject
{
    int m_ScannerID;
    int m_ConnectionType;
    BOOL m_AutoCommunicationSessionReestablishment;
    BOOL m_Active;
    BOOL m_Available;
    NSString *m_ScannerName;
    int m_ScannerModel;
}

- (id) init;
- (void) dealloc;
- (void) setScannerID:(int)scannerID;
- (void) setConnectionType:(int)connectionType;
- (void) setAutoCommunicationSessionReestablishment:(BOOL)enable;
- (void) setActive:(BOOL)active;
- (void) setAvailable:(BOOL)available;
- (void) setScannerName:(NSString*)scannerName;
- (void) setScannerModel:(int)scannerModel;

- (int) getScannerID;
- (int) getConnectionType;
- (BOOL) getAutoCommunicationSessionReestablishment;
- (BOOL) isActive;
- (BOOL) isAvailable;
- (NSString*) getScannerName;
- (int) getScannerModel;

@end
```

**Table 4-9** *SbtScannerInfo* Variable Descriptions

Variable Name	Description
m_ScannerID	Unique identifier of a specific scanner assigned by SDK.
m_ConnectionType	Communication mode of a specific scanner.
m_AutoCommunicationSessionReestablishment	State of "Automatic communication session reestablishment" option.

**Table 4-9** *SbtScannerInfo Variable Descriptions (Continued)*

Variable Name	Description
m_Active	State of a specific scanner (i.e., active scanner is a scanner with which a communication session was already established).
m_ScannerName	Name of a specific scanner.
m_ScannerModel	Model code of a specific scanner.

## ISbtSdkApi

Objective C protocol which defines SDK API functions.

```
@protocol ISbtSdkApi <NSObject>

- (SBT_RESULT) sbtSetDelegate:(id<ISbtSdkApiDelegate>) delegate;
- (NSString*) sbtGetVersion;
- (SBT_RESULT) sbtSetOperationalMode:(int) operationalMode;
- (SBT_RESULT) sbtSubscribeForEvents:(int) sdkEventsMask;
- (SBT_RESULT) sbtUnsubscribeForEvents:(int) sdkEventsMask;
- (SBT_RESULT) sbtGetAvailableScannersList:(NSMutableArray**) availableScannersList;
- (SBT_RESULT) sbtGetActiveScannersList:(NSMutableArray**) activeScannersList;
- (SBT_RESULT) sbtEstablishCommunicationSession:(int) scannerID;
- (SBT_RESULT) sbtTerminateCommunicationSession:(int) scannerID;
- (SBT_RESULT) sbtEnableAvailableScannersDetection:(BOOL) enable;
- (SBT_RESULT) sbtEnableAutomaticSessionReestablishment:(BOOL) enable
forScanner:(int) scannerID;
- (SBT_RESULT) sbtExecuteCommand:(int) opCode aInXML:(NSString*) inXML
aOutXML:(NSMutableString**) outXML forScanner:(int) scannerID;
- (SBT_RESULT) sbtLedControl:(BOOL) enable aLedCode:(int) ledCode forScanner:(int) scannerID;
- (SBT_RESULT) sbtBeepControl:(int) beepCode forScanner:(int) scannerID;

@end
```

See [Functions on page 4-9](#) for the descriptions of API functions.

## SbtSdkFactory

Used to create and access a single shared instance of an API object. API object implements `ISbtSdkApi` protocol and is used to perform specific API calls.

```
@interface SbtSdkFactory : NSObject

+(id<ISbtSdkApi>) createSbtSdkApiInstance;

@end
```

**Table 4-10** *SbtScannerInfo Variable Descriptions*

Name	Description
<code>+(id&lt;ISbtSdkApi&gt;) createSbtSdkApiInstance</code>	Class method, returns a single shared instance of an object which conforms to <code>ISbtSdkApi</code> protocol and should be used to perform specific API calls.

## ISbtSdkApiDelegate

Objective C protocol which defines the SDK callbacks interface. Registration of a specific object which conforms to the `ISbtSdkApiDelegate` protocol is required to receive particular from the SDK.

```

@protocol ISbtSdkApiDelegate <NSObject>

- (void) sbtEventScannerAppeared: (SbtScannerInfo*) availableScanner;
- (void) sbtEventScannerDisappeared: (int) scannerID;
- (void) sbtEventCommunicationSessionEstablished: (SbtScannerInfo*) activeScanner;
- (void) sbtEventCommunicationSessionTerminated: (int) scannerID;
- (void) sbtEventBarcode: (NSString*) barcodeData barcodeType: (int) barcodeType
fromScanner: (int) scannerID;
- (void) sbtEventBarcodeData: (NSData *) barcodeData barcodeType: (int) barcodeType
fromScanner: (int) scannerID;
- (void) sbtEventImage: (NSData*) imageData fromScanner: (int) scannerID;
- (void) sbtEventVideo: (NSData*) videoFrame fromScanner: (int) scannerID;

@end

```

See [Notifications on page 4-17](#) for descriptions of SDK callbacks.

## Functions

API functions are defined by `ISbtSdkApi` Objective C protocol. A specific object which implements the protocol is accessible via `createSbtSdkApiInstance` method of `SbtSdkFactory` class. See [ISbtSdkApi on page 4-8](#) and [SbtSdkFactory on page 4-8](#) for details.

### sbtGetVersion

#### Description

Returns version of the SDK.

```
- (NSString*) sbtGetVersion;
```

#### Parameters

None.

#### Return Values

SDK version as NSString autoreleased object.

### sbtSetDelegate

#### Description

Registers a specific object which conforms to `ISbtSdkApiDelegate` Objective C protocol as a receiver of SDK notifications. Registration of a specific object which conforms to `ISbtSdkApiDelegate` protocol is required to receive notifications from the SDK.

```
- (SBT_RESULT) sbtSetDelegate: (id<ISbtSdkApiDelegate>) delegate;
```

#### Parameters

```
(id<ISbtSdkApiDelegate>) delegate
```

[in] An object which conforms to `ISbtSdkApiDelegate` protocol.

#### Return Values

```
SBT_RESULT_SUCCESS
```

The receiver of SDK notifications was registered successfully.

### sbtSetOperationalMode

#### Description

Configures operating mode of the SDK.

```
- (SBT_RESULT) sbtSetOperationalMode: (int) operationalMode;
```

**Parameters**`(int)operationalMode`

[in] Identifier of requested operating mode.

**Return Values**`SBT_RESULT_SUCCESS`

The requested operating mode of the SDK was configured successfully.

`SBT_RESULT_FAILURE`

Invalid parameters.

**NOTES** If operating mode of the SDK is not configured intentionally, the SDK remains disabled and is not able to communicate with scanners in either "iOS BT MFI" or "iOS BT LE" mode.

Selection of a new operating mode results in:

- Disconnection of active incompatible scanners (e.g., "iOS BT LE" scanners in `SBT_OPMODE_MFI` mode), removing available incompatible scanners and providing corresponding notifications if these notifications are enabled.
- Performing discovery of available scanners compatible with requested operating mode (if "Available scanners detection" option is enabled) and providing corresponding notifications if these notifications are enabled.

**sbtSubscribeForEvents****Description**

Enables providing of notification of requested types.

- `(SBT_RESULT) sbtSubscribeForEvents:(int) sdkEventsMask;`**Parameters**`(int) sdkEventsMask`

[in] Subscription option flags.

**Return Values**`SBT_RESULT_SUCCESS`

Subscription was performed successfully.

**sbtUnsubscribeForEvents****Description**

Disables providing of notification of requested types.

- `(SBT_RESULT) sbtUnsubscribeForEvents:(int) sdkEventsMask;`**Parameters**`(int) sdkEventsMask`

[in] Unsubscription option flags.

**Return Values**`SBT_RESULT_SUCCESS`

Unsubscription was performed successfully.

**sbtGetAvailableScannersList****Description**

Requests the list of available scanners.

- `(SBT_RESULT) sbtGetAvailableScannersList:(NSMutableArray**) availableScannersList;`

**Parameters**

(NSMutableArray\*\*) availableScannersList

[out] Pointer to NSMutableArray object which contains SbtScannerInfo objects that represents available scanners.

**Return Values**

SBT\_RESULT\_SUCCESS

Discovery procedure was performed and availableScannersList parameter is filled with available scanners.

SBT\_RESULT\_INVALID\_PARAMS

Invalid parameter (e.g., nil pointer).

**NOTES**

- The caller is responsible for allocation, initialization and deallocation of NSMutableArray object.
- All objects from received NSMutableArray object are removed.
- The SDK performs discovery of available scanners without providing any notifications.
- Received NSMutableArray object is filled with SbtScannerInfo objects that represents available scanners.

**sbtGetActiveScannersList****Description**

Requests the list of active scanners.

- (SBT\_RESULT) sbtGetActiveScannersList:(NSMutableArray\*\*) activeScannersList;

**Parameters**

(NSMutableArray\*\*) activeScannersList

[out] Pointer to NSMutableArray object which contains SbtScannerInfo objects that represents active scanners.

**Return Values**

SBT\_RESULT\_SUCCESS

Discovery procedure was performed and activeScannersList parameter is filled with available scanners.

SBT\_RESULT\_INVALID\_PARAMS

Invalid parameter (e.g., nil pointer).

**NOTES**

- The caller is responsible for allocation, initialization and deallocation of NSMutableArray object.
- All objects from received NSMutableArray object are removed.
- Received NSMutableArray object is filled with SbtScannerInfo objects that represents available scanners.

**sbtEstablishCommunicationSession****Description**

Requests to establish communication session with a specific available scanner in "SSI" mode.

- (SBT\_RESULT) sbtEstablishCommunicationSession:(int) scannerID;

**Parameters**

(int) scannerID

[in] Unique identifier of a specific scanner assigned by SDK.

**Return Values**

SBT\_RESULT\_SUCCESS

The communication session was established successfully.

SBT\_RESULT\_FAILURE

The communication session was not established.

SBT\_RESULT\_SCANNER\_NO\_SUPPORT

A specific scanner does not support SSI communication mode.

✓ **NOTE** "Session Established" notification is provided if this type of notification is enabled.

**sbtTerminateCommunicationSession****Description**

Requests to terminate communication session with a specific active scanner.

- (SBT\_RESULT) sbtTerminateCommunicationSession:(int) scannerID;

**Parameters**

(int) scannerID

[in] Unique identifier of a specific scanner assigned by SDK.

**Return Values**

SBT\_RESULT\_SUCCESS

The communication session was terminated successfully.

SBT\_RESULT\_FAILURE

The communication session was not terminated.

✓ **NOTE** "Session Termination" notification is provided if this type of notification is enabled.

**sbtEnableAvailableScannersDetection****Description**

Requests to enable/disable "Available scanners detection" option.

- (SBT\_RESULT) sbtEnableAvailableScannersDetection:(BOOL) enable;

**Parameters**

(BOOL) enable

[in] Whether the option should be enabled or disabled:

YES

Requests to enable "Available scanners detection" option.

NO

Requests to disable "Available scanners detection" option.

**Return Values**

SBT\_RESULT\_SUCCESS

"Available scanners detection" option was enabled/disabled successfully.

**NOTES**

- The SDK performs a discovery of available scanners once the option is enabled and provides corresponding notifications if the notifications are enabled.
- If the option is enabled, the SDK detects connection and disconnection of scanners operating in "iOS BT MFi" mode in both foreground and background execution modes of a specific application linked with the SDK and provides corresponding notifications if the notifications are enabled.
- If the option is enabled the SDK detects appearance and disappearance of scanners operating in "iOS BT LE" mode in only foreground execution mode of a specific application linked with the SDK through periodic discovery operation and provides corresponding notifications if the notifications are enabled.

**sbtEnableAutomaticSessionReestablishment****Description**

Requests to enable/disable "Automatic communication session reestablishment" option for a specific scanner.

```
- (SBT_RESULT) sbtEnableAutomaticSessionReestablishment:(BOOL)enable
forScanner:(int)scannerID;
```

**Parameters**

(int) scannerID

[in] Unique identifier of a specific scanner assigned by SDK.

(BOOL) enable

[in] Whether the option should be enabled or disabled:

YES

Requests to enable "Automatic communication session reestablishment" option.

NO

Requests to disable "Automatic communication session reestablishment" option.

**Return Values**

SBT\_RESULT\_SUCCESS

"Automatic communication session reestablishment" option was enabled/disabled successfully.

SBT\_RESULT\_FAILURE

The specified scanner was not found.

**NOTES**

- If the option is enabled for a specific scanner the SDK automatically establishes a communication session in "SSI" mode with this scanner once the scanner is recognized as available:
- The scanner can be recognized as available automatically by the SDK if the "Available scanners detection" option is enabled.
- The scanner can be recognized as available during discovery procedure requested by the `sbtGetAvailableScannersList` API.
- "Session Established" notification is provided once the communication session is established if this type of notification is enabled.

## sbtExecuteCommand

### Description

Provides synchronous execution of a specific method via an opcode. See [Methods of the "Execute Command" API on page 4-20](#) for description of specific methods.

```
- (SBT_RESULT) sbtExecuteCommand:(int)opCode aInXML:(NSString*)inXML
aOutXML:(NSMutableString**)outXML forScanner:(int)scannerID;
```

### Parameters

(int)opCode

[in] Method to be executed.

(NSString\*)inXML

[in] Relevant argument list for the opcode, structured into an XML string.

(NSMutableString\*\*)outXML

[out] Results of method execution, structured into an XML string.

(int)scannerID

[in] Unique identifier of a specific scanner assigned by SDK.

### Return Values

SBT\_RESULT\_SUCCESS

The requested method was executed successfully.

SBT\_RESULT\_SCANNER\_NOT\_AVAILABLE

The requested method was not executed because the scanner specified by scannerID parameter was not available.

SBT\_RESULT\_SCANNER\_NOT\_ACTIVE

The requested method was not executed because the scanner specified by scannerID parameter was not active.

SBT\_RESULT\_OPCODE\_NOT\_SUPPORTED

The requested method is not supported by SDK.

SBT\_RESULT\_INVALID\_PARAMS

The requested method was not executed due to invalid input and/or output parameters.

SBT\_RESULT\_RESPONSE\_TIMEOUT

The requested method was completed unsuccessfully due to expiration of a response timeout during communication with a specific scanner

SBT\_RESULT\_FAILURE

The requested method was executed unsuccessfully.

SBT\_RESULT\_SCANNER\_NO\_SUPPORT

The requested method is not supported by the scanner.



### NOTES

- The caller is responsible for allocation, initialization and deallocation of outXML NSMutableString object.
- scannerID identifier of scanner is the same as scanner ID identifier provided in inXML string argument.



## sbtLedControl

### Description

Perform an action involving the scanner's LEDs. See [LED Codes on page 4-5](#) for supported LED codes.

```
- (SBT_RESULT) sbtLedControl:(BOOL)enable aLedCode:(int)ledCode
forScanner:(int)scannerID;
```

### Deprecation Statement

This API will be removed in the future. Use the `sbtExecuteCommand` function instead to perform an LED action.

### Parameters

(BOOL) enable

[in] Whether a specified LED is turned on or off.

(int) ledCode

[in] Code of a specific LED.

(int) scannerID

[in] Unique identifier of a specific scanner assigned by SDK.

### Return Values

SBT\_RESULT\_SUCCESS

LED control operation was performed successfully.

SBT\_RESULT\_SCANNER\_NOT\_AVAILABLE

LED control operation was not performed because the scanner specified by `scannerID` parameter was not available.

SBT\_RESULT\_SCANNER\_NOT\_ACTIVE

LED control operation was not performed because the scanner specified by `scannerID` parameter was not active.

SBT\_RESULT\_INVALID\_PARAMS

LED control operation was not performed due to invalid LED code or LED code that is not supported by the scanner.

SBT\_RESULT\_RESPONSE\_TIMEOUT

LED control operation was completed unsuccessfully due to expiration of a response timeout during communication with a specific scanner

SBT\_RESULT\_FAILURE

LED control operation was executed unsuccessfully.

SBT\_RESULT\_SCANNER\_NO\_SUPPORT

A specific scanner does not support required SSI commands.



### NOTES

- The LED control operation is performed through sending LED\_ON or LED\_OFF SSI protocol command.
- The LED control operation is supposed to be performed successfully if CMD\_ACK is received from a scanner.

## sbtBeepControl

### Description

Perform an action involving the scanner's beeper. See [Beep Codes on page 4-5](#) for supported beep codes.

- (SBT\_RESULT) sbtBeepControl:(int)beepCode forScanner:(int)scannerID;

### Deprecation Statement

This API will be removed in the future. Use the `sbtExecuteCommand` function instead to perform a beeper action.

### Parameters

(int)beepCode

[in] A specific beep code.

(int)scannerID

[in] Unique identifier of a specific scanner assigned by SDK.

### Return Values

SBT\_RESULT\_SUCCESS

The beep control operation was performed successfully.

SBT\_RESULT\_SCANNER\_NOT\_AVAILABLE

The beep control operation was not performed because the scanner specified by `scannerID` parameter was not available.

SBT\_RESULT\_SCANNER\_NOT\_ACTIVE

The beep control operation was not performed because the scanner specified by `scannerID` parameter was not active.

SBT\_RESULT\_INVALID\_PARAMS

The beep control operation was not performed due to invalid beep code or beep code that is not supported by the scanner.

SBT\_RESULT\_RESPONSE\_TIMEOUT

The beep control operation was completed unsuccessfully due to expiration of a response timeout during communication with a specific scanner

SBT\_RESULT\_FAILURE

The beep control operation was executed unsuccessfully.

SBT\_RESULT\_SCANNER\_NO\_SUPPORT

A specific scanner does not support required SSI commands.



### NOTES

- The beep control operation is performed through sending BEEP SSI protocol command for action commands involving scanner's beeper.
- The beep control operation is supposed to be performed successfully if `CMD_ACK` is received from a scanner.

## Notifications

The SDK callback interface is defined by `ISbtSdkApiDelegate` Objective C protocol. Registration of a specific object which conforms to `ISbtSdkApiDelegate` protocol is required to receive specific notifications from Zebra Scanner SDK for iOS.

### **sbtEventScannerAppeared**

#### **Description**

"Device Arrival" notification informs about appearance of a specific available scanner.

- (void) sbtEventScannerAppeared:(SbtScannerInfo\*)availableScanner

#### **Parameters**

(SbtScannerInfo\*)availableScanner

SbtScannerInfo object representing an appeared available scanner.



**NOTE** The SDK is responsible for allocation and deallocation of availableScanner object. The SDK deallocates the availableScanner object immediately after execution of the callback.

### **sbtEventScannerDisappeared**

#### **Description**

"Device Disappeared" notification informs about disappearance of a specific available scanner.

- (void) sbtEventScannerDisappeared:(int)scannerID;

#### **Parameters**

(int)scannerID

Unique identifier of a disappeared available scanner assigned by SDK.

### **sbtEventCommunicationSessionEstablished**

#### **Description**

"Session Established" notification informs about appearance of a specific active scanner.

- (void) sbtEventCommunicationSessionEstablished:(SbtScannerInfo\*)activeScanner;

#### **Parameters**

(SbtScannerInfo\*)activeScanner

SbtScannerInfo object representing an appeared active scanner.



**NOTE** The SDK is responsible for allocation and deallocation of activeScanner object. The SDK deallocates the activeScanner object immediately after execution of the callback.

### **sbtEventCommunicationSessionTerminated**

#### **Description**

"Session Terminated" notification informs about disappearance of a specific active scanner

- (void) sbtEventCommunicationSessionTerminated:(int)scannerID;

#### **Parameters**

(int)scannerID

Unique identifier of a disappeared active scanner assigned by SDK.

**sbtEventBarcode (deprecated)****Description**

"Barcode Event" notification informs about reception of a specific bar code of a specific type from a specific active scanner.

```
- (void) sbtEventBarcode:(NSString*)barcodeData barcodeType:(int)barcodeType
fromScanner:(int)scannerID;
```

**Deprecation Statement**

This API will be removed in the future. Use the `sbtEventBarcodeData` function instead (see [sbtEventBarcodeData](#) below).

**Parameters**

```
(NSString*)barcodeData
```

NSString object representing raw data of the scanned bar code.

```
(int)barcodeType
```

Bar code type of the scanned bar code. Values of bar code data types are available in the *Zebra Scanner SDK For Windows Developer Guide* (p/n 72E-149784-XX).

```
(int)scannerID
```

Unique identifier of a specific active scanner assigned by SDK.

- ✓ **NOTE** The SDK is responsible for allocation and deallocation of `barcodeData` object. The SDK deallocates the `barcodeData` object immediately after execution of the callback.

**sbtEventBarcodeData****Description**

"Barcode Event" notification informs about reception of a particular barcode of a particular type from a particular active scanner.

```
- (void) sbtEventBarcodeData:(NSData*)barcodeData barcodeType:(int)barcodeType
fromScanner:(int)scannerID;
```

**Parameters**

```
(NSData*)barcodeData
```

NSData object representing raw data byte of the scanned bar code.

```
(int)barcodeType
```

Bar code type of the scanned bar code. Values of bar code data types are available in the *Zebra Scanner SDK For Windows Developer Guide* (p/n 72E-149784-XX).

```
(int)scannerID
```

Unique identifier of a particular active scanner assigned by SDK.

- ✓ **NOTE** - Use this function instead of `sbtEventBarcode`.

**sbtEventImage****Description**

"Image Event" notification is triggered when an active imaging scanner captures images in image mode.

```
- (void) sbtEventImage:(NSData*)imageData fromScanner:(int)scannerID;
```

**Parameters**

```
(NSData*)barcodeData
```

NSData object representing raw data of the received image.

```
(int)scannerID
```

Unique identifier of a specific active scanner assigned by SDK.

✓ **NOTE** The SDK is responsible for allocation and deallocation of `imageData` object. The SDK deallocates the `imageData` object immediately after execution of the callback.

**Not implemented.**

**sbtEventVideo****Description**

"Video Event" notification is triggered when an active imaging scanner captures video in video mode.

```
(void)sbtEventVideo:(NSData*)videoFrame fromScanner:(int)scannerID;
```

**Parameters**

```
(NSData*)videoFrame
```

NSData object representing raw data of the received video frame.

```
(int)scannerID
```

Unique identifier of a specific active scanner assigned by SDK.

✓ **NOTE** The SDK is responsible for allocation and deallocation of `videoFrame` object. The SDK deallocates the `videoFrame` object immediately after execution of the callback.

**Not implemented.**

## Methods of the "Execute Command" API

The described methods are invoked through the "Execute Command" API. See [sbtExecuteCommand on page 4-14](#) for details. The execution of most of the described methods results in communicating with a specific active scanner via SSI and RMD protocols. Refer to the *Simple Serial Interface Version 2.0 Internal Specification* and RMD protocol specification for details.

### DEVICE\_PULL\_TRIGGER

Pull the trigger of a specified active scanner.

#### Opcode

SBT\_DEVICE\_PULL\_TRIGGER

#### inXML

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
</inArgs>
```

#### outXML

Not used.



#### NOTES

- Execution of the method results in sending START\_SESSION SSI protocol command.
- The method is supposed to be executed successfully if CMD\_ACK is received from a scanner.

### DEVICE\_RELEASE\_TRIGGER

Release the pulled trigger of a specified active scanner.

#### Opcode

SBT\_DEVICE\_RELEASE\_TRIGGER

#### inXML

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
</inArgs>
```

#### outXML

Not used.



#### NOTES

- Execution of the method results in sending STOP\_SESSION SSI protocol command.
- The method is supposed to be executed successfully if CMD\_ACK is received from a scanner.

**SCAN\_DISABLE**

Disable scanning of a specified active scanner.

**Opcode**

SBT\_DEVICE\_SCAN\_DISABLE

**inXML**

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
</inArgs>
```

**outXML**

Not used.

**NOTES**

- Execution of the method results in sending `SCAN_DISABLE` SSI protocol command.
- The method is supposed to be executed successfully if `CMD_ACK` is received from a scanner.

**SCAN\_ENABLE**

Enable scanning on a specified active scanner.

**Opcode**

SBT\_DEVICE\_SCAN\_ENABLE

**inXML**

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
</inArgs>
```

**outXML**

Not used.

**NOTES**

- Execution of the method results in sending `SCAN_ENABLE` SSI protocol command.
- The method is supposed to be executed successfully if `CMD_ACK` is received from a scanner.

**DEVICE\_CAPTURE\_BARCODE**

Change a specified active scanner to decode mode.

**Opcode**

SBT\_DEVICE\_CAPTURE\_BARCODE

**inXML**

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
</inArgs>
```

**outXML**

Not used.

**NOTES**

- Execution of the method results in sending `IMAGER_MODE` SSI protocol command.
- The method is supposed to be executed successfully if `CMD_ACK` is received from a scanner.

**DEVICE\_CAPTURE\_IMAGE**

Change a specified active scanner to snapshot mode.

**Opcode**

SBT\_DEVICE\_CAPTURE\_IMAGE

**inXML**

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
</inArgs>
```

**outXML**

Not used.

**NOTES**

- Execution of the method results in sending `IMAGER_MODE` SSI protocol command.
- The method is supposed to be executed successfully if `CMD_ACK` is received from a scanner.



## DEVICE\_CAPTURE\_VIDEO

Change a specified active scanner to video mode.

### **Opcode**

SBT\_DEVICE\_CAPTURE\_VIDEO

### **inXML**

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
</inArgs>
```

### **outXML**

Not used.



#### **NOTES**

- Execution of the method results in sending IMAGER\_MODE SSI protocol command.
- - The method is supposed to be executed successfully if CMD\_ACK is received from a scanner.

## SET\_ACTION

Perform an action involving the scanner's beeper or LEDs. Values for the SET\_ACTION method are available in the *Zebra Scanner SDK Attribute Dictionary* (p/n 72E-149786-XX).

### **Opcode**

SBT\_SET\_ACTION

### **inXML**

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
  <cmdArgs>
    <arg-int>0</arg-int>             ? Scanner Action Command
  </cmdArgs>
</inArgs>
```

### **outXML**

Not used.



#### **NOTES**

- Execution of the method results in sending ATTRIBUTE\_SET RMD protocol command via SSI\_MGMT\_COMMAND SSI protocol command in order to set a required value of "Beeper/LED" attribute.
- The method is supposed to be executed successfully if ATTRIBUTE\_SET RMD protocol command for "Beeper/LED" attribute was sent to a scanner and a corresponding response for the issued ATTRIBUTE\_SET RMD command is received via SSI\_MGMT\_COMMAND SSI packets from a scanner.

**ATTR\_GETALL**

Get all attributes of a specified active scanner. Refer to the *Zebra Scanner SDK Attribute Dictionary* (p/n 72E-149786-XX) for attribute numbers, types and possible values.

**Opcode**

SBT\_RSM\_ATTR\_GETALL

**inXML**

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
</inArgs>
```

**outXML**

```
<?xml version="1.0" encoding="UTF-8"?>
<outArgs>
  <scannerID>1</scannerID>           ? Scanner ID of Data Receiving
  <arg-xml>
    <modelnumber>DS670-SR20001ZZR</modelnumber>
    <serialnumber>7116000501003</serialnumber>
    <response>
      <opcode>5000</opcode>          ? Method response received
      <attrib_list>
        <attribute name="">0</attribute>? Attribute Numbers
        <attribute name="">1</attribute>
        <attribute name="">2</attribute>
        <attribute name="">17</attribute>
        <attribute name="">18</attribute>
        <attribute name="">19</attribute>
        <attribute name="">655</attribute>
        <attribute name="">656</attribute>
        <attribute name="">657</attribute>
        <attribute name="">705</attribute>
        <attribute name="">716</attribute>
        <attribute name="">718</attribute>
        <attribute name="">735</attribute>
        <attribute name="">745</attribute>
        <attribute name="">6000</attribute>
        <attribute name="">6001</attribute>
        <attribute name="">6002</attribute>
        <attribute name="">6003</attribute>
        <attribute name="">6004</attribute>
        <attribute name="">20004</attribute>
        <attribute name="">20006</attribute>
        <attribute name="">20007</attribute>
        <attribute name="">20008</attribute>
        <attribute name="">20009</attribute>
        <attribute name="">20010</attribute>
        <attribute name="">20011</attribute>
        <attribute name="">20013</attribute>
      </attrib_list>
    </response>
  </arg-xml>
</outArgs>
```

**NOTES**

- Execution of the method results in sending ATTRIBUTE\_GETALL RMD protocol command(s) via SSI\_MGMT\_COMMAND SSI protocol command.
- The method is supposed to be executed successfully if a set of ATTRIBUTE\_GETALL RMD responses are received via SSI\_MGMT\_COMMAND packets from a scanner and the last response packet indicates that the end of the attribute table was reached.

**ATTR\_GET**

Query the values of attribute(s) of a specified active scanner. Refer to the *Zebra Scanner SDK Attribute Dictionary* (p/n 72E-149786-XX) for attribute numbers, types and possible values.

**Opcode**

SBT\_RSM\_ATTR\_GET

**inXML**

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
  <cmdArgs>
    <arg-xml>
      <attrib_list>535,20004,1,140,392</attrib_list> ? Attribute Numbers
    </arg-xml>
  </cmdArgs>
</inArgs>
```

**outXML**

```
<?xml version="1.0" encoding="UTF-8"?>
<outArgs>
  <scannerID>1</scannerID>           ? Scanner ID of Data Receiving
  <arg-xml>
    <modelnumber>DS670-SR20001ZZR</modelnumber>
    <serialnumber>7116000501003</serialnumber>
    <response>
      <opcode>5001</opcode>         ? Method Response Received
      <attrib_list>
        <attribute>
          <id>535</id>              ? Attribute Number
          <name></name>
          <datatype>S</datatype>    ? Attribute Data Type
          <permission>R</permission> ? Permissions of the Attribute
          <value>27APR07</value>   ? Attribute Value
        </attribute>
        <attribute>
          <id>20004</id>
          <name></name>
          <datatype>S</datatype>
          <permission>R</permission>
          <value>DS6707X4</value>
        </attribute>
        <attribute>
          <id>1</id>
          <name></name>
          <datatype>F</datatype>
          <permission>RWP</permission>
          <value>True</value>
        </attribute>
        <attribute>
          <id>140</id>
          <name></name>
          <datatype>B</datatype>
          <permission>RWP</permission>
          <value>0</value>
        </attribute>
      </attrib_list>
    </response>
  </arg-xml>
</outArgs>
```

(continued on next page)

```

        </attribute>
        <attribute>
            <id>392</id>
            <name></name>
            <datatype>A</datatype>
            <permission>RWP</permission>
            <value>0x01 0x00 0x58 0x55 0x41 0x00 0x0b 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
        </attribute>
    </attrib_list>
</response>
</arg-xml>
</outArgs>

```

**NOTES**

- Execution of the method results in sending `ATTRIBUTE_GET` RMD protocol command(s) via `SSI_MGMT_COMMAND` SSI protocol command.
- The method should be executed successfully if all required attributes are requested from a scanner through a set of `ATTRIBUTE_GET` RMD commands and responses for all issued `ATTRIBUTE_GET` RMD commands are received via `SSI_MGMT_COMMAND` SSI packets from a scanner.
- The successful execution of the method does not guarantee that all requested attributes are received from a scanner if some of requested attributes are not supported by a specific scanner.
- The method supports the query of maximum 100 attributes values. If more than 100 attribute values are requested, the "Execute Command" API returns `SBT_RESULT_INVALID_PARAMS` to indicate that input parameters are not valid.

**ATTR\_GET\_OFFSET**

Query the value of an attribute of a specified active scanner. The method provides the ability to retrieve string/array attribute values that do not fit within a single RMD protocol packet. It is the responsibility of the application to determine what the starting offset is. Refer to the *Zebra Scanner SDK Attribute Dictionary* (p/n 72E-149786-XX) for attribute numbers, types, and possible values.

**Opcode**

SBT\_RSM\_ATTR\_GET\_OFFSET

**inXML**

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
  <cmdArgs>
    <arg-xml>
      <attribute>
        <id>6002</id>                 ? Attribute Number
        <offset>200</offset>         ? Starting Offset
      </attribute>
    </arg-xml>
  </cmdArgs>
</inArgs>
```

**outXML**

```
<?xml version="1.0" encoding="UTF-8"?>
<outArgs>
  <scannerID>1</scannerID>           ? Scanner ID of Data Receiving
  <arg-xml>
    <modelnumber>DS670-SR20001ZZR</modelnumber>
    <serialnumber>7116000501003</serialnumber>
  </arg-xml>
  <response>
    <opcode>5003</opcode>            ? Method Response Received
    <attrib_list>
      <attribute>
        <id>6002</id>                 ? Attribute Number
        <name></name>
        <datatype>A</datatype>       ? Attribute Data Type
        <permission>R</permission>   ? Permissions of the Attribute
        <length>3936</length>        ? Total Length of Value
        <offset>200</offset>         ? Starting Offset
        <value_length>227</value_length> ? Length of Received Value
        <value>                       ? Received Attribute Value
          0x00 0x00 0x00 0x41 0x00 0x00 0x02 0x02 0x00 0x01 0x00 0x0A 0x01 0x0F 0x00 0x00
0x0A 0x14 0x01 0x00 0x01 0x01 0x01 0x0A 0x00 0x0D 0x02 0x01 0x02 0x01 0x08 0xB4 0x4C 0x32
0x03 0x01 0x00 0x00 0x00 0x00 0x02 0x00 0x02 0x0A 0x00 0x01 0x00 0x05 0x05 0x05 0x01 0x0F
0x03 0x21 0x11 0x0E 0x00 0x01 0x1E 0x01 0x0A 0xFF 0x03 0x06 0x00 0x02 0x00 0x03 0x00 0x00
0x00 0x00 0x00 0x01 0x03 0x64 0x01 0x50 0x00 0x32 0x32 0x00 0x08 0x00 0x01 0x00 0x03 0x01
0x01 0x28 0x00 0x00 0x01 0x01 0x00 0x01 0x01 0x02 0x02 0x02 0x02 0x02 0x01 0x02 0x01 0x1E
0x01 0x00 0x01 0x03 0xA0 0x00 0x64 0x00 0x00 0x00 0x00 0x00 0xDF 0x01 0xEF 0x02 0x90 0x01
0x64 0x00 0x4E 0x01 0xFF 0xFF 0xFF 0xFF 0xB4 0x00 0x00 0x00 0xC8 0x00 0x00 0x00 0x06 0x00
0x01 0x00 0x42 0x54 0x53 0x00 0x0C 0x41 0xE5 0xFD 0xF0 0xB6 0xCA 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x07 0x07 0x07 0x07 0x07
```

(continued on next page)

```
        </value>
      </attribute>
    </attrib_list>
  </response>
</arg-xml>
</outArgs>
```

**NOTES**

- Execution of the method results in sending one `ATTRIBUTE_GET_OFFSET` RMD protocol command via `SSI_MGMT_COMMAND` SSI protocol command.
- The method is supposed to be executed successfully if a required attribute is requested from a scanner through `ATTRIBUTE_GET_OFFSET` RMD command and a response for an issued `ATTRIBUTE_GET_OFFSET` RMD command is received via `SSI_MGMT_COMMAND` SSI packet from a scanner.

## ATTR\_SET

Set the values of the attribute(s) of a specified scanner. The attribute(s) set using this method is lost after the next power down. Refer to the *Zebra Scanner SDK Attribute Dictionary* (p/n 72E-149786-XX) for attribute numbers, types and possible values.

### Opcode

SBT\_RSM\_ATTR\_SET

### inXML

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
  <cmdArgs>
    <arg-xml>
      <attrib_list>
        <attribute>
          <id>1</id>                 ? Attribute Number
          <datatype>F</datatype>    ? Attribute Data Type
          <value>False</value>      ? Attribute Value
        </attribute>
        <attribute>
          <id>2</id>                 ? Attribute Number
          <datatype>S</datatype>    ? Attribute Data Type
          <value>Symbol</value>     ? Attribute Value
        </attribute>
      </attrib_list>
    </arg-xml>
  </cmdArgs>
</inArgs>
```

### outXML

Not used.

### ✓ NOTES

- Execution of the method results in sending ATTRIBUTE\_SET RMD protocol command(s) via SSI\_MGMT\_COMMAND SSI protocol command.
- The method is supposed to be executed successfully if ATTRIBUTE\_SET RMD protocol commands for all required attributes were sent to a scanner and responses for all issued ATTRIBUTE\_SET RMD commands are received via SSI\_MGMT\_COMMAND SSI packets from a scanner.
- The method supports setting of maximum 100 attributes values. If more than 100 attribute values are requested to be set, the "Execute Command" API returns SBT\_RESULT\_INVALID\_PARAMS to indicate that input parameters are not valid.

## ATTR\_STORE

Store the values of attribute(s) of a specified scanner. Attribute(s) store using this method are persistent over power down and power up cycles. Refer to the *Zebra Scanner SDK Attribute Dictionary* (p/n 72E-149786-XX) for attribute numbers, types and possible values.

### Opcode

SBT\_RSM\_ATTR\_STORE

### inXML

```
<inArgs>
  <scannerID>1</scannerID>           ? Specified Scanner ID
  <cmdArgs>
    <arg-xml>
      <attrib_list>
        <attribute>
          <id>1</id>                 ? Attribute Number
          <datatype>F</datatype>     ? Attribute Data Type
          <value>False</value>       ? Attribute Value
        </attribute>
        <attribute>
          <id>2</id>                 ? Attribute Number
          <datatype>S</datatype>     ? Attribute Data Type
          <value>Symbol</value>      ? Attribute Value
        </attribute>
      </attrib_list>
    </arg-xml>
  </cmdArgs>
</inArgs>
```

### outXML

Not used.



#### NOTES

- Execution of the method results in sending ATTRIBUTE\_STORE RMD protocol command(s) via SSI\_MGMT\_COMMAND SSI protocol command.
- The method should execute successfully if ATTRIBUTE\_STORE RMD protocol commands for all required attributes were sent to a scanner and responses for all issued ATTRIBUTE\_STORE RMD commands are received via SSI\_MGMT\_COMMAND SSI packets from a scanner.
- The method supports storing of maximum 100 attributes values. If more than 100 attribute values are requested to be stored, the "Execute Command" API returns SBT\_RESULT\_INVALID\_PARAMS to indicate that input parameters are not valid.



# INDEX

## A

accessing the SDK	3-2
application programming interface definition	
constants	4-2
functions	4-9
methods	4-20
notifications	4-17
types	4-7

## B

beep codes	4-5
------------	-----

## C

chapter descriptions	ix
configuring xcode project	2-7
connectivity management	3-5
constants	
opcodes	
SBT_DEVICE_SCAN_DISABLE	4-4
scanner ios	
beep codes	
SBT_BEEPCODE_FAST_WARBLE	4-6
SBT_BEEPCODE_LONG_HIGH_1	4-6
SBT_BEEPCODE_LONG_HIGH_2	4-6
SBT_BEEPCODE_LONG_HIGH_3	4-6
SBT_BEEPCODE_LONG_HIGH_4	4-6
SBT_BEEPCODE_LONG_HIGH_5	4-6
SBT_BEEPCODE_LONG_LOW_1	4-6
SBT_BEEPCODE_LONG_LOW_2	4-6
SBT_BEEPCODE_LONG_LOW_3	4-6
SBT_BEEPCODE_LONG_LOW_4	4-6
SBT_BEEPCODE_LONG_LOW_5	4-6
SBT_BEEPCODE_MIX1_HIGH_LOW	4-6
SBT_BEEPCODE_MIX2_LOW_HIGH	4-6
SBT_BEEPCODE_MIX3_HIGH_LOW_HIGH	4-6
SBT_BEEPCODE_MIX4_LOW_HIGH_LOW	4-6
SBT_BEEPCODE_SHORT_HIGH_1	4-5
SBT_BEEPCODE_SHORT_HIGH_2	4-5
SBT_BEEPCODE_SHORT_HIGH_3	4-5

SBT_BEEPCODE_SHORT_HIGH_4	4-5
SBT_BEEPCODE_SHORT_HIGH_5	4-5
SBT_BEEPCODE_SHORT_LOW_1	4-5
SBT_BEEPCODE_SHORT_LOW_2	4-5
SBT_BEEPCODE_SHORT_LOW_3	4-5
SBT_BEEPCODE_SHORT_LOW_4	4-5
SBT_BEEPCODE_SHORT_LOW_5	4-6
SBT_BEEPCODE_SLOW_WARBLE	4-6
LED codes	
SBT_LEDCODE_AMBER	4-5
SBT_LEDCODE_BLUE	4-5
SBT_LEDCODE_GREEN	4-5
SBT_LEDCODE_RED	4-5
SBT_LEDCODE_YELLOW	4-5
notifications	
SBT_EVENT_BARCODE	4-3
SBT_EVENT_IMAGE	4-3
SBT_EVENT_RAW_DATA	4-3
SBT_EVENT_SCANNER_APPEARANCE	4-3
SBT_EVENT_SCANNER_DISAPPEARANCE	4-3
SBT_EVENT_SESSION_ESTABLISHMENT	4-3
SBT_EVENT_SESSION_TERMINATION	4-3
SBT_EVENT_VIDEO	4-3
opcodes	
SBT_DEVICE_CAPTURE_BARCODE	4-4
SBT_DEVICE_CAPTURE_IMAGE	4-4
SBT_DEVICE_CAPTURE_VIDEO	4-4
SBT_DEVICE_PULL_TRIGGER	4-4
SBT_DEVICE_RELEASE_TRIGGER	4-4
SBT_DEVICE_SCAN_ENABLE	4-4
SBT_RSM_ATTR_GET	4-4
SBT_RSM_ATTR_GET_OFFSET	4-4
SBT_RSM_ATTR_GETALL	4-4
SBT_RSM_ATTR_SET	4-4
SBT_RSM_ATTR_STORE	4-4
SBT_SET_ACTION	4-4
operating modes	
SBT_OPMODE_ALL	4-2
SBT_OPMODE_BTLE	4-2
SBT_OPMODE_MFI	4-2
result codes	
SBT_RESULT_FAILURE	4-2
SBT_RESULT_INVALID_PARAMS	4-2

SBT_RESULT_OPCODE_NOT_SUPPORTED	4-2
SBT_RESULT_RESPONSE_TIMEOUT	4-2
SBT_RESULT_SCANNER_NO_SUPPORT	4-2
SBT_RESULT_SCANNER_NOT_ACTIVE	4-2
SBT_RESULT_SCANNER_NOT_AVAILABLE	4-2
SBT_RESULT_SUCCESS	4-2
scanner models	
SBT_DEVMODEL_INVALID	4-5
SBT_DEVMODEL_RFID_RFD8500	4-5
SBT_DEVMODEL_SSI_CS4070	4-5
SBT_DEVMODEL_SSI_GENERIC	4-5
SBT_DEVMODEL_SSI_RFD8500	4-5
scanner modes	
SBT_CONNTYPE_BTLE	4-3
SBT_CONNTYPE_INVALID	4-3
SBT_CONNTYPE_MFI	4-3
<b>conventions</b>	
notational	X

## F

<b>functions</b>	
sbtBeepControl	4-16
sbtEnableAutomaticSessionReestablishment	4-13
sbtEnableAvailableScannersDetection	4-12
sbtEstablishCommunicationSession	4-12
sbtExecuteCommand	4-14
sbtGetActiveScannersList	4-11
sbtGetAvailableScannersList	4-10
sbtGetVersion	4-9
sbtLedControl	4-15
sbtSetDelegate	4-9
sbtSetOperationalMode	4-9
sbtSubscribeForEvents	4-10
sbtTerminateCommunicationSession	4-12
sbtUnsubscribeForEvents	4-10

## G

<b>guides, related</b>	.ix
------------------------	-----

## I

<b>information, service</b>	.xi
<b>installing</b>	
from Apple App store	2-2
launching	2-2
prerequisites	2-2

## L

<b>launching the app</b>	2-2, 2-3
<b>LED codes</b>	4-5

## M

<b>methods</b>	
ATTR_GET	4-25
ATTR_GET_OFFSET	4-27
ATTR_GETALL	4-24

ATTR_SET	4-29
ATTR_STORE	4-30
DEVICE_CAPTURE_BARCODE	4-22
DEVICE_CAPTURE_IMAGE	4-22
DEVICE_CAPTURE_VIDEO	4-23
DEVICE_PULL_TRIGGER	4-20
DEVICE_RELEASE_TRIGGER	4-20
SCAN_DISABLE	4-21
SCAN_ENABLE	4-21
SET_ACTION	4-23

## N

<b>notational conventions</b>	X
<b>notifications</b>	4-3
sbtEventBarcode	4-18
sbtEventCommunicationSessionEstablished	4-17
sbtEventCommunicationSessionTerminated	4-17
sbtEventImage	4-19
sbtEventScannerAppeared	4-17
sbtEventScannerDisappeared	4-17
sbtEventVideo	4-19

## O

<b>opcodes</b>	4-3
<b>operating modes</b>	4-2

## P

<b>performing operations</b>	
beeper	3-8
LED	3-12
scan capability	3-13
symbolologies	3-14
trigger	3-18

## R

<b>related documents</b>	.ix
<b>requirements, system</b>	1-2
<b>result codes</b>	4-2

## S

<b>scanner ios</b>	
beep codes	4-5
LED codes	4-5
notifications	4-3
opcodes	4-3
operating modes	4-2
result codes	4-2
scanner models	4-5
scanner modes	4-3
<b>scanner models</b>	4-5
<b>scanner modes</b>	4-3
<b>scanners, supported</b>	1-1
<b>screen shots</b>	
scanner app active scanners	2-5
scanner app available scanners	2-4

scanner app barcode	2-5
scanner app initial	2-3
scanner app scanners	2-4
scanner app settings	2-6
scanner app symbologies	2-6
<b>SDK</b>	
accessing	3-2
connectivity management	3-5
event handling	3-2
performing operations	3-8
<b>service information</b>	.Xi
<b>support</b>	.Xi
<b>supported scanners</b>	1-1
<b>system requirements</b>	1-2

## T

<b>types</b>	
ISbtSdkApiDelegate	4-8
SBT_RESULT	4-7
SbtScannerInfo	4-7
SbtSdkApi	4-8
SbtSdkFactory	4-8

## X

<b>xcode, configuring</b>	2-7
---------------------------	-----







Zebra Technologies Corporation  
Lincolnshire, IL U.S.A.  
<http://www.zebra.com>

Zebra and the stylized Zebra head are trademarks of ZIH Corp., registered in many jurisdictions worldwide. All other trademarks are the property of their respective owners.

©2015 ZIH Corp and/or its affiliates. All rights reserved.

