

DENSO



高出力ハンディスキャナ

DENSO SP1 SDK for Android 解説書

Copyright © 2018 DENSO WAVE INCORPORATED
All rights reserved.

この取扱説明書の著作権は、株式会社デンソーウェーブにあります。

QRコード、iQRコード、SQRC および QBdirect は(株)デンソーウェーブの登録商標です。
Bluetooth®は、商標権利者が所有しており、デンソーウェーブはライセンスに基づき使用しています。

その他、本書に掲載されている会社や製品は、一般に各社の商標または登録商標です。尚、本文中では、™、®マークはすべてについては明記していません。

目次

0. はじめに	4
1. 注意事項	5
1.1. RF タグの読み取りについて	5
1.2. スキャナからの応答について	5
1.3. SP1 のファームウェア更新 API 実装について	5
2. 開発環境	8
3. API 使用の流れ	9
3.1. SP1 との接続方法の選択	9
1:1 運用	9
n:n 運用	10
4. 共通 API	11
4.1. 概要	11
4.2. 共通API	12
CommScanner#upgradeFirmware	12
CommScanner#buzzer	13
CommScanner#setLED	14
CommScanner#getSerialNum	15
CommScanner#getPartNum	16
CommScanner#getRemainingBattery	17
CommScanner#setParams	18
CommScanner#saveParams	19
CommScanner#getParams	20
CommScanner#setBtSettings	21
CommScanner#getBtSettings	22
CommScanner#init	23
CommException#getErrorCode	24
CommScanner#getType	25
CommScanner#getVersion	26
CommScanner#addStatusListener	27
CommScanner#removeStatusListener	28
ScannerStatusListener#onScannerStatusChanged	29
CommStatusChangedEvent#getStatus	30
CommManager#addAcceptStatusListener	31
CommManager#removeAcceptStatusListener	32
CommManager#startAccept	33
CommManager#endAccept	34
ScannerAcceptStatusListener#OnScannerAppeared	35
CommScanner#claim	36
CommScanner#close	37
CommManager#getScanners	38
CommScanner#getBTAddress	39
CommScanner#getBTLocalName	40
CommScanner#getModel	41
CommScanner#getBarcodeScanner	42
CommScanner#getRFIDScanner	43
CommScanner#addKeyStatusListener	44
CommScanner#removeKeyStatusListener	45
ScannerKeyStatusListener#onScannerKeyStatusChanged	46
CommKeyStatusChangedEvent#getKeyName	47
CommKeyStatusChangedEvent#getKeyStatus	48

CommScanner#getCurrentKeyStatus.....	49
CommScanner#getRegion.....	50
サンプルコード1	51
CommScannerParams クラス.....	52
CommScannerBtSettings クラス.....	54
5. RFID	55
5.1. 概要	55
5.1.1. RFID データ	55
5.1.2. バッチモード	55
5.2. RFID 関連 API	56
RFIDScanner#openInventory	56
RFIDScanner#openInventory(Index)	57
RFIDScanner#openRead(bank, addr, size, pwd).....	58
RFIDScanner#openRead(bank, addr, size, pwd, UII).....	59
RFIDScanner#openRead(bank, addr, size, pwd, index)	60
RFIDScanner#close	61
RFIDScanner# writeOneTag	62
RFIDScanner# lockOneTag.....	64
RFIDScanner# killOneTag.....	66
RFIDScanner#setSettings.....	67
RFIDScanner#getSettings.....	68
RFIDScanner#setAutoLinkProfile.....	69
RFIDScanner#getAutoLinkProfile.....	70
RFIDScanner#setFilter	71
RFIDScanner#clearFilter.....	72
RFIDScanner#clearDoubleReadingBuffer.....	73
RFIDScanner#setDataDelegate.....	74
RFIDDataDelegate#onRFIDDataReceived.....	75
RFIDDataReceivedEvent#getRFIDData	76
RFIDData#getData	77
RFIDData#getRSSI.....	78
RFIDData#getPC	79
RFIDData#getAntenna.....	80
RFIDData#getPolarization.....	81
RFIDData#getUII	82
RFIDData#getIndex	83
RFIDData#getResult.....	84
RFIDData#getCh	85
RFIDData#getPhase	86
RFIDScanner#getCount.....	87
RFIDScanner#pullData.....	88
RFIDScanner#setResponse	89
RFIDScanner#getResponse	90
RFIDException#getErrorCode	91
サンプルコード 2	93
RFIDScannerFilter クラス.....	95
RFIDScannerSettings クラス	96
RFIDScannerResponse クラス	99
6. バーコード	100
6.1. 概要	100
6.1.1. 読み取り許可	100
6.1.2. バーコードデータ	101
6.1.3. SQRC	101
6.2. バーコード関連 API	102

BarcodeScanner#openReader	102
BarcodeScanner#closeReader	103
BarcodeScanner#setSettings.....	104
BarcodeScanner#getSettings.....	105
BarcodeScanner#setDataDelegate.....	106
BarcodeDataDelegate#onBarcodeDataReceived.....	107
BarcodeDataReceivedEvent#getBarcodeData	108
BarcodeData#getData	109
BarcodeData#getSymbologyDenso.....	110
BarcodeData#getSymbologyAim	111
BarcodeData#getPrivateDataLength	114
BarcodeException#getErrorCode	115
サンプルコード 3	116
BarcodeScannerSettings クラス	118

0. はじめに

このマニュアルは、RFID とバーコード読み取りなどの SP1 の機能を使ったアプリケーションを開発するための解説書です。

SDK 同梱のライブラリで使用可能な機能と、今後対応予定の機能を記載しています。

対応予定の機能は*対応予定と記載してあります。対応予定となっている機能でも、一部動作することがありますが、動作保障するものではありません。

本書記載の内容は予告なしに変更することがあります。

また、本書は Android および Android Studio を用いたプログラミング経験者を対象に書かれています。Android 製品の使い方や開発方法、各言語の文法に関しましては、市販図書を参考にしてください。

1. 注意事項

1.1. RF タグの読み取りについて

RF タグとの通信においては非接触通信を行う特性上、次のような点を考慮していただく必要があります。

- (1) RF タグが壊れていて応答できない、電波干渉が発生し RF タグに十分な電力供給が行えない等の理由により、読み取りエリア内に RF タグが存在しても、読み取れない場合があります。
- (2) 読み取り中に、意図しない RF タグが存在する場合に、それを読み取ることがあります。
- (3) 読み取り操作を繰り返す場合、前回読み取った RF タグが読み取り範囲内に残っていると、再度読み取る場合があります。
- (4) 同一の UII を持つ RF タグは、複数存在しても1つの RF タグとして扱われます。

上記のことから、必要枚数分の読み取り完了にて終了判断を行う等の処理を行うと期待と異なる場合があります。そのため上位アプリでは RF タグの読み取り時に必要に応じて以下のような対処をお願いいたします。

(1)～(3)に対して

- ・ 読み取った RF タグ数、内容の確認が可能なシステムを構築し、オペレータが読み取り結果(RF タグ数、内容)を確認してください。
- ・ 読めなかった RF タグだけを別途かざして読み取る操作が可能、もしくは別手段で登録できるようにしてください。

(4)に対して

- ・ UII データ内容が重複する RF タグを利用しないでください。

1.2. スキャナからの応答について

ご使用のホスト端末や環境によっては、スキャナからの応答が遅延する場合があります。事前に問題なく通信が行えることを確認した上で、使用してください。

1.3. SP1 のファームウェア更新 API 実装について

多台数で運用する場合、SP1 のファームウェア更新機能(CommScanner#upgradeFirmware)の実装を推奨します。SP1 ファームウェアの改善がおこなわれた際、運用を止めずに書き換えが可能となり、書き換えコストの低減が図れます。

■更新履歴

版数	変更内容	対応 SDK	対応 OS
第1版	-		
第2版	注意事項を追加 [バーコード] BarcodeScannerSettings クラスの QR コード設定時の注意事項を追加 [RFID] 拡張省電力モード追加 getCount と pullData の例外 ErrorCode.SCANNER_CHARGING を削除。充電中でも動作可能 電源オン中二度読み防止実施時の条件を変更	1.0.2	1.03
第3版	サンプルコード 2 を変更(getCommand())を削除) [共通 API] CommScanner#upgradeFirmware を追加 [RFID] バッチモードを追記 Read/Write 時の出力設定に関する注記を追加 RFIDData#getResult を追加 RFIDScanner# writeOneTag の説明追加 [バーコード] SQRC 読取関連設定値を追加	1.0.4	1.05
第4版	[共通 API] CommScanner#addKeyStatusListener を追加 CommScanner#removeKeyStatusListener を追加 ScannerKeyStatusListener#onScannerKeyStatusChanged を追加 CommKeyStatusChangedEvent#getKeyName を追加 CommKeyStatusChangedEvent#getKeyStatus を追加 CommScanner#getCurrentKeyStatus を追加	1.0.5	1.06
第5版	[RFID] RFIDData#getCh を追加 RFIDData#getPhase を追加 RFIDScanner#setResponse を追加 RFIDScanner#getResponse を追加 RFIDScannerResponse クラスを追加 [共通 API] CommScanner#getRegion を追加 CommScanner#setBtSettings と CommScanner#getBtSettings を有効化 Bluetooth 切断時の設定追加 Bluetooth 設定値を追加	1.0.6	1.11

版数	変更内容	対応 SDK	対応 OS
第 6 版	[API 使用の流れ] 新規作成 [RFID] RFIDScanner#setFilter の説明追加 RFIDScanner#getRSSI の説明追加 RFIDScanner#getUII の説明追加 RFIDScanner#writeOneTag のサンプルプログラム追加 [バーコード] BarcodeScannerSettings クラスの mirrorReflection に対応 BarcodeScannerSettings クラスの備考追加	1.0.7	1.12
第 7 版	[開発環境] ライブラリインポート方法を HP 記載に変更(Android Studio 仕様変更のため) [API 使用の流れ] n:n 運用のサンプルコードを変更(現在値を確認してから設定に変更) [RFID] RFIDScanner#setAutoLinkProfile を追加(Autopilot 有効/無効を設定) RFIDScanner#getAutoLinkProfile を追加(Autopilot 有効/無効設定値取得)	1.0.9	1.14

2. 開発環境

■ アプリケーション開発ツール

・Android アプリケーション開発にあたって、下記の開発ツールをご用意ください。

- ・Java SE 開発キット(JDK、Java SE 8 以上推奨)
- ・Android SDK
- ・Android Studio

※上記に挙げたソフトウェアツールの入手方法は、それぞれ公式のウェブサイトをご参照ください。

※開発者は、Android のプログラミングに関する知識があることを前提とします。

■ APIについて

アプリケーションを開発する前に、提供されている「*DENSOScannerSDK.aar*」をプロジェクトにインポートしてください。

必要なライブラリは以下の通りです。

(1) DENSOScannerSDK.aar (Android Studio の場合)

■ ライブラリのインポート

ライブラリのインポート方法は Android Studio の各バージョンの仕様によって手順が異なる可能性があります。

ご利用の環境での手順については、Android Studio の仕様をご確認ください。(参考)

<https://developer.android.com/studio/projects/android-library?hl=ja#psd-add-library-dependency>

※本ライブラリの minSdkVersion 値は 23 です。TargetSdkVersion は 30 です。

■ AndroidManifest.xml について

AndroidManifest.xml に Bluetooth 使用許可を設定してください

```
<uses-permission android:name="android.permission.BLUETOOTH" />  
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

3. API 使用の流れ

以下では接続までの流れを説明します。

3.1. SP1 との接続方法の選択

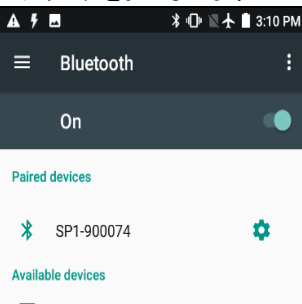
現場の運用形態に応じて接続方法が選択可能です。必ずしもここで記載する方法に従う必要はありません。実際の運用に応じて対応してください。

1:1 運用 SP1 と Android デバイスが 1 台ずつ対応している運用(組み合わせを変えない運用)

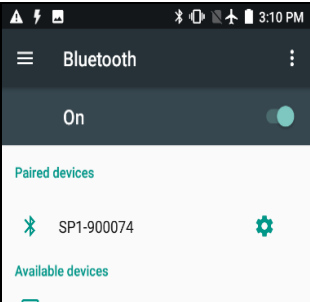
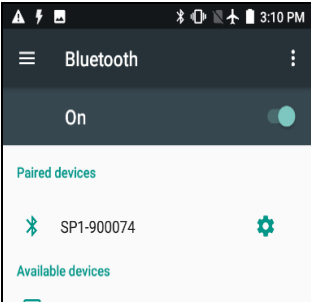
i)初回接続時、SDK に同梱されている接続先登録アプリを使用する場合

初回接続(工場出荷状態)	初回以降の接続
接続先登録アプリで接続します  ※工場出荷状態の SP1 の Bluetooth モードは AUTO となります AUTO は初回接続時 SLAVE、 初回以降は MASTER として動作します。 接続先登録アプリでペアリング→接続をおこなうことで 次回 SP1 は MASTER(設定値は AUTO のまま) として動作します。	<pre>addAcceptStatusListener(listener); // リスナー登録 startAccept(); //待ち受け開始</pre> ↓ SP1 が Android デバイスに対して接続を試みると OnScannerAppeared イベントが通知されます OnScannerAppeared イベント内で下記処理をおこないます <pre>endAccept(); // 待ち受け終了 scanner.claim(); // 接続確立 removeAcceptStatusListener(listener); // リスナー削除</pre> ↓ 接続確立

ii)初回接続時、Android 設定画面にある Bluetooth 設定を使用する場合

初回接続(工場出荷状態)	初回以降の接続
接続したい Android の Bluetooth 設定画面で ペアリングをおこないます  ↓ <pre>// ペアリング済みリストを取得 List<CommScanner> listCommScanner; listCommScanner = CommManager.getScanners(); if (listCommScanner != null) { for (CommScanner scanner: listCommScanner) { //SP1 デバイス名が目的のデバイス名と一致するか確認 if (scanner.getBTLocalName().equals(name)) { scanner.claim(); // 接続確立 } } }</pre> ※工場出荷状態の SP1 の Bluetooth モードは AUTO となります AUTO は初回接続時 SLAVE、 初回以降は MASTER として動作します。 getScanners→claim で接続をおこなうことで 次回 SP1 は MASTER(設定値は AUTO のまま) として動作します。	<pre>addAcceptStatusListener(listener); // リスナー登録 startAccept(); //待ち受け開始</pre> ↓ SP1 が Android デバイスに対して接続を試みると OnScannerAppeared イベントが通知されます OnScannerAppeared イベント内で下記処理をおこないます <pre>endAccept(); // 待ち受け終了 scanner.claim(); // 接続確立 removeAcceptStatusListener(listener); // リスナー削除</pre> ↓ 接続確立

n:n 運用 SP1 と Android デバイスの組み合わせが固定されない運用

初回接続(工場出荷状態)	初回以降の接続
接続したい Android の Bluetooth 設定画面でペアリングをおこないます  ↓ <pre>// ペアリング済みリストを取得 List<CommScanner> listCommScanner; listCommScanner = CommManager.getScanners(); if (listCommScanner != null) { for (CommScanner scanner: listCommScanner) { //SP1 デバイス名が目的のデバイス名と一致するか確認 if (scanner.getBTLocalName().equals(name)) { scanner.claim(); // 接続確立 // SP1 の Bluetooth 設定を取得 CommScannerBtSettings btSet=scanner.getBtSettings(); If(btSet.mode != CommScannerBtSettings.Mode.SLAVE){ // SP1 を SLAVE に btSet.mode = CommScannerBtSettings.Mode.SLAVE; scanner.setBtSettings(btSet); // 設定を SP1 へ反映 } } } }</pre> ※工場出荷状態の SP1 の Bluetooth モードは AUTO となります AUTO は初回接続時 SLAVE、 初回以降は MASTER として動作します。 接続確立後、常時 SLAVE モードに変更して 次回接続時にも SLAVE で動作するように変更します。	接続したい Android の Bluetooth 設定画面でペアリングをおこないます  ↓ <pre>// ペアリング済みリストを取得 List<CommScanner> listCommScanner; listCommScanner = CommManager.getScanners(); if (listCommScanner != null) { for (CommScanner scanner: listCommScanner) { //SP1 デバイス名が目的のデバイス名と一致するか確認 if (scanner.getBTLocalName().equals(name)) { scanner.claim(); // 接続確立 // SP1 の Bluetooth 設定を取得 CommScannerBtSettings btSet=scanner.getBtSettings(); If(btSet.mode != CommScannerBtSettings.Mode.SLAVE){ // SP1 を SLAVE に btSet.mode = CommScannerBtSettings.Mode.SLAVE; scanner.setBtSettings(btSet); // 設定を SP1 へ反映 } } } }</pre> ※工場出荷状態の端末と接続される可能性もあるため、 Bluetooth モードを確認して、設定が SLAVE モードでない場合、 SLAVE モードにしています。

4. 共通 API

4.1. 概要

SP1 デバイスの情報取得、Bluetooth の接続、RFID/Barcode モード共通の設定などを行う API です。

4.2. 共通API

CommScanner#upgradeFirmware

【機能】

SP1 のファームウェアを更新します。SY3 ファイルをオープンしてストリームを入力してください。

同期関数となります。更新が完了するまで処理が返ってきません。

正常終了後に、自動的に SP1 がリブートします。

その際に接続が切れるため、再度接続処理を実行してください。

通信環境の良い場所を実施してください。

多台数で運用する場合、実装するようにしてください。

SP1 ファームウェアの改善がおこなわれた際、運用を止めずに書き換えが可能となり、

書き換えコストの低減も図れます。

【書式】

```
void upgradeFirmware (InputStream in);
```

【引数】

in

SP1 の Firmware ファイルのストリーム

【戻り値】

なし

【例外】

CommException

CommException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode.INVALID_COMMAND	無効オプション指定時
ErrorCode.COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode.INVALID_STATUS	不正状態エラー
ErrorCode.INVALID_MODEL	Firmware ファイルが 当該スキャナの機種と一致しない
ErrorCode.INVALID_VERSION	ダウングレード時
ErrorCode.INVALID_FILE	Firmware ファイルが不正
ErrorCode.SCANNER_WRITE_FIRMWARE	Firmware 書き込みエラー発生
ErrorCode.SCANNER_INCOMPLETE_UPGRADE	UPGRADE 用通信エラーなどで UPGRADE が 完了しない状態で終了したとき
ErrorCode.NOT_SUPPORT_COMMAND	未サポートコマンド

CommScanner#buzzer

【機能】

ブザーを鳴動させます

【書式】

```
void buzzer (  
    CommBuzzerType b_type  
);
```

【引数】

b_type

CommBuzzerType.B1

ブザーを鳴動させます。

ブザー鳴動時間は、システム用ブザー鳴動時間の設定に従います。

またブザー音量およびブザー高さは、システム用ブザー音量およびブザー高さの設定に従います。

CommBuzzerType.B2

ブザーを約 120ms 鳴動させます。

ブザー音量およびブザー高さは、システム用ブザー音量およびブザー高さの設定に従います。

CommBuzzerType.B3

ブザーを約 240ms 鳴動させます。

ブザー音量およびブザー高さは、システム用ブザー音量およびブザー高さの設定に従います。

【戻り値】

なし

【例外】

CommException

CommException#getErrorCode()でエラーを取得できます。

本 API で発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_COMMAND	無効オプション指定時
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

CommScanner#setLED

【機能】

指定 LED を指定色で点灯させます。組み合わせによっては指定色で点灯できないこともあります。
SP1 では下記組み合わせのみ有効です。

- ・表示 LED1 (SP1 では電源・充電 LED) 赤 橙 緑
- ・表示 LED2 (SP1 ではペアリング LED) 青
- ・表示 LED3 (SP1 では読取・MODE LED) 赤 青

【書式】

```
void setLED (  
    CommLEDType led_type  
    CommLEDColor color  
);
```

【引数】

led_type

- CommLEDType.LED1
表示 LED1 を約 500ms 点灯させる
- CommLEDType.LED2
表示 LED2 を約 500ms 点灯させる
- CommLEDType.LED3
表示 LED3 を約 500ms 点灯させる

color

- CommLEDColor.RED
赤色で点灯させる
- CommLEDColor.ORANGE
橙色で点灯させる
- CommLEDColor.BLUE
青色で点灯させる
- CommLEDColor.GREEN
緑色で点灯させる

【戻り値】

なし

【例外】

- CommException
CommException#getErrorCode()でエラーを取得できます。
本 API で発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode.INVALID_COMMAND	無効オプション指定時
ErrorCode.INVALID_STATUS	不正状態エラー
ErrorCode.NOT_SUPPORT_COMMAND	未サポートコマンド

CommScanner#getSerialNum

【機能】

シリアルナンバーを取得します。

【書式】

```
String getSerialNum ();
```

【引数】

なし

【戻り値】

シリアルナンバー

【例外】

本 API では例外をスローしません(エラー発生しません)。
claim 前に本 API をコールした場合は、API は Null を返します

【備考】

【例】

CommScanner#getPartNum

【機能】

品番を取得します。

【書式】

```
String getPartNum ();
```

【引数】

なし

【戻り値】

品番

【例外】

本 API では例外をスローしません(エラー発生しません)。
claim 前に本 API をコールした場合は、API は Null を返します

【備考】

【例】

CommScanner#getRemainingBattery

【機能】

バッテリー残量を取得します。

【書式】

```
CommBattery getRemainingBattery ();
```

【引数】

なし

【戻り値】

CommBattery.UNDER10
残量 10 パーセント未満

CommBattery.UNDER40
残量 40 パーセント未満

CommBattery.40_AND_OVER
残量 40 パーセント以上

【例外】

CommException
CommException#getErrorCode()でエラーを取得できます。
本 API で発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【例】

CommScanner#setParams

【機能】

共通パラメータを設定します。

【書式】

```
void setParams (  
    CommScannerParams params  
);
```

【引数】

params

共通パラメータ設定の CommScannerParams クラス変数

【戻り値】

なし

【例外】

NullPointerException

params が null の場合

CommException

CommException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_COMMAND	無効オプション指定時
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

【例】

サンプルコード 1 を参照してください

CommScanner#saveParams

【機能】

共通パラメータを保存します。

【書式】

```
void saveParams ();
```

【引数】

なし

【戻り値】

なし

【例外】

CommException

CommException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

CommScanner#getParams

【機能】

共通パラメータを取得します。

【書式】

```
CommScannerParams getParams ();
```

【引数】

なし

【戻り値】

CommScannerParams 現在の共通パラメータ設定値

【例外】

CommException

CommException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

【例】

サンプルコード 1 を参照してください

CommScanner#setBtSettings

【機能】

Bluetooth 設定値を設定します。

【書式】

```
void setBtSettings (  
    CommScannerBtSettings bt_settings  
);
```

【引数】

bt_settings

Bluetooth 設定の CommScannerBtSettings クラス変数

【戻り値】

なし

【例外】

NullPointerException

bt_settings が null の場合

CommException

CommException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_COMMAND	無効オプション指定時
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

【例】

CommScanner#getBtSettings

【機能】

Bluetooth 設定値を取得します。

【書式】

```
CommScannerBtSettings getBtSettings ();
```

【引数】

なし

【戻り値】

CommScannerBtSettings 現在の Bluetooth 設定値

【例外】

CommException

CommException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

CommScanner#init

【機能】

SP1 で保持している設定値の初期化をおこないます。

【書式】

```
void init (  
    CommInitType init_type  
);
```

【引数】

init_type
CommInitType.COMM_PARAMS
 共通パラメータ初期化

CommInitType.BT_PARAMS
 Bluetooth パラメータ初期化

CommInitType.RFID_PARAMS
 RFID パラメータ初期化

CommInitType.BARCODE_PARAMS
 バーコードパラメータ初期化

CommInitType.ALL_PARAMS
 全てのパラメータ初期化

【戻り値】

なし

【例外】

CommException
CommException#getErrorCode()でエラーを取得できます。
発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode.INVALID_COMMAND	無効オプション指定時
ErrorCode.INVALID_STATUS	不正状態エラー
ErrorCode.NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

初期化処理に数秒間かかることがあります。

CommException#getErrorCode

【機能】

発生した例外を取得します。

【書式】

```
CommErrorCode error_code getErrorCode ();
```

【引数】

なし

【戻り値】

ErrorCode. COMMUNICATION_ERROR

スキャナとの通信エラー

ErrorCode. COMMUNICATION_TIMEOUT

スキャナからの応答タイムアウト

ErrorCode. SESSION_NOT_ESTABLISHED

セッション確立エラー

ErrorCode. HANDSHAKE_INVALID_FORMAT

ハンドシェイク不正フォーマット

ErrorCode. HANDSHAKE_NO_REQUIRED_ITEM

未サポートスキャナ

ErrorCode. CLOSE_TIMEOUT

停止タイムアウト

ErrorCode. TOO_MUCH_COMMAND_QUEUE

コマンドキューがあふれたとき

ErrorCode. INVALID_COMMAND

無効オプション指定時

ErrorCode. INVALID_STATUS

claim 前に API を呼んだときなど

ErrorCode. INVALID_RESPONSE

不正な応答がスキャナからあったとき

ErrorCode. NOT_SUPPORT_COMMAND

スキャナが対応していないコマンド送信時

ErrorCode. UNKNOWN

原因不明です。

CommScanner#getType

【機能】

スキャナの種類を取得します。

【書式】

```
CommScannerType getType ();
```

【引数】

なし

【戻り値】

CommScannerType.TYPE_1D

1次元バーコードスキャナ

CommScannerType.TYPE_2D

2次元バーコードスキャナ

CommScannerType.TYPE_2D_LONG

ロングレンジ 2次元バーコードスキャナ

CommScannerType.TYPE_UNKOWN

不明なバーコードスキャナ

CommScannerType.TYPE_RFID

RFID スキャナ

CommScannerType.TYPE_2D_RFID

2次元バーコードとRFIDスキャナ複合機

【例外】

本APIでは例外をスローしません(エラー発生しません)。

claim前に本APIをコールした場合は、APIはNullを返します

【例】

CommScanner#getVersion

【機能】

スキャナのファームウェアバージョンを取得します。

【書式】

```
String getVersion ();
```

【引数】

なし

【戻り値】

バーコードスキャナのファームウェアバージョン

【例外】

本 API では例外をスローしません(エラー発生しません)。
claim 前に本 API をコールした場合は、API は Null を返します

【備考】

【例】

CommScanner#addStatusListener

【機能】

スキャナの状態が変わった時(切断)にスキャナの状態を受け取るためのリスナーを登録します。

【書式】

```
void addStatusListener (ScannerStatusListener listener);
```

【引数】

listener

バーコードスキャナの状態取得用リスナー

【戻り値】

なし

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

CommScanner#removeStatusListener

【機能】

CommScanner#addStatusListener で登録したリスナーを登録解除します。

【書式】

```
void removeStatusListener (ScannerStatusListener listener);
```

【引数】

listener

バーコードスキャナの状態取得用リスナー

【戻り値】

なし

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

ScannerStatusListener#onScannerStatusChanged

【機能】

スキャナ-ホストデバイス間の Bluetooth 接続状態が変更されたときに、通知するイベントハンドラ。

機器が圏外となり切断されたときや、意図しない切断時にも通知されます。

本イベントハンドラは、ScannerStatusListener をオーバーライドし、addStatusListener でリスナー登録したインスタンスに通知されます。

【書式】

```
void onScannerStatusChanged (  
    CommScanner* scanner  
    CommStatusChangedEvent* state  
);
```

【引数】

scanner

Bluetooth 切断された、スキャナクラスのインスタンス

state

状態通知クラスのインスタンス

【戻り値】

なし

【例】

CommStatusChangedEvent#getStatus

【機能】

スキャナの状態を取得します。

【書式】

```
ScannerStatus getStatus ();
```

【引数】

なし

【戻り値】

ScannerStatus.CLAIMED
スキャナが獲得されました。

ScannerStatus.CLOSE_WAIT
クローズ待ちです。
切断検知すると CommScanner#close されるまでこの状態となります

ScannerStatus.CLOSED
スキャナが解放されました。

ScannerStatus.UNKNOWN
不明な状態です。

【例】

CommManager#addAcceptStatusListener

【機能】

スキャナからの接続通知を受け取るためのリスナーを登録します。

【書式】

```
void addAcceptStatusListener (ScannerAcceptStatusListener listener);
```

【引数】

listener

スキャナからの接続通知用リスナー

【戻り値】

なし

【例】

サンプルコード 1 を参照してください

【例外】

本 API では例外をスローしません(エラー発生しません)。

CommManager#removeAcceptStatusListener

【機能】

CommManager#addAcceptStatusListener で登録したリスナーを登録解除します。

【書式】

```
void removeAcceptStatusListener (ScannerAcceptStatusListener listener);
```

【引数】

listener
状態取得用リスナー

【戻り値】

なし

【例】

サンプルコード 1 を参照してください

【例外】

本 API では例外をスローしません(エラー発生しません)。

CommManager#startAccept

【機能】

スキャナからの待ち受けを開始します。

【書式】

```
void startAccept ();
```

【引数】

なし

【戻り値】

なし

【例】

サンプルコード 1 を参照してください

【例外】

本 API では例外をスローしません(エラー発生しません)。

CommManager#endAccept

【機能】

スキャナからの待ち受けを停止します。

【書式】

```
void endAccept ();
```

【引数】

なし

【戻り値】

なし

【例】

サンプルコード 1 を参照してください

【例外】

本 API では例外をスローしません(エラー発生しません)。

ScannerAcceptStatusListener#OnScannerAppeared

【機能】

CommManager#addAcceptStatusListener にて待ち受けを開始し、スキャナから接続されたときに通知するイベントハンドラ。

本イベントハンドラは、ScannerAcceptStatusListener をオーバーライドし、addAcceptStatusListener でリスナー登録したインスタンスに通知されます。

【書式】

```
void OnScannerAppeared (CommScanner* scanner);
```

【引数】

scanner

接続された、スキャナクラスのインスタンス

【戻り値】

なし

【例】

サンプルコード 1 を参照してください

CommScanner#claim

【機能】

SPP または iAP のセッション開始とハンドシェイクをおこないます。

【書式】

```
void claim ();
```

【引数】

なし

【戻り値】

なし

【例外】

CommException

CommException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode. SESSION_NOT_ESTABLISHED	セッション確立エラー
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. HANDSHAKE_INVALID_FORMAT	ハンドシェイク不正フォーマット
ErrorCode. HANDSHAKE_NO_REQUIRED_ITEM	未サポートスキャナ

【例】

サンプルコード 1 を参照してください

CommScanner#close

【機能】

claim で開始したセッションを終了します

【書式】

```
void close ();
```

【引数】

なし

【戻り値】

なし

【例外】

CommException

CommException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. CLOSE_TIMEOUT	停止タイムアウト

【例】

CommManager#getScanners

【機能】

ペアリング済みの CommScanner の List を取得します。※Android の場合。

C

【書式】

```
static List<CommScanner> getScanners ();
```

【引数】

なし

【戻り値】

ペアリング済みの CommScanner インスタンスの List です。

ペアリング済みデバイスがない場合は null が返ります。

【例外】

本 API では例外をスローしません(エラー発生しません)。

【備考】

【例】

CommScanner#getBTAddress

【機能】

当該スキャナの Bluetooth アドレスを取得します。

【書式】

```
String getBTAddress ();
```

【引数】

なし

【戻り値】

Bluetooth アドレス

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

CommScanner#getBTLocalName

【機能】

当該スキャナの Bluetooth ローカルネームを取得します。

【書式】

```
String getBTLocalName ();
```

【引数】

なし

【戻り値】

Bluetooth ローカルネーム

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

CommScanner#getModel

【機能】

当該スキャナのモデル名を取得します。SP1 の場合は SP1-QUBi が取得できます。

【書式】

```
String getModel ();
```

【引数】

なし

【戻り値】

モデル名を返します。

【例外】

本 API では例外をスローしません(エラー発生しません)。
claim 前に本 API をコールした場合は、API は Null を返します。

【例】

CommScanner#getBarcodeScanner

【機能】

バーコードスキャナインスタンスを取得します。

【書式】

```
BarcodeScanner getBarcodeScanner ();
```

【引数】

なし

【戻り値】

CommScanner#claim で生成したバーコードスキャナインスタンスを取得します。

【例外】

本 API では例外をスローしません(エラー発生しません)。
claim 前に本 API をコールした場合は、API は Null を返します。

【例】

```
CommScanner[0].getBarcodeScanner.openReader();
```

CommScanner#getRFIDScanner

【機能】

RFID インスタンスを取得します。

【書式】

```
RFIDScanner getRFIDScanner ();
```

【引数】

なし

【戻り値】

CommScanner#claim で生成した RFID スキャナインスタンスを取得します。

【例外】

本 API では例外をスローしません(エラー発生しません)。
claim 前に本 API をコールした場合は、API は Null を返します。

【例】

```
CommScanner[0].getRFIDScanner.openInventory();
```

サンプルコード 2 を参照してください

CommScanner#addKeyStatusListener

【機能】

スキャナのキー状態が変わった時にキーの状態を受け取るためのリスナーを登録します。

【書式】

```
void addKeyStatusListener (ScannerKeyStatusListener listener);
```

【引数】

listener

バーコードスキャナのキー状態取得用リスナー

【戻り値】

なし

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

CommScanner#removeKeyStatusListener

【機能】

CommScanner#addKeyStatusListener で登録したリスナーを登録解除します。

【書式】

```
void removeKeyStatusListener (ScannerKeyStatusListener listener);
```

【引数】

listener

バーコードスキャナのキー状態取得用リスナー

【戻り値】

なし

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

ScannerKeyStatusListener#onScannerKeyStatusChanged

【機能】

スキャナのキー状態が変更されたときに、通知するイベントハンドラ。
キーの押し下げされたとき、押し下げ状態から離された時に通知されます。
本イベントハンドラは、ScannerKeyStatusListener をオーバーライドし、
addKeyStatusListener でリスナー登録したインスタンスに通知されます。

【書式】

```
void onScannerKeyStatusChanged (  
    CommScanner* scanner  
    CommKeyStatusChangedEvent* keystate  
);
```

【引数】

scanner

キー状態が変化した、スキャナクラスのインスタンス

keystate

キー状態通知クラスのインスタンス

【戻り値】

なし

【例】

CommKeyStatusChangedEvent#getKeyName

【機能】

状態が変化したキーの名前を取得します。

【書式】

```
ScannerKeyName getKeyName ();
```

【引数】

なし

【戻り値】

ScannerKeyName.TRIGGER

トリガキーの状態が変化

ScannerKeyName.UNKNOWN

不明なキーの状態が変化

【例】

CommKeyStatusChangedEvent#getKeyStatus

【機能】

キーが押されたか離されたか取得します。

【書式】

```
ScannerKeyStatus getKeyStatus ();
```

【引数】

なし

【戻り値】

ScannerKeyStatus.RELEASE

キーが離された

ScannerKeyStatus.PRESS

キーが押された

【例】

CommScanner#getCurrentKeyStatus

【機能】

指定キーの現在の状態を取得。同期関数。

【書式】

```
ScannerKeyStatus getCurrentKeyStatus (ScannerKeyName name);
```

【引数】

name

取得したいキー名称。Trigger キーのみ取得可能。

【戻り値】

ScannerKeyStatus.RELEASE

キーが離された

ScannerKeyStatus.PRESS

キーが押された

【例外】

CommException

CommException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_COMMAND	無効オプション指定時
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【例】

CommScanner#getRegion

【機能】

当該スキャナの仕向け地の国コードを取得します。日本モデルの場合は JP が取得できます。

【書式】

```
String getRegion();
```

【引数】

なし

【戻り値】

国コードを返します。

【例外】

本 API では例外をスローしません(エラー発生しません)。

claim 前に本 API をコールした場合は、API は Null を返します。

【備考】

国コード	仕向け地
AU	豪州
CN	中国
EU	欧州
HK	香港、シンガポール、タイ
ID	インドネシア
IL	イスラエル
IN	インド
JP	日本
KR	韓国
MX	メキシコ
MY	マレーシア
PH	フィリピン
RU	ロシア
TW	台湾
US	米国、カナダ
VN	ベトナム

サンプルコード1

```
public class ExecuteRFIDActivity extends AppCompatActivity implements ScannerAcceptStatusListener {

    private RFIDScanner mRfidScanner = null;

    private TextView textGetVersion = null;
    int count = 0;
    String data;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_execute_connect);

        // Start accept
        CommManager.addAcceptStatusListener(this);
        CommManager.startAccept();

    }

    @Override
    public void OnScannerAppeared(CommScanner commScanner) {

        Log.d("ExecuteConnectActivity", " OnScannerAppeared ");
        Log.d("ExecuteConnectActivity", "BT Address=" + commScanner.getBTAddress());

        //Establish connection
        commScanner.claim();

        // Stop accept
        CommManager.endAccept();
        CommManager.removeAcceptStatusListener(this);

        // for disconnect notify
        commScanner.addStatusListener(this);

    }

}
```

CommScannerParams クラス

【機能】

共通パラメータの設定値を格納します。

(✓:サポート)

プロパティ : 説明	設定値		初期値	SP1
CommScannerParams 関連				
powerSave : 低消費設定	true	低消費モード		✓
	false	ノーマル	○	✓
ponBuzzer : パワーオン時ブザー鳴動	true	ブザー鳴動あり	○	
	false	ブザー鳴動なし		
batteryConf : 電源ボタン短押しによる残バッテリー確認許可	true	許可する	○	
	false	許可しない		
buzzerVolume : ブザー音量設定	BuzzerVolume.LOW	小		✓
	BuzzerVolume.MIDDLE	中		✓
	BuzzerVolume.LOUD	大	○	✓
buzzerTone : ブザー音高さ設定 起動音と警告音以外に反映	BuzzerTone.LOW	低音		✓
	BuzzerTone.MEDIUM	中音		✓
	BuzzerTone.HIGH	高音	○	✓
buzzerDuration : ブザー音時間設定 起動音と警告音以外に反映	BuzzerDuration.SHORT	短		✓
	BuzzerDuration.MIDDLE	中	○	✓
	BuzzerDuration.LONG	長		✓
CommScannerParams.autopoweroff 関連				
settings : オートパワーオフ	true	オートパワーオフあり	○	✓
	false	オートパワーオフなし		✓
duration : オートパワーオフ時間	0x5- 0x280	オートパワーオフ時間 (範囲:5 分-640 分)	0x3C (60 分)	✓
CommScannerParams.btbutton 関連				
disconnectPermit : ペアリングボタン 長押しによる切断許可	true	切断許可	○	✓
	false	切断禁止		✓
reConnect : 切断時の設定	ReConnect.REPAIRING	ペアリングボタンで接続待ち		✓
	ReConnect.WAITHOST	ホストからの接続待ち	○	✓
	ReConnect.ANY	毎回接続待ち		✓
CommScannerParams.notification.sound : 読み取り通知音関連				
buzzer : ブザー鳴動設定	Buzzer.ENABLE	有効	○	✓
	Buzzer.DISABLE	無効		✓
CommScannerParams.notification.led : 読み取り通知 LED 関連				
led : LED 点灯設定	true	有効	○	✓
	false	無効		✓

【備考】

reConnect 設定値と切断イベント発生後の状態について

切断イベント reConnect 設定	ペアリングボタン で切断	Mode ボタンでモード切替 (Mode.AUTO の場合)	API で Close (CommScanner#close)	その他 (圏外への移動等)
REPAIRING	切断	接続待ち	切断	切断
WAITHOST	切断	接続待ち	切断	接続待ち
ANY	切断	接続待ち	接続待ち	接続待ち

CommScannerBtSettings クラス

【機能】

Bluetooth の設定値を格納します。

(✓:サポート)

プロパティ : 説明	設定値		初期値	SP1
CommScannerBTSettings 関連				
mode : マスタースレーブ 設定	Mode.MASTER	マスター動作		✓
	Mode.SLAVE	スレーブ動作		✓
	Mode.AUTO	AUTO 動作	○	✓
CommScannerBTSettings.master 関連				
connectedTo : マスター接続先設 定	ConnectedToBtAddress	Bluetooth アドレスでの接続	○	
	ConnectedToLocalName	ローカルネームでの接続		
btAddress : 接続先 Bluetooth アドレス	Bluetooth アドレスを 0-9,A-F の 12 文字の 16 進数文字列で指定	マスター接続先 Bluetooth アドレス	"FFFFFFFFFFFF"	✓
localName : 接続先ローカルネ ーム	ローカルネームを 1 文字から 16 文字までの可変長文字列で、 英数字または記号で指定	マスター接続先 ローカルネーム	"FFFFFFFFFFFF"	
pincode : 接続時 PIN コード	PIN コードを 8 桁以下の英数字 または記号で指定	マスター接続時 PIN コード	"1234" (4 桁)	
tryingTime : 接続試行時間	TryingTime.30SEC	マスター接続試行時間 30 秒	○	✓
	TryingTime.NONE	タイムアウト無し		✓
CommScannerBTSettings.slave 関連				
waitingTime : 接続待ち時間	WaitingTime.2MIN	スレーブ待ち時間 2 分	○	✓
	WaitingTime.4MIN	スレーブ待ち時間 4 分		✓
	WaitingTime.10MIN	スレーブ待ち時間 10 分		✓
	WaitingTime.30MIN	スレーブ待ち時間 30 分		✓
	WaitingTime.NONE	タイムアウト無し		✓
stealth : ステルスモード	true	ステルスモード		
	false	通常モード	○	
pincode : 接続時 PIN コード	PIN コードを 8 桁以下の英数字 または記号で指定	スレーブ接続時 PIN コード	"1234" (4 桁)	

【備考】

マスタースレーブ設定について

Mode.AUTO

初期状態では SP1 がスレーブとして動作し、ホストからの接続を待ちます。

接続すると、マスター接続先 Bluetooth アドレスが自動で設定され、

以降は、SP1 がマスターとして動作し、接続先 Bluetooth アドレスへ接続を試行します。

MODE ボタンを 3 秒以上押すことにより、手動でマスタースレーブを切り替えることができます。

Mode.MASTER

SP1 が常にマスターとして動作し、接続先 Bluetooth アドレス(btAddress)への接続を試行します。

MODE ボタンによるマスタースレーブ切り替えは無効です。

Mode.SLAVE

SP1 が常にスレーブとして動作し、ホストからの接続を待ちます。

MODE ボタンによるマスタースレーブ切り替えは無効です。

5. RFID

5.1. 概要

RFID 読み取りを実行したり、読み取り時の設定などを行う API です。

5.1.1. RFID データ

読み取った RFID データは RFIDData の List 配列に格納されます。

RFIDDataクラスに格納されているRFIDデータの情報を読み出すには、
以下のメソッドを使用します

- ・RFIDData#getData: RFIDデータ取得メソッド

※バーコードデータ長を取得したい場合は、下記のようにlengthメソッドを使用して下さい。

例: int RFIDDataLength = data.length();

5.1.2. バッチモード

RFID 読み取り中に、Bluetooth 切断した場合、バッチモードとして読み取りを継続します。

バッチモードで読み取ったデータは内部メモリに蓄積されます。

再度 Bluetooth 接続すると、読み取りを停止します。

蓄積されたデータは以下のメソッドで取得できます。

- ・RFIDScanner#getCount: バッチモード中に読み取ったタグの枚数取得

- ・RFIDScanner#pullData: バッチモード中に読取ったタグデータの取得

5.2. RFID 関連 API

RFIDScanner#openInventory

【機能】

RFIDScanner#setSettings()で設定された内容に従って、RFID の Inventory を許可します。

設定可能項目は RFIDScannerSettings クラスを参照ください。

バーコードとの同時オープンはできません。

【書式】

```
void openInventory ();
```

【引数】

なし

【戻り値】

なし

【例外】

RFIDException

RFIDException #getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド
ErrorCode. SCANNER_CHARGING	充電中エラー

【例】

サンプルコード 2 を参照してください

RFIDScanner#openInventory(Index)

【機能】

RFIDScanner#setSettings()で設定された内容に従って、RFID の Inventory を許可します。
Inventory をおこないつつ、バッチモードで SP1 に蓄積された引数 Index からのデータを
送信するように SP1 に対して要求します。蓄積データをすべて取得後に、
Inventory で読取ったデータが引き続き送信されます。
バーコードとの同時オープンはできません。

【書式】

```
void openInventory (int index);
```

【引数】

index

この値からの SP1 に蓄積されたタグデータを取得

【戻り値】

なし

【例外】

RFIDException

RFIDException #getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド
ErrorCode. SCANNER_CHARGING	充電中エラー
ErrorCode. INVALID_COMMAND	無効オプション指定時

【例】

RFIDScanner#openRead(bank, addr, size, pwd)

【機能】

RFIDScanner#setSettings()で設定された内容に従って、RFID の Read を許可します。

設定可能項目は RFIDScannerSettings クラスを参照ください。

UII 未指定の Read となります。

バーコードとの同時オープンはできません。

【書式】

```
void openRead (RFIDBank bank, short addr, short size, byte[] pwd);
```

【引数】

bank

RFIDBank.RESERVED Reserved bank

RFIDBank.UII UII bank

RFIDBank.TID TID bank

RFIDBank.USER User bank

addr

読み出し開始位置(2の倍数のみ指定可)。RFID タグは WORD 単位アクセスのため。

size

読み出しサイズ(2の倍数のみ指定可)。RFID タグは WORD 単位アクセスのため。

pwd

アクセスパスワード

【戻り値】

なし

【例外】

RFIDException

RFIDException #getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode.COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode.INVALID_STATUS	不正状態エラー
ErrorCode.NOT_SUPPORT_COMMAND	未サポートコマンド
ErrorCode.SCANNER_CHARGING	充電中エラー
ErrorCode.INVALID_COMMAND	無効オプション指定時

【例】

RFIDScanner#openRead(bank, addr, size, pwd, UII)

【機能】

UII 指定の openRead をおこないます。
バーコードとの同時オープンはできません。

【書式】

```
void openRead (RFIDBank bank, short addr, short size, byte[] pwd, byte[] UII);
```

【引数】

bank

RFIDBank.RESERVED	Reserved bank
RFIDBank.UII	UII bank
RFIDBank.TID	TID bank
RFIDBank.USER	User bank

addr

読み出し開始位置(2の倍数のみ指定可)。RFID タグは WORD 単位アクセスのため。

size

読み出しサイズ(2の倍数のみ指定可)。RFID タグは WORD 単位アクセスのため。

pwd

アクセスパスワード

UII

指定 UII

RFIDScanner#openRead(bank, addr, size, pwd, index)

【機能】

UII 未指定の Read をおこないつつ、バッチモードで SP1 に蓄積された引数 Index からのデータを送信するように SP1 に対して要求します。蓄積データをすべて取得後に、Read で読取ったデータが引き続き送信されます。
バーコードとの同時オープンはできません。

【書式】

```
void openRead (RFIDBank bank, short addr, short size, byte[] pwd, int index);
```

【引数】

bank

RFIDBank.RESERVED	Reserved bank
RFIDBank.UII	UII bank
RFIDBank.TID	TID bank
RFIDBank.USER	User bank

addr

読み出し開始位置(2の倍数のみ指定可)。RFID タグは WORD 単位アクセスのため。

size

読み出しサイズ(2の倍数のみ指定可)。RFID タグは WORD 単位アクセスのため。

pwd

アクセスパスワード

index

この値からの SP1 に蓄積されたタグデータを取得

RFIDScanner#close

【機能】

openInventory と openRead を close します。

他のアプリケーションが RFID 機能を利用できるように、openInventory と openRead 実行後は、必ず、このメソッドを実行してください。

【書式】

```
void close ();
```

【引数】

なし

【戻り値】

なし

【例外】

RFIDException

RFIDException #getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【例】

サンプルコード 2 を参照してください

RFIDScanner# writeOneTag

【機能】

指定した UII のタグ 1 枚に対して write をおこないます。
同期関数となります。書き込みエラー時は例外を投げます。
対応トリガモードはオートオフモードと連続読み取りモード 2 となります。
バーコードオープン時は使用できません。

【書式】

```
void writeOneTag (  
    RFIDBank bank, short addr, short size, byte[] pwd,  
    byte[] data, byte[] UII, long timeout  
);
```

【引数】

bank

RFIDBank.RESERVED	Reserved bank
RFIDBank.UII	UII bank
RFIDBank.TID	TID bank
RFIDBank.USER	User bank

addr

各 bank 先頭からの書き込み開始位置(2 の倍数のみ指定可)。RFID タグは WORD 単位アクセスのため。
UII bank には PC も含まれます。

size

書き込みサイズ

pwd

アクセスパスワード

data

書き込みデータ(2 の倍数長のデータのみ指定可)。RFID タグは WORD 単位アクセスのため。

UII

書き込み対象の UUI

timeout

タイムアウト値 duration(単位は msec)

【戻り値】

なし

【例外】

RFIDException

RFIDException #getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode.COMMUNICATION_TIMEOUT	スキヤナからの応答タイムアウト
ErrorCode.INVALID_STATUS	不正状態エラー
ErrorCode.NOT_SUPPORT_COMMAND	未サポートコマンド
ErrorCode.SCANNER_CHARGING	充電中エラー
ErrorCode.INVALID_COMMAND	無効オプション指定時
ErrorCode.ACCESS_OUT_OF_RANGE_MEMORY	メモリ範囲外へのアクセス
ErrorCode.SCANNER_ACCESS_LOCK_MEMORY	ロックメモリへのアクセス
ErrorCode.SCANNER_POWER_SHORTAGE_OF_WRITE_MEMORY	メモライト電力不足
ErrorCode.SCANNER_WRITE_LOCK_KILL_TIMEOUT	時間内に Write or Lock or Kill が完了しなかったとき
ErrorCode.SCANNER_WRITE_LOCK_KILL_UNKNOWN	Write or Lock or Kill エラー (理由不明)
ErrorCode.SCANNER_TRIGGER_MODE	トリガモードエラー (未対応のトリガモードで WRITE or LOCK or KILL 実行)

【例】

[illegible]

```
byte[] pass = {0,0,0,0}; // パスワード認証無し
```

```
byte[] uii = {0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11};
```

```
byte[] after_uui = {0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22};
```

```
short size = (short)after uii.length;
```

```
short addr = 4; // UII BANK 0~3 バイトは CRC,PC 領域。4 バイト目から EPC 領域。
```

```
try {
```

```
// 10 秒のタイムアウトで uii から after uii へ UI bank 書き換え
```

```
scanner.getRFIDScanner().writeOneTag(RFIDScannerSettings.RFIDBank.U11,addr,size,pass,after uii,uii,10000);
```

```
    } catch (Exception e) {
```

```
e.printStackTrace();
```

}

RFIDScanner# lockOneTag

【機能】

指定した UII のタグ 1 枚に対して Lock をおこないます
同期関数となります。エラー時は例外を投げます
バーコードオープン時は使用できません。

【書式】

```
void lockOneTag (byte target, RFIDLock type, byte[] pwd, byte[] UII, long timeout);
```

【引数】

target

Lock 対象

ビットフィールドで指定(複数同時設定可能)

01 : キルパスワード

02 : アクセスパスワード

10 : UII Bank

20 : TID Bank

40 : User Bank

type

RFIDLock.UNLOCK	Unlock
RFIDLock.LOCK	Lock
RFIDBank.PERMANENT_UNLOCK	永久 Unlock
RFIDBank.PERMANENT_LOCK	永久 Lock

pwd

アクセスパスワード

UII

Lock 対象の UII

timeout

タイムアウト値 duration(単位は msec)

【戻り値】

なし

【例外】

RFIDException

RFIDException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode.COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode.INVALID_STATUS	不正状態エラー
ErrorCode.NOT_SUPPORT_COMMAND	未サポートコマンド
ErrorCode.SCANNER_CHARGING	充電中エラー
ErrorCode.INVALID_COMMAND	無効オプション指定時
ErrorCode.SCANNER_ACCESS_LOCK_MEMORY	ロックメモリへのアクセス
ErrorCode.SCANNER_POWER_SHORTAGE_OF_WRITE_MEMORY	メモリライト電力不足
ErrorCode.SCANNER_WRITE_LOCK_KILL_TIMEOUT	時間内に Write or Lock or Kill が完了しなかったとき
ErrorCode.SCANNER_WRITE_LOCK_KILL_UNKNOWN	Write or Lock or Kill エラー (理由不明)
ErrorCode.SCANNER_TRIGGER_MODE	トリガモードエラー (未対応のトリガモードで WRITE or LOCK or KILL 実行)

RFIDScanner# killOneTag

【機能】

指定した UII のタグ 1 枚に対して Kill をおこないます。
同期関数となります。エラー時は例外を投げます。
バーコードオープン時は使用できません。

【書式】

```
void killOneTag (byte[] pwd, byte[] UII, long timeout);
```

【引数】

pwd

Kill パスワード

UII

Kill 対象の UII

timeout

タイムアウト値 duration(単位は msec)

【戻り値】

なし

【例外】

RFIDException

RFIDException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode.COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode.INVALID_STATUS	不正状態エラー
ErrorCode.NOT_SUPPORT_COMMAND	未サポートコマンド
ErrorCode.SCANNER_CHARGING	充電中エラー
ErrorCode.INVALID_COMMAND	無効オプション指定時
ErrorCode.SCANNER_POWER_SHORTAGE_OF_WRITE_MEMORY	メモリアイト電力不足
ErrorCode.SCANNER_WRITE_LOCK_KILL_TIMEOUT	時間内に Write or Lock or Kill が完了しなかったとき
ErrorCode.SCANNER_WRITE_LOCK_KILL_UNKNOWN	Write or Lock or Kill エラー (理由不明)
ErrorCode.SCANNER_TRIGGER_MODE	トリガモードエラー (未対応のトリガモードで WRITE or LOCK or KILL 実行)

RFIDScanner#setSettings

【機能】

RFID 関連の設定をします。

【書式】

```
void setSettings (  
    RFIDScannerSettings settings  
);
```

【引数】

settings

読取関連の設定が設定されている RFIDScannerSettings クラス変数

【戻り値】

なし

【例外】

NullPointerException
settings が null の場合

RFIDException
RFIDException#getErrorCode()でエラーを取得できます。
発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_COMMAND	無効オプション指定時
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

本メソッドで設定された設定値は、次回の RFID 読み取り、書き込み動作に反映されます。

【例】

サンプルコード 2 を参照してください

RFIDScanner#getSettings

【機能】

RFID 関連の設定を取得します

【書式】

```
RFIDScannerSettings getSettings ();
```

【引数】

なし

【戻り値】

RFIDScannerSettings 現在の読取関連の設定

【例外】

RFIDException

RFIDException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

【例】

RFIDScanner#setAutoLinkProfile

【機能】

AutoLinkProfile(Autopilot)動作の有効/無効を設定します

自動的に LinkProfile を変更しながら Inventory をおこなうことが可能となります

有効時は RFIDScannerSettings クラスの LinkProfile 設定値を使用せず自動的に切り替わります

無効時は RFIDScannerSettings クラスの LinkProfile 設定値にしたがって動作します

初期値は無効です

【書式】

```
void setAutoLinkProfile(boolean auto);
```

【引数】

auto

true のときは AutoLinkProfile(Autopilot)有効。false のときは AutoLinkProfile(Autopilot)無効。

【戻り値】

なし

【例外】

RFIDException

RFIDException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

本メソッドで設定された設定値は、次回 Inventory 動作に反映されます。

Inventory 動作時のみ本設定値に従い動作します。

RFIDScanner#getAutoLinkProfile

【機能】

AutoLinkProfile(Autopilot)動作が有効か無効かの設定値を取得します

【書式】

```
boolean getAutoLinkProfile();
```

【引数】

なし

【戻り値】

true のときは AutoLinkProfile(Autopilot)有効。false のときは AutoLinkProfile(Autopilot)無効

【例外】

RFIDException

RFIDException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

RFIDScanner#setFilter

【機能】

Filter の設定をします。設定できるフィルタ数は最大 6 個までとなります。

【書式】

```
void setFilter (  
    RFIDScannerFilter filter[ ], RFIDLogicalOpe l_ope  
);
```

【引数】

filter

フィルタ制御関連が設定されている RFIDScannerFilter クラス変数配列

l_ope

フィルタの複合条件

RFIDLogicalOpe.AND and

RFIDLogicalOpe.OR or

【戻り値】

なし

【例外】

NullPointerException

filter が null の場合

RFIDException

RFIDException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_COMMAND	無効オプション指定時
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

本メソッドで設定された設定値は、次回の RFID 読み取り動作に反映されます。

Ull bank には PC,CRC 領域も含まれています。そのため EPC に対してフィルタをおこなう場合は、bitOffset に 0x20 を加算してください。

【例】

RFIDScanner#clearFilter

【機能】

Filter の設定をクリアします

【書式】

```
void clearFilter ();
```

【引数】

なし

【戻り値】

なし

【例外】

RFIDException

RFIDException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

【例】

RFIDScanner#clearDoubleReadingBuffer

【機能】

2 度読みバッファをクリアします

【書式】

```
void clearDoubleReadingBuffer ();
```

【引数】

なし

【戻り値】

なし

【例外】

RFIDException

RFIDException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【例】

RFIDScanner#setDataDelegate

【機能】

RFID 読み取り完了時に読取データを受け取るための delegate を登録します。

登録した delegate を削除する場合は setDataDelegate(null)を実行してください。

【書式】

```
void setDataDelegate (RFIDDataDelegate delegate);
```

【引数】

delegate

読取データ取得の delegate

【戻り値】

なし

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

サンプルコード 2 を参照してください

RFIDDataDelegate#onRFIDDataReceived

【機能】

RFIDScanner#openInventory または RFIDScanner#openRead にて待ち受けを開始し、スキャナからデータが受信されたときに通知するイベントハンドラ。

本イベントハンドラは、RFIDDataDelegate をオーバーライドし、setDataDelegate で登録したインスタンスに通知されます。

【書式】

```
void onRFIDDataReceived (  
    CommScanner* scanner,  
    RFIDDataReceivedEvent* rfidEvent  
);
```

【引数】

scanner

接続された、スキャナクラスのインスタンス

rfidEvent

イベントクラスのインスタンス

【戻り値】

【例】

サンプルコード 2 を参照してください

RFIDDataReceivedEvent#getRFIDData

【機能】

読み取った RFID 情報を ArrayList<RFIDData>形式で返します。

取得できる RFID 情報は、PC,RSSI,UII,アンテナ情報,偏波情報、読み出しデータ、エラーコードです。

【書式】

```
List<RFIDData> getRFIDData ();
```

【引数】

なし

【戻り値】

List<RFIDData>

※RFID 情報が格納されている RFIDData クラスの List

【例】

サンプルコード 2 を参照してください

RFIDData#getData

【機能】

読み取ったデータを読み出します(バイナリ値)。

openRead で指定したデータが格納されます。

【書式】

```
byte[] getData ();
```

【引数】

なし

【戻り値】

読み出しデータ

【例外】

本 API では例外をスローしません(エラー発生しません)

【例】

サンプルコード 2 を参照してください

RFIDData#getRSSI

【機能】

タグ読み取り時の RSSI 値を取得します

【書式】

```
int getRSSI ();
```

【引数】

なし

【戻り値】

タグ読み取り時の RSSI 値

※RSSI は、実データに 10 を掛けた値を返します。

例 -60.5dBm の場合、-605 を返します。

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

サンプルコード 2 を参照してください

RFIDData#getPC

【機能】

タグ読み取り時の PC 値を取得します。

【書式】

```
int getPC ();
```

【引数】

なし

【戻り値】

読み出しデータ

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

サンプルコード 2 を参照してください

RFIDData#getAntenna

【機能】

タグ読み取り時のアンテナ値を取得します。SP1 では未サポート

【書式】

```
byte getAntenna ();
```

【引数】

なし

【戻り値】

読み出しデータ

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

サンプルコード 2 を参照してください

RFIDData#getPolarization

【機能】

タグ読み取り時の偏波値を取得します。

【書式】

```
byte getPolarization ();
```

【引数】

なし

【戻り値】

読み出しデータ

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

サンプルコード 2 を参照してください

RFIDData#getUII

【機能】

読み取ったタグの UII を取得します。

【書式】

```
byte[] getUII ();
```

【引数】

なし

【戻り値】

読み出し UII データ

※XPC を実装しているタグの場合、本 API を実行すると先頭2バイト分に XPC の値が付加されたデータが取得できます。

XPC を実装したタグ読み取り時に本 API で下記値が取得できたとき
先頭2バイトの 0080 が XPC となり、以降の 123456789ABCDEF012345678 が UII の値となります。

0080123456789ABCDEF012345678

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

サンプルコード 2 を参照してください

RFIDData#getIndex

【機能】

読み取ったタグの Index 値を取得します。0～79999 までの値となります。

SP1 は読取毎に、この値を加算し、RFID データと共に送信します。

切断→再接続時に、どこまでデータを受信できたかを SP1 に通知するためなどに使用します。

【書式】

```
int getIndex ();
```

【引数】

なし

【戻り値】

読み出し Index データ

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

RFIDData#getResult

【機能】

タグの読み取りが正常におこなわれたかどうかの結果を取得します。

openRead 時に使用します。

【書式】

```
RFIDResult getResult ();
```

【引数】

なし

【戻り値】

RFIDResult.OK

正常終了

RFIDResult.ERROR_MEM_RANGE

メモリ範囲外アクセスエラー

RFIDResult.ERROR_LOCK_MEM_ACCESS

ロックメモリへのアクセスエラー

RFIDResult.ERROR_LACK_OF_POWER

メモリライト電力不足

RFIDResult.ERROR_OTHER

その他エラー

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

RFIDData#getCh

【機能】

タグ読み取り時の周波数を取得します。

RFIDScanner#setResponse での設定が必要です。

未設定の場合、不定値となります。

【書式】

```
byte getCh();
```

【引数】

なし

【戻り値】

タグ読取時 INDEX 値

INDEX 値と CH の関係は下記参照してください

日本モデル

INDEX	CH
0	5
1	11
2	17
3	23
4	24
5	25

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

RFIDData#getPhase

【機能】

タグ読取時の位相差値を取得します。

RFIDScanner#setResponse での設定が必要です。

未設定の場合、不定値となります。

【書式】

```
byte getPhase();
```

【引数】

なし

【戻り値】

タグ読取時の位相差値

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

RFIDScanner#getCount

【機能】

バッチモード中に読取ったタグ枚数を取得します。

引数 Index からバッチモードで SP1 に蓄えられたタグ枚数を取得できます

【書式】

```
int getCount (int index);
```

【引数】

index

この値からの SP1 に蓄積されたタグ枚数を取得

通常、切断前に取得できたデータの Index+1 の値を指定

【戻り値】

引数 index からバッチモードで SP1 に蓄えられたタグ枚数

【例外】

RFIDException

RFIDException #getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【例】

RFIDScanner#pullData

【機能】

バッチモード中に読取ったタグデータを取得します。読取り動作はしません。

データは RFIDScanner#setDataDelegate で登録した delegate に通知されます。

RFIDScanner#get Count であらかじめ取得した枚数分のデータ取得後に RFIDScanner#close を実行してください。

【書式】

```
void pullData (int index);
```

【引数】

index

この値からの SP1 に蓄積されたタグデータを取得

【戻り値】

なし

【例外】

RFIDException

RFIDException #getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode. INVALID_COMMAND	無効オプション指定時
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【例】

RFIDScanner#setResponse

【機能】

RFID 読取時の取得情報項目を設定します

【書式】

```
void setResponse (  
    RFIDScannerResponse response  
);
```

【引数】

response

読取時の取得情報の項目が設定されている RFIDScannerResponse クラス変数

【戻り値】

なし

【例外】

NullPointerException
response が null の場合

RFIDException
RFIDException#getErrorCode()でエラーを取得できます。
発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_COMMAND	無効オプション指定時
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

RFID 読み取り動作中には本メソッドを使用しないでください。本メソッドで設定された設定値は、次回 RFID 読み取り動作に反映されます。

【例】

RFIDScanner#getResponse

【機能】

RFID 読取時の取得情報項目を取得します。

【書式】

```
RFIDScannerResponse getResponse ();
```

【引数】

なし

【戻り値】

RFIDScannerResponse 現在の取得情報項目

【例外】

RFIDException

RFIDException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

【例】

RFIDException#getErrorCode

【機能】

RFIDException のエラーコードを返します。

RFIDException が throw されたとき、エラーコードを確認して原因を調べることができます。

【書式】

```
ErrorCode getErrorCode ();
```

【引数】

なし

【戻り値】

ErrorCode. COMMUNICATION_ERROR

スキャナとの通信エラー

ErrorCode. COMMUNICATION_TIMEOUT

スキャナからの応答タイムアウト

ErrorCode. TOO_MUCH_COMMAND_QUEUE

コマンドキューがあふれたとき

ErrorCode. INVALID_COMMAND

無効オプション指定時

ErrorCode. INVALID_STATUS

状態が不正なとき。open していないのに close 呼び出しなど。

ErrorCode. INVALID_RESPONSE

不正な応答がスキャナからあったとき

ErrorCode. NOT_SUPPORT_COMMAND

スキャナが対応していないコマンド送信時

ErrorCode. UNKNOWN

原因不明です。

ErrorCode.ACCESS_OUT_OF_RANGE_MEMORY

メモリ範囲外へのアクセス

ErrorCode.SCANNER_ACCESS_LOCK_MEMORY

ロックメモリへのアクセス

ErrorCode.SCANNER_POWER_SHORTAGE_OF_WRITE_MEMORY

メモリライト電力不足

ErrorCode.SCANNER_WRITE_LOCK_KILL_TIMEOUT

時間内に Write or Lock or Kill が完了しなかったとき

ErrorCode.SCANNER_WRITE_LOCK_KILL_UNKNOWN

Write or Lock or Kill エラー（理由不明）

ErrorCode.SCANNER_CHARGING

充電中エラー(充電中は RFID 読み取り不可)

ErrorCode.SCANNER_TRIGGER_MODE

トリガモードエラー(未対応のトリガモードで WRITE or LOCK or KILL 実行)

ErrorCode. SCANNER_TEMP_ERROR

温度エラー発生時

ErrorCode.EXECUTE

実行エラー(ROM 保存失敗など)

ErrorCode. SCANNER_EPD_LOST

対象タグ消失時

ErrorCode. SCANNER_EPD_CHARGE_TIMEOUT

タグ給電タイムアウト

ErrorCode. SCANNER_EPD_OPERATION_TIMEOUT

EPD 処理タイムアウト

【例】

サンプルコード 2

```
public class ExecuteRFIDActivity extends AppCompatActivity implements RFIDDataDelegate {

    private CommScanner mScanner = null;
    private RFIDScanner mRfidScanner = null;

    private TextView textGetVersion = null;
    int count = 0;
    String data;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_execute_rfid);

        textGetVersion = (TextView) findViewById(R.id.get_version);

        // get paired list
        List<CommScanner> scanner_list = CommManager.getScanners();

        for (int i = 0; i < scanner_list.size(); ++i) {
            Log.d("ExecuteRFIDActivity", "BT Address=" + scanner_list.get(i).getBTAddress());
            Log.d("ExecuteRFIDActivity", "BT LC=" + scanner_list.get(i).getBTLocalName());

            // select Bluetooth local name
            if (scanner_list.get(i).getBTLocalName().equals("SP1-800001")) {
                mScanner = scanner_list.get(i);
            }
        }

        // connect SP1-800001
        mScanner.claim();

        textGetVersion.setText(mScanner.getVersion());
    }

    public void onBtnClick(View v) {
        switch (v.getId()) {
            case R.id.demo_start_button:
                if (mScanner != null) {

                    CommScannerParams commScannerParams = new CommScannerParams();
                    commScannerParams.buzzerVolume = CommScannerParams.BuzzerVolume.LOW;

                    if (mRfidScanner == null) {
                        mRfidScanner = mScanner.getRFIDScanner();
                    }
                    mRfidScanner.setDataDelegate(this);
                }

                if (mRfidScanner != null) {

                    // get current settings from SP1
                    RFIDScannerSettings rfidScannerSettings = mRfidScanner.getSettings();

                    rfidScannerSettings.scan.triggerMode = RFIDScannerSettings.Scan.TriggerMode.AUTO_OFF;

                    mRfidScanner.setSettings(rfidScannerSettings);

                    // start Inventory
                    mRfidScanner.openInventory();
                }
                break;
        }
    }
}
```

```

        case R.id.demo_stop_button:
            if (mRfidScanner != null) {

                mRfidScanner.setDataDelegate(null);

                mRfidScanner.close();
            }
            break;
    }
}

@Override
public void onRFIDDataReceived(CommScanner commScanner, RFIDDataReceivedEvent rfidDataReceivedEvent)
{
    Log.d("ExecuteRFIDActivity", "OnRFIDDataReceived");
    count++;
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append("count" + count + "\n");

    List<RFIDData> rfidDataList = rfidDataReceivedEvent.getRFIDData();
    for (RFIDData rfidData : rfidDataList) {
        stringBuilder.append("data:" + byteToString(rfidData.getData()) + "\n");
        stringBuilder.append("PC:" + rfidData.getPC() + "\n");
        stringBuilder.append("Polarization:" + rfidData.getPolarization() + "\n");
        stringBuilder.append("RSSI:" + rfidData.getRSSI() + "\n");
        stringBuilder.append("UII:" + byteToString(rfidData.getUII()) + "\n");
    }
    data = stringBuilder.toString();
    Log.d("ExecuteRFIDActivity", "data=" + data);
}

private String byteToString(byte[] bytes){
    StringBuilder stringBuilder = new StringBuilder();
    for(byte b : bytes){
        stringBuilder.append(String.format("%02X", b));
    }
    return stringBuilder.toString();
}

private byte[] stringToByte(String hex) {
    byte[] bytes = new byte[hex.length() / 2];
    for (int index = 0; index < bytes.length; index++) {
        bytes[index] = (byte) Integer.parseInt(hex.substring(index * 2, (index + 1) * 2), 16);
    }
    return bytes;
}
}

```

RFIDScannerFilter クラス

【機能】

Filter 制御の指定をおこないます。

プロパティ : 説明	設定値		初期値	SP1
RFIDScannerFilter				
bank :バンク指定	Bank.UII	UII bank		✓
	Bank.TID	TID bank		✓
	Bank.USER	user bank		✓
bitOffset :ビットオフセット	"00000000" - "0007FFFF" (HEX)	ビットオフセット		✓
bitLength :有効ビット長	"01" - "FF" (HEX)	フィルタデータ最下位ビットからの有効ビット長		✓
filterData :フィルタデータ	最大 255bit byte[]	フィルタデータ (最下位ビットから有効ビット数分を使用。最大 255bit)		✓

RFIDScannerSettings クラス

【機能】

RFID 読み取り関連の設定値を格納します。

(✓:サポート)

プロパティ : 説明	設定値		初期値	SP1
RFIDScannerSettings.scan : 読み取り動作関連				
triggerMode : トリガモード	TriggerMode.AUTO_OFF	オートオフモード		✓
	TriggerMode.MOMENTARY	モメンタリモード		✓
	TriggerMode.ALTERNATE	オルタネートモード		✓
	TriggerMode.CONTINUOUS1	連続読み取りモード 1	○	✓
	TriggerMode.CONTINUOUS2	連続読み取りモード 2		✓
oneShot : ワンショット秒数	0-5	0 のときはオートオフモード 1-5 ワンショット 1~5 秒(byte)	0	✓
powerLevelRead : Read 時の出力	4-30	指定したい dBm 値	30	✓
powerLevelWrite : Write 時の出力	4-30	指定したい dBm 値	30	✓
channel : 周波数設定	0x00000000 – 0x0007FFFF	日本向け 0x00000001 : Ch5, 0x00000002 : Ch11, 0x00000004 : Ch17, 0x00000008 : Ch23, 0x00000010 : Ch24, 0x00000020 : Ch25	0x0000003F (Ch5,11,17, 23 ~Ch25 許可)	✓
		欧州向け 0x00000001 : Ch4, 0x00000002 : Ch7, 0x00000004 : Ch10, 0x00000008 : Ch13	0x0000000F (Ch4,7,10, 13 許可)	
		イスラエル向け 0x00000001 : Ch1, 0x00000002 : Ch2, 0x00000004 : Ch3, 0x00000008 : Ch4, 0x00000010 : Ch5	0x0000001F (Ch1,2,3,4,5 許可)	
		インド向け 0x00000001 : Ch1, 0x00000002 : Ch2, 0x00000004 : Ch3	0x00000007 (Ch1,2,3 許可)	
		その他の仕向けは設定値が無視される。	0x0007FFFF	
qParam : Q 値	0-7	RF タグからの応答が衝突する確率を調整	4	✓
sessionFlag : セッションフラグ	SessionFlag.S0	RF タグ通信待機状態後、再度同じ RF タグ の UII 取得可能となる。	○	✓
	SessionFlag.S1	RF タグの UII 取得後、規定時間以上経過す れば再度同じ RF タグの UII 取得可能となる。		✓
	SessionFlag.S2	RF タグ通信待機状態後、規定時間以上経過 すれば再度同じ RF タグの UII 取得可能とな る。		✓
	SessionFlag.S3	S2 と同じ		✓
sessionInit : セッション初期化	true	有効		✓
	false	無効	○	✓
polarization : 偏波設定	Polarization.V	垂直のみ		✓
	Polarization.H	水平のみ		✓
	Polarization.Both	垂直、水平両方	○	✓
powerSave : 省電力設定	true	省電力モード1		✓
	false	省電力なし	○	✓

doubleReading : 二度読み防止	DoubleReading.Free	二度読み防止なし	○	✓
	DoubleReading.PREVENT1	読み取り動作中二度読み防止実施		✓
	DoubleReading.PREVENT2	電源オン中二度読み防止実施		✓
writeVeri : Write 時 Verify	false	ベリファイなし	○	✓
	true	ベリファイあり		✓
linkProfile : Link Profile 番号	1 or 2 or 4 or 5	Link Profile番号(byte) 欧州向けを除く	4	✓
		欧州向け	2	
powerSaveExt : 拡張省電力設定	PowerSaveExt.DISABLE	省電力なし	○	✓
	PowerSaveExt.MODE1	省電力モード1		✓
	PowerSaveExt.MODE2	省電力モード2		✓

【備考】

各トリガモードについて

オートオフ(ノーマル)

トリガ押下してから最大で 5 秒間 RF タグ通信状態になります。

通信状態で 5 秒経過する前に RF タグ制御が完了することで待機状態になります。

また、トリガを離すことでも待機状態となります。

オートオフ(ワンショット)

トリガ押下してからワンショット時間だけ RF タグ通信状態になります。

ノーマルと異なりトリガを離してもワンショット時間が経過するまで通信状態を維持します。

通信状態でワンショット時間経過する前に RF タグ制御が完了することで待機状態になります。

モメンタリ

トリガを押下している間、RF タグ通信状態になります。

本モードは Write,Lock,Kill が使用できません。

オルタネート

トリガを押下する毎に RF タグ通信状態と待機状態を交互に切り替えます。

本モードは Write,Lock,Kill が使用できません。

連続モード 1

タグ機能制御開始 API(openInventory など)の実行から、close API 実行までの間、

RF タグ通信状態になります。

本モードは Write,Lock,Kill が使用できません。

連続モード 2

タグ機能制御開始 API の実行から、RF タグ制御が完了するまでの間、RF タグ通信状態になります。

RF タグ制御が完了すると自動的に待機状態となり、再度通信状態にするには close API 実行後、タグ機能制御開始 API を実行してください。

二度読み防止モードについて

設定内容	二度読み防止動作
二度読み防止なし	同一の RF タグを何度でも検出します。
読み取り動作中 二度読み防止実施	RF タグ読み取り動作中の間、二度読み防止を継続します。 以下の条件を満たした場合に、二度読み防止バッファがクリアされます。 ・トリガモードによる読み取り停止 ・RF タグ制御停止コマンドによる読み取り停止 ・バーコード読み取りモードへの遷移 ・充電開始 ・電源 OFF ・RFIDScanner#clearDoubleReadingBuffer 実行時
電源オン中 二度読み防止実施	電源オンしている間、二度読み防止を継続します。 以下の条件を満たした場合に、二度読み防止バッファがクリアされます。 ・電源 OFF ・RFIDScanner#clearDoubleReadingBuffer 実行時 ・RFIDScanner#openInventory と RFIDScanner#openRead 間の切替時 ・RFIDScanner#openRead の size を 128 以下と 129 以上で切替時

省電力設定について

拡張省電力設定を使用するようにしてください。こちらの省電力設定は互換のためにあります。
将来サポート外になる可能性があります。

拡張省電力設定について

省電力効果は、

通常モード(省電力禁止) < 省電力モード 1 < 省電力モード 2 となります。

省電力効果が高いほど、読取速度は低下します。

省電力設定と拡張省電力設定の関係について

	PowerSaveExt.DISABLE	PowerSaveExt.MODE1	PowerSaveExt.MODE2
PowerSave.false	通常モード(省電力禁止)	省電力モード 1	省電力モード 2
PowerSave.true	省電力モード 1	省電力モード 1	省電力モード 2

Read/Write 時の出力設定について

仕様値は 10dBm-30dBm です。

4dBm-9dBm は使用タグとの動作確認を実施して、実用的かどうかを判断してください。

RFIDScannerResponse クラス

【機能】

RFID 読取時の取得情報項目を格納します

(✓:サポート)

プロパティ : 説明	設定値		初期値	SP1
RFIDScannerResponse:				
format : フォーマット	Format.BINARY	バイナリモード	○	✓
	Format.TEXT	テキストモード		
pc :PC 情報付加	true	許可	○	✓
	false	禁止		
rssi :RSSI 情報付加	true	許可	○	✓
	false	禁止		
antenna :アンテナ情報付加	true	許可	○	✓
	false	禁止		
polarization :偏波情報付加	true	許可	○	✓
	false	禁止		
ch :周波数情報付加	true	許可		✓
	false	禁止	○	✓
phase :位相情報付加	true	許可		✓
	false	禁止	○	✓
autolinkprofile : AutoLinkProfile 情報付加	true	許可		
	false	禁止	○	

【備考】

RFIDData#getCh と RFIDData#getPhase で情報を取得するためには、ch と phase を許可にして RFIDScanner#setResponse を実行してください。
 デフォルト以外の設定を使用する場合は、接続確立時に RFIDScanner#setResponse を毎回実行してください。
 autolinkprofile は互換のためにありますが使用できません。

6. バーコード

6.1. 概要

バーコード読み取りを実行や、読み取り時の設定などを行う API です。

6.1.1. 読み取り許可

バーコードデバイスは、BarcodeScanner#claim()メソッドを使うことにより読み取りが許可されます。指定できるコードタイプには次のものがあり、BarcodeScanner#setSettings()メソッドを使うことにより 1 つ以上のバーコードタイプを組み合わせて指定することができます。下記バーコードタイプのデフォルト値、オプションについては BarcodeScannerSettings class の説明を参照してください。

コード種	SP1 サポート
1 次元コード	
EAN-13 UPC-A	✓
EAN-8	✓
UPC-E	✓
ITF	✓
STF	✓
Codabar	✓
Code39	✓
Code93	✓
Code128, GS1-128 (EAN-128)	✓
MSI	
GS1 Databar	✓
GS1 Databar Limited	✓
GS1 Databar Expanded	✓
2 次元コード	
QR Model1, 2	✓
Micro QR	✓
iQR	
Data Matrix	✓
PDF417	✓
Micro PDF417	✓
Maxi Code	✓
GS1 Composite	✓
多段コード	✓

6.1.2. バーコードデータ

読み取ったバーコードデータは BarcodeData の List 配列に格納されます。

バーコードデータの最大値は 8192 バイトとなります。

BarcodeDataクラスに格納されているバーコードデータの情報を読み出すには、以下のメソッドを使用します

- ・BarcodeData#getSymbologyAim:コード種(AIM準拠)取得メソッド
- ・BarcodeData#getSymbologyDenso:コード種(DENSO規定)取得メソッド
- ・BarcodeData#getData:エンコード済バーコードデータ取得メソッド

※バーコードデータ長を取得したい場合は、下記のようにlengthメソッドを使用して下さい。

例: int barcodeDataLength = data.length();

6.1.3. SQRC

SQRC は、データの読み取り制限機能を持った QR コードです。

詳細は SQRC Setting ユーザーズマニュアルをご覧ください。

6.2. バーコード関連 API

BarcodeScanner#openReader

【機能】

BarcodeScanner#setSettings()で設定された内容に従って、バーコードスキャナの読み取りを許可します。
設定可能項目は BarcodeScannerSettings クラスを参照ください。
RFID との同時オープンはできません。

【書式】

```
void openReader ();
```

【引数】

なし

【戻り値】

なし

【例外】

BarcodeException
BarcodeException#getErrorCode()でエラーを取得できます。
発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【例】

サンプルコード 3 を参照してください

BarcodeScanner#closeReader

【機能】

バーコードデバイスをクローズし、読み取りを禁止します。

【書式】

```
void closeReader ();
```

【引数】

なし

【戻り値】

なし

【例外】

BarcodeException

BarcodeException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【例】

サンプルコード 3 を参照してください

BarcodeScanner#setSettings

【機能】

読取関連の設定を設定します。

【書式】

```
void setSettings (  
    BarcodeScannerSettings settings  
);
```

【引数】

settings

読取関連の設定が設定されている BarcodeScannerSettings クラス変数

【戻り値】

なし

【例外】

NullPointerException
settings が null の場合

BarcodeException
BarcodeException#getErrorCode()でエラーを取得できます。
発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode. INVALID_COMMAND	無効オプション指定時
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【備考】

本メソッドで設定された設定値は、次回バーコード読み取り動作に反映されます。

【例】

サンプルコード 3 を参照してください

BarcodeScanner#getSettings

【機能】

現在の読取関連の設定を取得します。

【書式】

```
BarcodeScannerSettings getSettings ();
```

【引数】

なし

【戻り値】

BarcodeScannerSettings 現在の読取関連の設定

【例外】

BarcodeException

BarcodeException#getErrorCode()でエラーを取得できます。

発生するエラーは以下の通りです。

エラーラベル	内容
ErrorCode. COMMUNICATION_TIMEOUT	スキャナからの応答タイムアウト
ErrorCode. INVALID_STATUS	不正状態エラー
ErrorCode. NOT_SUPPORT_COMMAND	未サポートコマンド

【例】

サンプルコード 3 を参照してください

BarcodeScanner#setDataDelegate

【機能】

バーコード読み取り完了時に読取データを受け取るための delegate を登録します。

登録した delegate を削除する場合は setDataDelegate(null)を実行してください。

【書式】

```
void setDataDelegate (BarcodeDataDelegate delegate);
```

【引数】

delegate

読取データ取得の delegate

【戻り値】

なし

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

サンプルコード 3 を参照してください

BarcodeDataDelegate#onBarcodeDataReceived

【機能】

BarcodeScanner#openReader にて待ち受けを開始し、スキャナからデータが受信されたときに通知するイベントハンドラ。

本イベントハンドラは、BarcodeDataDelegate をオーバーライドし、setDataDelegate で登録したインスタンスに通知されます。

【書式】

```
void onBarcodeDataReceived (  
    CommScanner* scanner,  
    BarcodeDataReceivedEvent* barcodeEvent  
);
```

【引数】

scanner

接続された、スキャナクラスのインスタンス

barcodeEvent

イベントクラスのインスタンス

【戻り値】

【例】

サンプルコード 3 を参照してください

BarcodeDataReceivedEvent#getBarcodeData

【機能】

読み取ったバーコード情報を ArrayList<BarcodeData>形式で返します。
取得できるバーコード情報は、読み出しデータ、コード種(DENSO 規定、AIM)、読み取り桁数です。
※多段のような複数のバーコード情報が取得出来る場合でも取得可能です。
※バーコード情報についての詳細は BarcodeData クラスの各メソッド仕様をご参照ください

【書式】

```
List<BarcodeData> getBarcodeData ();
```

【引数】

なし

【戻り値】

List<BarcodeData>

※バーコード情報が格納されている BarcodeData クラスの List

【例】

サンプルコード 3 を参照してください

BarcodeData#getData

【機能】

読み取ったデータを取得します。

getData で取得できるデータはバイナリデータです。

バーコード内のデータに応じて、適宜エンコードを行ってください。

【書式】

```
byte[]  getData ();
```

【引数】

なし

【戻り値】

読み出しデータが格納されているバイト配列

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

サンプルコード 3 を参照してください

BarcodeData#getSymbologyDenso

【機能】

読み取ったコード種を獲得します。獲得されるコード種の形式は DENSO 規定です。

【書式】

```
String getSymbologyDenso ();
```

【引数】

なし

【戻り値】

コード種(DENSO 規定)

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

サンプルコード 3 を参照してください

BarcodeData#getSymbologyAim

【機能】

読み取ったコード種を獲得します。獲得されるコード種の形式は、AIM USA “Guidelines on Symbology Identifiers.”に準拠したコード種です。

【書式】

```
String getSymbologyAim ();
```

【引数】

なし

【戻り値】

コード種(AIM USA “Guidelines on Symbology Identifiers.”に準拠)

【例外】

本 API では例外をスローしません(エラー発生しません)。

【例】

サンプルコード 3 を参照してください

【備考】

getSymbologyDenso()と getSymbologyAim()で取得できる値とコード種は、以下のとおりです。

コード種			DENSO 規定	AIM USA (*1)
なし(未読)			0	""(NULL)
QR			Q	]Qm
	未編集の連結		S(*2)	S(*2)
マイクロ QR			Q(*2)	Q(*2)
SQRC			Q(*2)	Q(*2)
iQR			G	]Qm
	未編集の連結		S(*2)	S(*2)
PDF417			Y	]L0
MaxiCode			X	]Um
Data Matrix			Z	]dm
EAN-13(JAN-13) COMPOSITE	リニア コード	アドオン無し	A	]E0
		2 桁アドオン付き		]E3
		5 桁アドオン付き		
	2D コード		Y	]e0
UPC-A COMPOSITE	リニア コード	アドオン無し	A	]E0
		2 桁アドオン付き		]X3
		5 桁アドオン付き		
	2D コード		Y	]e0
EAN-8(JAN-8) COMPOSITE	リニア コード	アドオン無し	B	]E4
		2 桁アドオン付き		]E5
		5 桁アドオン付き		]E6
	2D コード		Y	]e0
UPC-E COMPOSITE	リニア コード	アドオン無し	C	]X0
		2 桁アドオン付き		]X3
		5 桁アドオン付き		
	2D コード		Y	]e0
GS1-128 (EAN-128) COMPOSITE	リニアコードタイプ		W	]e0
	2D コードタイプ		Y	無
GS1 DataBar COMPOSITE	リニアコードタイプ		R	]e0
	2D コードタイプ		Y	無
EAN-13 (JAN-13)	アドオン無し		A	]E0
	2 桁アドオン付き			]E3
	5 桁アドオン付き			
UPC-A	アドオン無し		A	]X0
	2 桁アドオン付き			]X3
	5 桁アドオン付き			
EAN-8 (JAN-8)	アドオン無し		B	]E4
	2 桁アドオン付き			]E5
	5 桁アドオン付き			]E6
UPC-E	アドオン無し		C	]X0
	2 桁アドオン付き			]X3
	5 桁アドオン付き			
ITF			I	]Im
STF	short		H	]R0
	normal			]S0
CODABAR (NW-7)			N	]Fm
CODE-39			M	]Am
CODE-93			L	]G0
CODE-128			K	]Cm
GS1-128 (EAN-128)			W	]C1
MSI			P	]M1
GS1 DataBar			R	]e0

(*1) 末尾「m」は下表に示すようにバーコード体系のデータフォーマットにより異なります。

例)]I1] : Flag Character (ASCII 93)

I : Code Character (ITF)

1 : Modifier Character (下表)

例えば、ITF で C/D 有りの読み取りに設定している場合、コードマークは “]I1” となります。

コードタイプ	Modifier Character	説明
ITF	0	チェックデジット無しの読み取り
	1	チェックデジット有りの読み取り
CODE-39	0	チェックデジット無しの読み取り
	1	チェックデジット有りの読み取り
CODABAR (NW-7)	0	チェックデジット無しの読み取り
	1	チェックデジット有りの読み取り
CODE-128	0	スタートコードから 1 番目と 2 番目のキャラクタに FNC1 を含まない
	2	スタートコードから 2 番目のキャラクタが FNC1
QR コード	0	モデル 1
	1	モデル 2
	2	モデル 2 ECI を含む(*3)
	3	モデル 2 スタートコードから 1 番目のキャラクタが FNC1
	4	モデル 2 ECI を含む スタートコードから 1 番目のキャラクタが FNC1(*3)
	5	モデル 2 スタートコードから 1 番目のキャラクタが FNC2(*3)
	6	モデル 2 ECI を含む スタートコードから 1 番目のキャラクタが FNC2(*3)
iQR コード	A	スタートコードから 1 番目のキャラクタに FNC1 を含まない
	C	スタートコードから 1 番目のキャラクタが FNC1
MaxiCode	0	mode4、mode5
	1	mode2、mode3
Data Matrix	1	ECC-200
	2	ECC-200 (スタートコードから 1 または 5 番目のキャラクタが FNC1)
	3	ECC-200 (スタートコードから 2 または 6 番目のキャラクタが FNC1)
	4	ECC-200 ECI を含む(*3)
	5	ECC-200 ECI を含む (スタートコードから 1 または 5 番目のキャラクタが FNC1) (*3)
	6	ECC-200 ECI を含む (スタートコードから 2 または 6 番目のキャラクタが FNC1) (*3)

(*2) AIM USA「Guidelines on Symbology Identifiers」に準拠していないコードタイプについては DENSO 規定のコード種値と等しくなります

(*3) 対応予定

BarcodeData#getPrivateDataLength

【機能】

SQRC コードの非公開部のコード桁数を獲得します。

【書式】

```
int getPrivateDataLength ();
```

【引数】

なし

【戻り値】

SQRC コードの非公開部のコード桁数

【備考】

【例】

BarcodeManager#create()のサンプルコードを参照してください。

BarcodeException#getErrorCode

【機能】

BarcodeException のエラーコードを返します。

BarcodeException が throw されたとき、エラーコードを確認して原因を調べることができます。

【書式】

```
ErrorCode getErrorCode ();
```

【引数】

なし

【戻り値】

ErrorCode. COMMUNICATION_ERROR

スキャナとの通信エラー

ErrorCode. COMMUNICATION_TIMEOUT

スキャナからの応答タイムアウト

ErrorCode. TOO_MUCH_COMMAND_QUEUE

コマンドキューがあふれたとき

ErrorCode. INVALID_COMMAND

無効オプション指定時

ErrorCode. INVALID_STATUS

状態が不正なとき。open していないのに close 呼び出しなど。

ErrorCode. INVALID_OPERATION

無効な操作したとき

ErrorCode. INVALID_RESPONSE

不正な応答がスキャナからあったとき

ErrorCode. NOT_SUPPORT_COMMAND

スキャナが対応していないコマンド送信時

ErrorCode. UNKNOWN

原因不明です。

【例】

BarcodeScanner#addStatusListener()のサンプルコードを参照してください。

サンプルコード 3

```
public class ExecuteBarcodeActivity extends AppCompatActivity implements BarcodeDataDelegate{

    private CommScanner mScanner = null;
    private BarcodeScanner mBarcodeScanner;

    int count = 0;
    String data;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_execute_barcode);

        // get paired list
        List<CommScanner> scanner_list = CommManager.getScanners();

        for (int i = 0; i < scanner_list.size(); ++i) {
            // select Bluetooth local name
            if (scanner_list.get(i).getBTLocalName().equals("SP1-800001")) {
                mScanner = scanner_list.get(i);
            }
        }

        // connect SP1-800001
        mScanner.claim();
    }

    public void onBtnClick(View v) {
        switch (v.getId()) {
            case R.id.demo_start_button:
                if(mScanner != null){

                    mBarcodeScanner = mScanner.getBarcodeScanner();

                    mBarcodeScanner.setDataDelegate(this);

                    // get scanner settings
                    BarcodeScannerSettings settings = mBarcodeScanner.getSettings();

                    // set scanner settings
                    mBarcodeScanner.setSettings(settings);
                }
                if(mBarcodeScanner != null){

                    // open
                    mBarcodeScanner.openReader();
                }
                break;
            case R.id.demo_stop_button:
                if(mBarcodeScanner != null){

                    mBarcodeScanner.setDataDelegate(null);

                    // close
                    mBarcodeScanner.closeReader();
                }
                break;
        }
    }
}
```

```

public void onBarcodeDataReceived(CommScanner commScanner, BarcodeDataReceivedEvent
barcodeDataReceivedEvent) {
    Log.d("ExecuteBarcodeActivity", "OnBarcodeDataReceived");
    count++;
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append("count" + count + "\n");

    List<BarcodeData> barcodeDataList = barcodeDataReceivedEvent.getBarcodeData();
    for(BarcodeData barcodeData : barcodeDataList) {
        stringBuilder.append("Data:" + new String(barcodeData.getData()) + "\n");

        stringBuilder.append("SymbologyAim:" + barcodeData.getSymbologyAim() + "\n");

        stringBuilder.append("SymbologyDenso:" + barcodeData.getSymbologyDenso() + "\n");
    }
    data = stringBuilder.toString();
    Log.d("ExecuteBarcodeActivity", "data=" + data);
}

}

```

BarcodeScannerSettings クラス

【機能】

バーコード読み取り関連の設定値を格納します。

(✓:サポート)

プロパティ : 説明	設定値		初期値	SP1
BarcodeScannerSettings.scan : 読み取り動作関連				
triggerMode : トリガモード	TriggerMode.AUTO_OFF	オートオフモード	○	✓
	TriggerMode.MOMENTARY	モメンタリモード		✓
	TriggerMode.ALTERNATE	オルタネートモード		✓
	TriggerMode.CONTINUOUS	連続読み取りモード		✓
	TriggerMode.TRIGGER_RELEASE	トリガリリースモード		✓
lightMode : 照明モード	LightMode.AUTO	自動	○	✓
	LightMode.ALWAYS_ON	常時 ON		✓
	LightMode.OFF	消灯		✓
markerMode : マーカモード	MarkerMode.NORMAL	ノーマル	○	✓
	MarkerMode.AHEAD	マーカ先行		✓
	MarkerMode.OFF	マーカなし		✓
sideLightMode : 補助照明	SideLightMode.ON	ON		/
	SideLightMode.OFF	OFF	○	
BarcodeScannerSettings.decode : デコード関連				
sameBarcodeIntervalTime : 二度読み防止解除時間	0~255	防止解除時間(x100ms)	10	✓
decodeLevel : デコードレベル	1~9	レベル 1~9	4	✓
invertMode : 白黒反転読取	InvertMode.DISABLED	禁止	○	✓
	InvertMode.INVERSION_ONLY	許可 (反転のみ)		
	InvertMode.AUTO	許可 (自動判別)		✓
pointScanMode : ポイントスキャンモード	PointScanMode.DISABLED	ノーマル	○	✓
	PointScanMode.ENABLED	ポイントスキャン		✓
reverseMode : 表裏反転読取	ReverseMode.DISABLED	禁止	○	✓
	ReverseMode.ENABLED	許可		✓
charset : バーコードの encoding 方式	文字列 ※Android Developer サイト参照	charset 指定文字列	"UTF-8"	
mirrorReflection : 鏡面反射読取	MirrorReflection.DISABLED	禁止		✓
	MirrorReflection.ENABLED	許可	○	✓

【備考】

照明モードについて

バーコード読み取り時の照明をつけるかどうか設定できます。

反射しやすい素材の場合に OFF にすることで読み取りを改善できることがあります。

自動でも数回に 1 度は OFF にして読み取れるようにしています。

マーカモードについて

バーコード読み取り時に目安となる赤いマーカの ON/OFF を設定することができます。

デコードレベルについて

読み取り許容レベルを設定します。

値が小さくなるとコードの読み取り率は向上しますが、品質の悪いコード(割れ、汚れなど)を誤読する危険性が大きくなります。 値が大きくなるとコードの読み取り率は低下しますが、誤読の危険性は小さくなります。

ポイントスキャンモードについて

マーカの中心部のにあるコードを狙って読み取りすることができます。

マーカにコードがない場合や外来光などによりマーカが検出できない場合には読み取りすることができません。また、この読み取りはマーカの点灯が許可されている場合のみ有効です。

プロパティ : 説明	設定値		初期値	SP1
BarcodeScannerSettings.decode.multiLineMode : 多段コード読み取り関連				
enabled	true	許可		✓
: 多段読取	false	禁止	○	✓
BarcodeScannerSettings.decode.multiLineMode.symbology1st/2nd/3rd : 1 段目/2 段目/3 段目の読取コード情報				
symbologyType : コード種	MultiLineSymbologyType.NONE	未指定(読取禁止)	○	✓
	MultiLineSymbologyType.EAN13UPCA	EAN13/UPC-A		✓
	MultiLineSymbologyType.EAN8	EAN-8		✓
	MultiLineSymbologyType.UPCE	UPC-E		✓
	MultiLineSymbologyType.INTERLEAVED2OF5	インターリーブド 2of5 (ITF)		✓
	MultiLineSymbologyType.STANDARD2OF5	スタンダード 2of5 (STF)		✓
	MultiLineSymbologyType.CODABAR	CODABAR (NW-7)		✓
	MultiLineSymbologyType.CODE39	CODE-39		✓
	MultiLineSymbologyType.CODE93	CODE-93		✓
	MultiLineSymbologyType.CODE128	CODE-128		✓
firstCharacter : 1 桁目の文字 (コード種が POS の場合のみ有効)	""	限定しない	○	✓
	"?"	限定しない		✓
	"0" ~ "9"	読取可能なラベルの 1 桁目の文字		✓
secondCharacter : 1 桁目の文字 (コード種が POS の場合のみ有効)	""(空文字列)	限定しない	○	✓
	"?"	限定しない		✓
	"0" ~ "9"	読取可能なラベルの 2 桁目の文字		✓
startStop Character : スタート ストップ キャラクタ(コード 種が Codabar の場合のみ有効)	""(空文字列)	スタートストップを限定 しない	○	✓
	スタートキャラクタとストップキャラクタを連結させた文字列 ex) "AA", "AB" "?"を指定すると、スタート/ストップのいずれか一方のみ指定可能 ex) "A?", "?D"	指定されたスタート/ストップ キャラクタをもつラベルのみ 読取可能		✓
lengthMin : 最小桁数(コード 種が POS 以外の 場合のみ有効)	0	最小桁数指定なし	○	✓
	2~99(ITF,STF の場合) 3~99(Codabar の場合) 1~99(上記以外の場合)	最小桁数		✓
lengthMax : 最大桁数(コード 種が POS 以外の 場合のみ有効)	2~99(ITF,STF の場合) 3~99(Codabar の場合) 1~99(上記以外の場合)	最大桁数	99	✓
verifyCheckDegit : チェックデジット 検証(コード種 が POS, Code39, Code93 以外の場合のみ有効)	true	有効		✓
	false	無効	○	✓

プロパティ : 説明	設定値		初期値	SP1
BarcodeScannerSettings.decode.symbologies.読取許可コード関連				
BarcodeScannerSettings.decode.symbologies.ean13upcA: EAN-13, UPC-A 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
firstCharacter : 1 桁目の文字	""	限定しない	○	✓
	"?"	限定しない		✓
	"0" ~ "9"	読取可能なコードの 1 桁目の文字		✓
secondCharacter : 2 桁目の文字	""	限定しない	○	✓
	"?"	限定しない		✓
	"0" ~ "9"	読取可能なコードの 2 桁目の文字		✓
BarcodeScannerSettings.decode.symbologies.ean13upcA.addOn : EAN-13, UPC-A アドオン関連				
enabled : アドオン付き	true	読取許可		✓
	false	読取禁止	○	✓
addOn2Digit : 2 桁アドオン付き	true	読取許可		/
	false	読取禁止	○	
addOn5Digit : 5 桁アドオン付き	true	読取許可		/
	false	読取禁止	○	
onlyWithAddOn : アドオン付きのみ	true	アドオン付きのみに限定する		/
	false	アドオン付きのみに限定しない	○	
BarcodeScannerSettings.decode.symbologies.ean8 : EAN-8 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
firstCharacter : 1 桁目の文字	""	限定しない	○	✓
	"?"	限定しない		✓
	"0" ~ "9"	読取可能なコードの 1 桁目の文字		✓
secondCharacter : 2 桁目の文字	""	限定しない	○	✓
	"?"	限定しない		✓
	"0" ~ "9"	読取可能なコードの 2 桁目の文字		✓
BarcodeScannerSettings.decode.symbologies.ean8.addOn : EAN-8 アドオン関連				
enabled : アドオン付き	true	読取許可		✓
	false	読取禁止	○	✓
addOn2Digit : 2 桁アドオン付き	true	読取許可		/
	false	読取禁止	○	
addOn5Digit : 5 桁アドオン付き	true	読取許可		/
	false	読取禁止	○	
onlyWithAddOn : アドオン付きのみ	true	アドオン付きのみに限定する		/
	false	アドオン付きのみに限定しない	○	
BarcodeScannerSettings.decode.symbologies.upcE: UPC-E 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
firstCharacter : 1 桁目の文字	""	限定しない	○	✓
	"?"	限定しない		✓
	"0" ~ "9"	読取可能なコードの 1 桁目の文字		✓
secondCharacter : 2 桁目の文字	""	限定しない	○	✓
	"?"	限定しない		✓
	"0" ~ "9"	読取可能なコードの 2 桁目の文字		✓
BarcodeScannerSettings.decode.symbologies.upcE.addOn : UPC-E アドオン関連				
enabled : アドオン付き	true	読取許可		✓
	false	読取禁止	○	✓
addOn2Digit : 2 桁アドオン付き	true	読取許可		/
	false	読取禁止	○	
addOn5Digit : 5 桁アドオン付き	true	読取許可		/
	false	読取禁止	○	
onlyWithAddOn : アドオン付きのみ	true	アドオン付きのみに限定する		/
	false	アドオン付きのみに限定しない	○	

プロパティ : 説明	設定値		初期値	SP1
BarcodeScannerSettings.decode.symbologies.itf: ITF 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
lengthMin : 最小桁数	2～99	最小桁数	4	✓
lengthMax : 最大桁数	2～99	最大桁数	99	✓
verifyCheckDigit : チェックデジット検証	true	有効		✓
	false	無効	○	✓
BarcodeScannerSettings.decode.symbologies.stf: STF 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
lengthMin : 最小桁数	2～99	最小桁数	3	✓
lengthMax : 最大桁数	2～99	最大桁数	99	✓
verifyCheckDigit : チェックデジット検証	true	有効		✓
	false	無効	○	✓
startStopCharacter : スタート ストップ キャラクタ形式	""	限定しない	○	
	"S"	Short		
	"N"	Normal		
BarcodeScannerSettings.decode.symbologies.codabar: Codabar 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
lengthMin : 最小桁数	3～99	最小桁数	4	✓
lengthMax : 最大桁数	3～99	最大桁数	99	✓
verifyCheckDigit : チェックデジット検証	true	有効		✓
	false	無効	○	✓
startStopCharacter : スタート ストップ キャラクタ	""(空文字列)	スタートストップを限定しない	○	✓
	スタートキャラクタと ストップキャラクタを 連結させた文字列 ex) "AA", "AB" "?"を指定すると、スタ ート/ストップのいづれ か一方のみ指定可能 ex) "A?", "?D"	指定されたスタート/ストップ キャラクタをもつラベルのみ読取可能		✓
BarcodeScannerSettings.decode.symbologies.code39: Code39 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
lengthMin : 最小桁数	1～99	最小桁数	1	✓
lengthMax : 最大桁数	1～99	最大桁数	99	✓
verifyCheckDigit : チェックデジット検証	true	有効		✓
	false	無効	○	✓
BarcodeScannerSettings.decode.symbologies.code93: Code93 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
lengthMin : 最小桁数	1～99	最小桁数	1	✓
lengthMax : 最大桁数	1～99	最大桁数	99	✓

プロパティ : 説明	設定値		初期値	SP1
BarcodeScannerSettings.decode.symbologies.code128: Code128 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
lengthMin : 最小桁数	1～99	最小桁数	1	✓
lengthMax : 最大桁数	1～99	最大桁数	99	✓
BarcodeScannerSettings.decode.symbologies.msi: MSI 関連				
enabled : 読取許可	true	読取許可		
	false	読取禁止	○	
lengthMin : 最小桁数	1～99	最小桁数	1	
lengthMax : 最大桁数	1～99	最大桁数	99	
numberOfCheckDigit Verification : チェック デジットに使用する桁数	1	1 桁	○	
	2	2 桁		
BarcodeScannerSettings.decode.symbologies.gs1DataBar: GS1 Databar 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
stacked : スタック形式読取許可	true	読取許可		✓
	false	読取禁止	○	✓
BarcodeScannerSettings.decode.symbologies.gs1DataBarLimited: GS1 Databar Limited 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
BarcodeScannerSettings.decode.symbologies.gs1DataBarExpanded: GS1 Databar Expanded 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
stacked : スタック形式読取許可	true	読取許可		✓
	false	読取禁止	○	✓
lengthMin : 最小桁数	1～99	最小桁数	1	✓
lengthMax : 最大桁数	1～99	最大桁数	99	✓
BarcodeScannerSettings.decode.symbologies.gs1Composite: GS1 Composite 関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓

プロパティ : 説明	設定値		初期値	SP1
BarcodeScannerSettings.decode.symbologies.qrCode: QR コード関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
splitMode : 連結読取モード	SplitModeQr.DISABLED	連結読取禁止	○	✓
	SplitModeQr.EDIT	編集モード		✓
	SplitModeQr.BATCH_EDIT	一括編集モード		✓
	SplitModeQr.NON_EDIT	未編集モード		✓
BarcodeScannerSettings.decode.symbologies.qrCode.model1: QR モデル 1 コード関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
versionMin : 最小コードバージョン	1~22	最小バージョンの値	1	✓
versionMax : 最大コードバージョン	1~22	最大バージョンの値	22	✓
BarcodeScannerSettings.decode.symbologies.qrCode.model2: QR モデル 2 コード関連				
enabled : 読取許可	true	読取許可	○	✓
	false	読取禁止		✓
versionMin : 最小コードバージョン	1~40	最小バージョンの値	1	✓
versionMax : 最大コードバージョン	1~40	最大バージョンの値	40	✓
BarcodeScannerSettings.decode.symbologies.microQr: Micro QR コード関連				
enabled : 読取許可	true	読取許可	○	✓
	False	読取禁止		✓
versionMin : 最小コードバージョン	1~4	最小バージョンの値	1	✓
versionMax : 最大コードバージョン	1~4	最大バージョンの値	4	✓
BarcodeScannerSettings.decode.symbologies.sqrc: SQRC コード関連				
enabled : 読取許可	true	読取許可		✓
	False	読取禁止	○	✓
versionMin : 最小コードバージョン	1~40	最小バージョンの値	1	✓
versionMax : 最大コードバージョン	1~40	最大バージョンの値	40	✓
BarcodeScannerSettings.decode.symbologies.iqrCode: iQR コード関連				
enabled : 読取許可	true	読取許可		
	false	読取禁止	○	
splitMode : 連結読取モード	SplitModelqr.DISABLED	読取禁止	○	
	SplitModelqr.EDIT	編集モード		
	SplitModelqr.NON_EDIT	未編集モード		
BarcodeScannerSettings.decode.symbologies.iqrCode.square: 正方形 iQR コード関連				
enabled : 読取許可	true	読取許可		
	false	読取禁止	○	
versionMin : 最小コードバージョン	1~61	最小バージョンの値	1	
versionMax : 最大コードバージョン	1~61	最大バージョンの値	61	
BarcodeScannerSettings.decode.symbologies.iqrCode.rectangle: 長方形 iQR コード関連				
enabled : 読取許可	true	読取許可		
	false	読取禁止	○	
versionMin : 最小コードバージョン	1~15	最小バージョンの値	1	
versionMax : 最大コードバージョン	1~15	最大バージョンの値	15	

※BarcodeScannerSettings.decode.symbologies.qrCode.enabled = false にすると、model1 と model2 と microQr も自動的に false となります。
BarcodeScannerSettings.decode.symbologies.qrCode.enabled = true にすると同時に、model1 と model2 と microQr の設定も同時におこなうようにしてください。

プロパティ : 説明	設定値		初期値	SP1
BarcodeScannerSettings.decode.symbologies.dataMatrix: Data Matrix コード関連				
enabled	true	読取許可	○	✓
: 読取許可	false	読取禁止		✓
BarcodeScannerSettings.decode.symbologies.dataMatrix.square: Data Matrix 正方形コード関連				
enabled	true	読取許可	○	✓
: 読取許可	false	読取禁止		✓
codeNumberMin	1~24	最小コード番号の値	1	✓
: 最小コード番号				
codeNumberMax	1~24	最大コード番号の値	24	✓
: 最大コード番号				
BarcodeScannerSettings.decode.symbologies.dataMatrix.rectangle: Data Matrix 長方形コード関連				
enabled	True	読取許可	○	✓
: 読取許可	False	読取禁止		✓
codeNumberMin	1~6	最小コード番号の値	1	✓
: 最小コード番号				
codeNumberMax	1~6	最大コード番号の値	6	✓
: 最大コード番号				
BarcodeScannerSettings.decode.symbologies.pdf417: PDF417 コード関連				
enabled	True	読取許可	○	✓
: 読取許可	False	読取禁止		✓
BarcodeScannerSettings.decode.symbologies.microPdf417: Micro PDF417 コード関連				
enabled	true	読取許可	○	✓
: 読取許可	false	読取禁止		✓
BarcodeScannerSettings.decode.symbologies.maxiCode: Maxi コード関連				
enabled	true	読取許可	○	✓
: 読取許可	false	読取禁止		✓
BarcodeScannerSettings.decode.symbologies.plessey: Plessey コード関連				
enabled	true	読取許可		
: 読取許可	false	読取禁止	○	
BarcodeScannerSettings.decode.symbologies.aztec: Aztec コード関連				
enabled	true	読取許可		
: 読取許可	false	読取禁止	○	

プロパティ : 説明	設定値		初期値	SP1
BarcodeScannerSetting.editing : 読み取ったデータの編集関連				
BarcodeScannerSetting.editing.ean13 : 読み取った EAN-13 コードの編集関連				
reportCheckDigit : チェックデジット付加	true	付加する	○	✓
	false	付加しない		✓
BarcodeScannerSetting.editing.upcA : 読み取った UPC-A コードのデータ編集関連				
reportCheckDigit : チェックデジット付加	true	付加する	○	✓
	false	付加しない		✓
addLeadingZero : 転送桁数調整用先頭キャラクタ付加	true	付加する	○	✓
	false	付加しない		✓
BarcodeScannerSetting.editing.ean8 : 読み取った EAN-8 コードのデータ編集関連				
reportCheckDigit : チェックデジット付加	true	付加する	○	✓
	false	付加しない		✓
convertToEan13 : EAN13 への変換	true	変換する		✓
	false	変換しない	○	✓
BarcodeScannerSetting.editing.upcE : 読み取った UPC-E コードのデータ編集関連				
reportCheckDigit : チェックデジット付加	true	付加する	○	✓
	false	付加しない		✓
addLeadingZero : 転送桁数調整用先頭キャラクタ付加	true	付加する		✓
	false	付加しない	○	✓
convertToUpcA : UPC-A への変換	true	変換する		✓
	false	変換しない	○	✓
reportNumberSystem CharacterOf ConvertedUpcA : UPC-A へ変換した場合の ナンバーシステム付加	true	付加する	○	✓
	false	付加しない		✓
BarcodeScannerSetting.editing.code39 : 読み取った Code39 コードのデータ編集関連				
reportCheckDigit : チェックデジット付加	true	付加する	○	✓
	false	付加しない		✓
reportStartStopCharacter : スタートストップ付加	true	付加する		✓
	false	付加しない	○	✓
BarcodeScannerSetting.editing.codabar : 読み取った Codabar コードのデータ編集関連				
reportCheckDigit : チェックデジット付加	true	付加する	○	✓
	false	付加しない		✓
reportStartStopCharacter : スタートストップ付加	true	付加する	○	✓
	false	付加しない		✓
convertToUpperCase : スタートストップの 大文字変換	true	変換する		✓
	false	変換しない	○	✓
BarcodeScannerSetting.editing.itf : 読み取った ITF コードのデータ編集関連				
reportCheckDigit : チェックデジット付加	true	付加する	○	✓
	false	付加しない		✓
BarcodeScannerSetting.editing.stf : 読み取った STF コードのデータ編集関連				
reportCheckDigit : チェックデジット付加	true	付加する	○	✓
	false	付加しない		✓
BarcodeScannerSetting.editing.sqrc : 読み取った SQRC コードのデータ編集関連				
correctKeyDecode : SQRC 出力データ部	CorrectKeyDecode. PUBLIC_AND_PRIVATE_DATA	公開部と非公開部	○	✓
	CorrectKeyDecode. ONLY_PRIVATE_DATA	非公開部のみ		✓
incorrectKeyDecode : SQRC 暗号キー不一致時の 出力データ部	IncorrectKeyDecode.NONE	出力なし	○	✓
	IncorrectKeyDecode. ONLY_PUBLIC_DATA	公開部のみ		✓

プロパティ : 説明	設定値		初期値	SP1
BarcodeScannerSettings.PowerOffDelay :				
time :closeBarcode 後、バーコード スキャナモジュールを電 源 OFF するまでの時間	最小値 0。int 型	OFF するまでの時間 (ミリ秒)	0	

DENSO SP1 SDK for Android 解説書 第 7 版

2021 年 9 月

株式会社デンソーウェーブ AUTO-ID 事業部

- このマニュアルの一部または全部を無断で複製・転載することはお断りします。
- このマニュアルの内容は将来予告なしに変更することがあります。
- このマニュアルを使用した結果の損害については責任を負いかねます。