unitech
because we care

UnitechSDK User Manual V1.0.0

日本語版

for EA630/PA760/HT730/WD200

改訂履歴

Version	改訂履歴
1.0.0	● 日本語版リリース

目次

第 1 章　はじめに

1.1　概要

UnitechSDK for Android は、ユニテックの Android モバイルデバイス用のデバイス制御ライブラリです。これは、Java(JAR) ベースの API で、アプリケーション開発者はユニテックモバイルデバイス特有の機能をビジネスアプリケーションに統合することができます。

この SDK は、ユニテックモバイルデバイス用のアプリケーションを作成する開発者に以下のようなメリットを提供します。

- ユニテック特有のファームウェア機能の設定
- ユニテック・アプリケーションの設定
- システム機能の呼び出しが可能
- 主流の Android Studio で動作
- 対象となるパートナーには無償でライセンスされます

1.2　ユニテック特有のファームウェア機能の設定

次のリストはユニテック特有のファームウェア機能です。開発者は API を使用してこれらの機能を設定することができます。

特徴	説明
デバイス情報	デバイスのシリアル番号と OS のビルド番号を取得します。
キーマップ	キーマップを書き換えて、別のキーコードをエミュレートしたり、アプリを起動したりすることができます。例えば、「VOLUME_UP」キーと「VOLUME_DOWN」キーを何もしないように書き換えて無効にしたり、「FUNCTION」キーを書き換えて、ボイスチャットアプリを起動したりすることができます。
デバイス制御 (DMI)	DMI(Device Management Interface) を使用して特定のデバイス機能を無効にし、ユーザーがその機能を使用することを制限します。例えば、カメラ機能を無効にすると、ユーザーはシステム内のどこでもカメラを使用することができなくなります。製品によって、カメラ、フラッシュライト、バーコードスキャナ、GPS、Bluetooth、Wi-Fi、4G/LTE、USB ファイル転送、仮想キーボード、物理キー、タッチパネルなどが含まれます。
OS アップデート	デバイスの内部または外部ストレージに保存されている、アップデート用の ZIP ファイルを使って OS をアップデートできます。ZIP ファイルは、内蔵ストレージ、SD カード、フラッシュドライブに入れることができます。
セーフモードロック	Android のセーフモードで起動した場合、デバイスをロックすることができます。セーフモードロック機能を有効にすると、デバイスがセーフモードに入ると、自動的にロックアウト画面が表示され、デバイスが正常に再起動されない限り、ユーザーはそれ以上デバイスを操作することはできません。
グローブモード	グローブモードのタッチ感度設定を有効または無効にします。この機能は、一部のユニテック製品でのみサポートされています。
WLAN Advanced	Wi-FI によるローミングの詳細なパラメータを設定します。

1.3　ユニテック・アプリケーションの設定

次のリストはユニテック特有のアプリケーションの機能です。開発者は API を使用してこれらの機能を設定することができます。MoboLink、StageGO、Unitech Scan Service、Startup、Software Update、WLAN Advanced 以外のアプリケーションは、必要に応じてユーザーによるインストールが必要です。

ユニテックアプリ	設定の種類
MoboLink	アプリの設定
StageGO	アプリの設定
Unitech Scanner Utility (USU)	アプリの設定。設定のインポート、エクスポート
Unitech Scan Service (USS)	設定のインポート、エクスポート
Startup	OS 起動時に自動起動するアプリの設定
ELauncher	アプリの設定
Software Update	アプリの設定。設定のインポート、エクスポート、リセット
RFID2KEY	アプリの設定。設定のインポート、エクスポート、リセット
WLAN Advanced	アプリの設定。設定のインポートとエクスポート

1.4　システム機能の呼び出し

通常、これらの Android 標準 API を呼び出すアプリは、デバイス固有のプラットフォームキーで署名されたシステムアプリでなければならず、またユーザーに権限の許可を求める必要があります。UntiechSDK では、これらの機能をプラットフォームキーで署名することなく、ユーザーに権限の許可を求めることなく呼び出すことができます。エンタープライズアプリケーションの GUI をより目立たなくすることができます。

機能領域	プロキシの Android API 機能が利用可能
アプリ制御	アプリのインストール、削除、実行、終了、有効化、無効化、データのクリア、アプリ一覧の取得、アプリ情報の取得など。
ファイル制御	読み取り、書き込み、削除、コピー、作成、ディレクトリの一覧表示、名前の変更など。
オーディオ	ボリュームの設定、着信音やサウンドの設定など。
時計	日付と時刻、タイムゾーン、NTP サーバーアドレスの設定など。
ディスプレイ	ディスプレイのタイムアウト、画面の明るさ、向き、自動回転、フォントサイズの設定。
4G/LTE & Wi-Fi	APN プロファイルの設定、WiFi への接続、プロキシの設定、キャプティブポータル、静的 IP の設定など。
セキュリティ	ロックスクリーン、IME、USB デバッグ、ADB ログの保存など。
システム	言語、位置情報、Bluetooth、NFC、出荷時リセット、再起動。

1.5　主流の Android Studio で動作

Android Studio は、世界中の開発者が使用している主流の IDE です。開発者は、以下の前提条件を満たす限り、Windows、MAC、Linux 上で Android アプリを開発することができます：**Android Studio Version 4.1 以降**

1.6　価格と使用範囲

UnitechSDK は、資格のあるユニテックパートナーに無料で提供されます。UnitechSDK ライセンスの対象となるのは、以下のパートナーです。

- UnitechSDK でサポートされる 1 つ以上の製品を購入している。
- UnitechSDK ライセンス契約に完全に同意している。ライセンス契約については、UnitechSDK パッケージに含まれている、LICENSE ファイルをお読みください。

1.7　リリース情報

リリース日	2021 年 6 月 30 日
最新版	V1.1.19
ダウンロードリンク	http://w3.tw.ute.com/pub/sw/UnitechSDK/1.1.19/UnitechSDK_V1.1.19.zip

1.8　既知の問題

問題	影響を受ける機能	回避方法
パラメータに空の値を入力すると、予期しない動作をする。	AppManagementCtrl → deactivateApp AppManagementCtrl → runSysCmd AppManagementCtrl → clearDefaultApps DebuggingCtrl → saveLogcatFile StageGoCtrl → setReportMaxNum StageGoCtrl → enabledBootupScript RFID2KeyCtrl → setMode USUCtrl → SetScannerSymbologyConfiguration	空のパラメータを関数に渡さないでください。 この問題は次のリリースで修正される予定です。

1.9　サポートされている製品

Device Model	Android OS	Minimum OS Version	Minimum API Level
PA760	9	MR6	24
PA760	10	MR1	24
EA630	9	173	24
EA630	10	092	24
WD200	10	CR3	24
HT730	10	1.0.13	24
TB85Plus	10	RU.00.53.84.09	24

1.10　依存関係にあるアプリ

UnitechSDK を使用するには、依存関係にあるアプリとして DMI Service と AppCtrlService が必要です。通常これらのアプリはデバイスにあらかじめインストールされています。もしもデバイスにインストールされているアプリのバージョンがが要件を満たさない場合、UnitechSDK のパッケージに含まれる APK を使用して更新またはインストールしてください。

✎ UnitechSDK V1.1.xx は、DMService V1.2.xx をサポートしておらず、一部の API が動作しませんのでご注意ください。UnitechSDK の次回の更新で、DMService V1.2.xx がサポートされる予定です。

依存関係にあるアプリ	最小バージョン
DMService	V1.1.18
AppCtrlSerive	V1.0.7

1.11　FAQ

質問	回答
UntiechSDK を使って初めてアプリケーションを作成する前に、どのようなドキュメントを検討すればよいのでしょうか？	以下の資料を用意しています。 - Introduction Presentation（PowerPoint ファイル。英語のみ。） http://w3.tw.ute.com/pub/sw/UnitechSDK/1.1.19/UnitechSDK_V1.1_Introduction_20210630.pptx - API ドキュメント (Javadoc) - ソースコード付きサンプルアプリ（Android Studio プロジェクト ）
私のアプリで、logcat ログに「java.lang.IllegalStateException: calling this from your main thread can lead to deadlock」エラーメッセージが発生します。この問題を解決する方法を教えてください。	このエラーは、通常、SDK への関数呼び出しがメインスレッドに常駐している場合に発生します。UnitechSDK を使用するには、関数呼び出しを別のスレッドで実行する必要があります。ソースコード付きのサンプルアプリを参照して、この仕組みを理解することができます。
アプリを実行すると、logcat ログに DMService や AppCtrlService のバインディングエラーが出るのですが、この問題を解決するにはどうしたらよいでしょうか？	アプリが UnitechSDK の JAR ライブラリで作成されている場合、依存関係にあるアプリの DMService と AppCtrlService の正しいバージョンがターゲットデバイスにインストールされている必要があります。依存関係にあるアプリの APK ファイルは、UnitechSDK パッケージに含まれています。アプリは、UnitechSDK の機能を呼び出す前に、以下のチェックを行う必要があります。 - DMService と AppCtrlService がデバイスにインストールされているかどうかを確認します。 - DMService と AppCtrlService の最小バージョンがデバイスにインストールされているかどうか。 - 依存関係にあるアプリがインストールされていない場合、またはバージョンの要件が満たされていない場合、アプリは事前に APK をインストールする必要があります。

第 2 章　使用方法

2.1　プロジェクトのビルド依存関係にライブラリを追加する

ダウンローした ZIP ファイル内の UnitechSDK.jar ファイルを、Android Studio のプロジェクト内の libs フォルダにコピーします。プロジェクトの build.gradle ファイルの dependencies に implementation などで指定します。

build.gradle

```
plugins {...}

android {...}

dependencies {
      implementation fileTree(dir: 'libs', include: ['*.jar'])
      ......
}
```

2.2　希望する API を初期化する

希望する API をそれぞれ初期化してください。以下は、ApnCtrl を初期化する例です。

Java

```
AppCtrl apnCtrl = new ApnCtrl(this.getApplicationContext());
```

Kotlin

```
private val apnCtrl by lazy { ApnCtrl(applicationContext) }
```

2.3　希望する API を使用する

メインスレッド以外から呼び出してください。以下は機内モードを ON にするときの例です。

Java

```
public class MainActivity extends AppCompatActivity {

      @Override
      protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);

           ApnCtrl apnCtrl = new ApnCtrl(this.getApplicationContext());

           ExecutorService executor = Executors.newSingleThreadExecutor();
           CompletableFuture<Bundle> future = CompletableFuture
                    .supplyAsync(() -> apnCtrl.setAirplaneMode(true), executor);

           future.whenComplete((bundle, e) -> {
                if (e == null) {
                     int errCode = bundle.getInt("errorCode");
                     String errMsg = bundle.getString("errorMsg");
                } else {
                     e.printStackTrace();
                }
           });
      }
}
```

Kotlin (コルーチンを使用)

```
class MainActivity : AppCompatActivity() {

      private val apnCtrl by lazy { ApnCtrl(applicationContext) }

      override fun onCreate(savedInstanceState: Bundle?) {
           super.onCreate(savedInstanceState)
           setContentView(R.layout.activity_main)

           CoroutineScope(Dispatchers.Main).launch {
                val bundle = setAirplaneMode(true)
                val errCode = bundle.getInt("errorCode")
                val errMsg = bundle.getString("errorMsg")
           }
      }

      private suspend fun setAirplaneMode(bEnabled: Boolean): Bundle =
           withContext(Dispatchers.Default) {
                apnCtrl.setAirplaneMode(bEnabled)
           }
}
```

第 3 章　モバイルネットワーク

パッケージ名 : com.unitech.api.apn

詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

3.1　APN プロファイルデータの追加

メソッド	ApnAddProfile(String profileData)
profileData	型　String 値　「;」コロンで区切られた 22 個の APN プロファイルオプション
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

3.2　APN プロファイルの削除

メソッド	ApnDeleteProfile(String apnName)
apnName	型　String 値　APN プロファイルの名前
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

3.3　APN プロファイルの存在を確認

メソッド	ApnProfileIsExist(String apnName)
apnName	型　String 値　APN プロファイルの名前
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

3.4　APN プロファイルデータの更新

メソッド	ApnUpdateProfile(String profileData)
profileData	型　String 値　「;」コロンで区切られた 22 個の APN プロファイルオプション
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

3.5　すべての APN プロファイルを削除

メソッド	ApnDeleteAll()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

3.6　アクティブな APN プロファイルに設定

メソッド	ApnSetActiveProfile(String apnName)
apnName	型　String 値　APN プロファイルの名前
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

3.7　アクティブな APN プロファイルを取得

メソッド	ApnGetActivieProfileApnName()
apnName	型　String 値　APN プロファイルの名前
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　AppName 値　アクティブなプロファイルの名前 型　String

3.8　APN プロファイル名のリストを取得

メソッド	ApnGetNameList()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　ApnNameList 値　既存プロファイルの名前リスト 型　StringArrayList

3.9　APN プロファイルデータを取得

メソッド	ApnGetProfile(String apnName)
apnName	型　String 値　APN プロファイルの名前
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　_id name　名前 numeric mcc　MCC mnc　MNC apn　APN user　ユーザー名 server　サーバー password　パスワード proxy　プロキシ port　ポート mmsproxy　MMS プロキシ mmsport　MMS ポート mmsc　MMSC authtype　認証タイプ type　APN タイプ current protocol　APN プロトコル roaming_protocol　APN ローミングプロトコル carrier_enabled　APN の有効 / 無効 bearer　ベアラー mvno_type　MVNO の種類 mvno_match_data　MVNO 値 値　プロファイルオプション 型　String

第 4 章　アプリケーション

パッケージ名 : com.unitech.api.app
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

4.1　APK のパッケージ名を取得

メソッド	getAppPackageName(String apkPath)
apkPath	型　String 値　APK ファイルのパス (例 : /xrom/apps/UNote.apk)
返値	値　パッケージ名 型　String

4.2　アプリをインストール

メソッド	installApp(String apkFilePath, String pkgName)
apkFilePath	型　String 値　APK ファイルのパス
pkgName	型　String 値　アプリのパッケージ名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

4.3　アプリをアンインストール

メソッド	removeApp(String pkgName)
pkgName	型　String 値　アプリのパッケージ名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

4.4　アプリをアクティブ

メソッド	activateApp(String pkgName)
pkgName	型　String 値　アプリのパッケージ名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

4.5　アプリを非アクティブ

メソッド	deactivateApp(String pkgName)
pkgName	型　String 値　アプリのパッケージ名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

4.6　アプリを有効化

メソッド	enableApp(String pkgName)
pkgName	型　String 値　アプリのパッケージ名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

4.7　アプリを無効化

メソッド	disableApp(String pkgName)
pkgName	型　String 値　アプリのパッケージ名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

4.8　アプリのストレージデータとキャッシュデータを削除

メソッド	clearAppData(String pkgName)
pkgName	型　String 値　アプリのパッケージ名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

4.9　実行中のアプリのラベル名を取得

メソッド	getRunningAppNameList()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　AppNameList 値　アプリ名のリスト 型　ArrayList<String>

4.10　実行中のアプリのパッケージ名を取得

メソッド	getRunningPkgNameList()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　PkgNameList 値　パッケージ名のリスト 型　ArrayList<String>

4.11　インストールされているアプリのラベル名を取得

メソッド	getInstallAppNameList()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　AppNameList 値　アプリ名のリスト 型　ArrayList<String>

4.12　インストールされているアプリのパッケージ名を取得

メソッド	getInstallPkgNameList()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　PkgNameList 値　パッケージ名のリスト 型　ArrayList<String>

4.13　インストールされているシステムアプリのラベル名を取得

メソッド	getBuiltinSystemAppNameList()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　AppNameList 値　アプリ名のリスト 型　ArrayList<String>

4.14　インストールされているシステムアプリのパッケージ名を取得

メソッド	getBuiltinSystemPkgNameList()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　PkgNameList 値　パッケージ名のリスト 型　ArrayList<String>

4.15　無効化されているアプリのラベル名を取得

メソッド	getDisabledAppNameList()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　AppNameList 値　アプリ名のリスト 型　ArrayList<String>

4.16　無効化されているアプリのパッケージ名を取得

メソッド	getDisabledPkgNameList()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　PkgNameList 値　パッケージ名のリスト 型　ArrayList<String>

4.17　OS 更新

メソッド	osUpdate(String updateZipPath)
updateZipPath	型　String 値　更新用の OS ファイル (ZIP 圧縮されたファイル)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

4.18　システムユーザーとしてシェルコマンドを実行

メソッド	runSysCmd(String command)
command	型　String 値　シェルコマンド
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　SysCmdResult 値　コマンドの実行結果 型　String

4.19　アプリ名からバージョン名とバージョンコードを取得

メソッド	getAppInfoByAppName(String appName)
appName	型　String 値　アプリのラベル名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　AppInfo 値　アプリの情報 型　Bundle
AppInfo	キー　AppVersionName 値　アプリのバージョン名 型　String キー　AppVersionCode 値　アプリのバージョンコード 型　String

4.20　パッケージ名からバージョン名とバージョンコードを取得

メソッド	getAppInfoByPkgName(String pkgName)
pkgName	型　String 値　アプリのパッケージ名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　AppInfo 値　アプリの情報 型　Bundle
AppInfo	キー　AppVersionName 値　アプリのバージョン名 型　String キー　AppVersionCode 値　アプリのバージョンコード 型　String

4.21　デフォルトのアプリを設定

メソッド	setDefaultApp(String action, String[] categories, String packageName, String activityName)
action	型　String 値　インテントフィルターの問合せに必要なアクション
categories	型　String 配列 値　インテントフィルターの問合せに必要なカテゴリーの配列
packageName	型　String 値　アプリのパッケージ名
activityName	型　String 値　アプリのアクティビティ名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

■　Unitech Elauncher をデフォルトのホームアプリに設定する場合の例

Java

```
String act = "android.intent.action.MAIN";
String[] cat = {"android.intent.category.HOME", "android.intent.category.DEFAULT"};
String pkgName = "com.ute.eu.ELauncher";
String activityName = "com.ute.eu.ELauncher.Launcher";

AppManagementCtrl appCtrl = new AppManagementCtrl(this.getApplicationContext());

ExecutorService executor = Executors.newSingleThreadExecutor();
CompletableFuture<Bundle> future = CompletableFuture
          .supplyAsync(() -> appCtrl.setDefaultApp(act, cat, pkgName, activityName), executor);
```

Kotlin (コルーチンを使用)

```
val appCtrl = AppManagementCtrl(applicationContext)

val act = "android.intent.action.MAIN"
val cat = arrayOf("android.intent.category.HOME", "android.intent.category.DEFAULT")
val pkgName = "com.ute.eu.ELauncher"
val activityName = "com.ute.eu.ELauncher.Launcher"

CoroutineScope(Dispatchers.Main).launch {
      withContext(Dispatchers.Default) {
           appCtrl.setDefaultApp(act, cat, pkgName, activityName)
      }.let { bundle ->
           val errCode = bundle.getInt("errorCode")
           val errMsg = bundle.getString("errorMsg")
      }
}
```

4.22　デフォルトのアプリを消去

メソッド	clearDefaultApps(String[] packages) clearDefaultApps(String action, String[] categories)
packages	型　String 配列 値　消去したいアプリのパッケージ名の配列
action	型　String 値　インテントフィルターの問合せに必要なアクション
categories	型　String 配列 値　インテントフィルターの問合せに必要カテゴリーの配列
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 5 章　オーディオ

パッケージ名 : com.unitech.api.audio
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

5.1　デフォルトの通知音を設定

メソッド	setDefaultNotificationSound(String data)
data	型　String 値　通知音のラベル名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

5.2　着信音を設定

メソッド	setRingtoneSound(String data)
data	型　String 値　着信音のラベル名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

5.3　ダイヤルパッドの操作音をオフ / オン

メソッド	setDialPadTouchTones(boolean enabled)
enabled	型　Boolean 値　true= オン、false= オフ
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

5.4　タッチの操作音をオフ / オン

メソッド	setTouchSound(boolean enabled)
enabled	型　Boolean 値　true= オン、false= オフ
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

5.5　タッチ操作時のバイブをオフ / オン

メソッド	setVibrateOnTouch(boolean enabled)
enabled	型　Boolean 値　true= オン、false= オフ
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

5.6　音のボリュームの変更

メソッド	setVolume(int type, int volumeValue)
type	型　Int 値　2= 着信音、3= メディア、4= アラーム
volumeValue	型　Int 値　0 (最小) - 100 (最大)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 6 章　Bluetooth

パッケージ名 : com.unitech.api.bluetooth
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

6.1　インスタンスの取得

メソッド	getInstance(Context context)
context	型　Context 値　コンテキスト
返値	型　BluetoothCtrl 値　BluetoothCtrl のインスタンス

6.2　デバイスの Bluetooth アドレスを取得

メソッド	getMacAddress()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　BluetoothMacAddress 値　Bluetooth アドレス 型　String

第 7 章　時計

パッケージ名 : com.unitech.api.clock
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

7.1　NTP サーバーを設定

メソッド	setNTPServer(String NTPServerURL)
NTPServerURL	型　String 値　NTP サーバーの URL アドレス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

7.2　現在の NTP サーバーを取得

メソッド	getNTPServer()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　NtpServer 値　NTP サーバー 型　String

7.3　日付と時刻の同期モードを設定

メソッド	setTimeMode(int mode)
mode	型　Int 値　0= 手動モード、1= 自動モード (ネットワーク合わせ)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

7.4　現在の日付と時刻の同期モードを取得

メソッド	getTimeMode()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　TimeMode 値　0= 手動モード、1= 自動モード (ネットワーク合わせ) 型　Int

7.5　日付を設定

日付と時刻の同期モードが「手動モード」に設定されている必要があります。

メソッド	setManualDate(String date)
date	型　String 値　日付の文字列、例 :31/12/2019 (dd/mm/yyyy)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

7.6　時刻を設定

日付と時刻の同期モードが「手動モード」に設定されている必要があります。

メソッド	setManualTime(String time)
time	型　String 値　時刻の文字列、例 :23:59 (時 : 分)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

7.7　現在の時刻を取得

メソッド	getManualTime()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　ManualTime 値　日付と時刻の文字列 (dd/MM/yyyy HH:mm) 型　String

7.8　タイムゾーンの同期モードを設定

メソッド	setTimeZoneMode(int mode)
mode	型　Int 値　0= 手動モード、1= 自動モード (ネットワーク合わせ)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

7.9　現在のタイムゾーンの同期モードを取得

メソッド	getTimeZoneMode()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　TimeZoneMode 値　0= 手動モード、1= 自動モード (ネットワーク合わせ) 型　Int

7.10　タイムゾーンを設定

タイムゾーンの同期モードが「手動モード」に設定されている必要があります。

メソッド	setTimeZone(String timeZone)
timeZone	型　String 値　タイムゾーンの文字列、例 : 日本標準時 =Asia/Tokyo
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

7.11　現在のタイムゾーンを取得

メソッド	getTimeZone()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　TimeZone 値　タイムゾーンの文字列 型　String

7.12　時刻形式を設定

メソッド	setTimeFormat(int timeFormat)
timeFormat	型　Int 値　12=12 時間表示、24=24 時間表示
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

7.13　現在の時刻形式を取得

メソッド	getTimeFormat()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　TimeFormat 値　12=12 時間表示、24=24 時間表示 型　Int

第 8 章　デバッギング

パッケージ名 : com.unitech.api.debugging
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

8.1　logcat ファイルの保存

メソッド	saveLogcatFile(String filePath, long seconds)
filePath	型　String 値　保存パス
seconds	型　Long 値　保存する秒数
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　Pid 値　PID 型　Int

8.2　logcat ファイルの保存を停止

メソッド	stopSaveLogcat(int pid)
pid	型　Int 値　実行中の PID
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第９章　ディスプレイ

パッケージ名 : com.unitech.api.display
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

9.1　画面消灯の時間を設定

メソッド	SetDisplayTimeout(String timeout)
timeout	型　String 値　Never =　なし 15 seconds または 15 =　15 秒 30 seconds または 30 =　30 秒 1 minute または 60 =　1 分 2 minutes または 120 =　2 分 5 minutes または 300 =　5 分 10 minutes または 600 =　10 分 30 minutes または 1800 =　30 分
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

9.2　現在の画面消灯の時間を取得

メソッド	GetDisplayTimeout()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　DisplayTimeout 値　Never =　なし 15 =　15 秒 30 =　30 秒 60 =　1 分 120 =　2 分 300 =　5 分 600 =　10 分 1800 =　30 分 型　String

9.3　充電中の画面消灯モードを設定

メソッド	SetStayAwake(int input)
input	型　Int 値　0 = 充電中でも画面を消灯する (標準値) 1 = 充電中は画面を消灯しない
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

9.4　現在の充電中の画面消灯モードを取得

メソッド	GetStayAwake()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　StayAwake 値　0 = 充電中でも画面を消灯する (標準値) 1 = 充電中は画面を消灯しない 型　Int

9.5　画面の回転状態を設定

メソッド	SetScreenOrientation(int orientation)
orientation	型　Int 値　0 = 回転しない　SCREEN_ROTATION_DISABLE 1 = 0°　SCREEN_ROTATION_PORTRAIT 2 = 90°　SCREEN_ROTATION_LANDSCAPE 3 = 180°　SCREEN_ROTATION_REVERSE_PORTRAIT 4 = 270°　SCREEN_ROTATION_REVERSE_LANDSCAPE 5 = 自動回転　SCREEN_ROTATION_AUTO ※すべて反時計回り
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

9.6　ロック画面上の通知を設定

メソッド	ShowLockScreenNotification(boolean show)
show	型　Boolean 値　true = 警告やサイレント通知を表示する false = 通知を表示しない
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

9.7　画面上のテキストサイズを設定

メソッド	SetFontSize(String data)
data	型　String 値　small= 小、default= デフォルト、large= 大、largest= 最大
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

9.8　画面上のテキストサイズを取得

メソッド	GetFontSize()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　FontSize 値　small= 小、default= デフォルト、large= 大、largest= 最大 型　String

9.9　画面の明るさのレベルを設定

メソッド	SetScreenBacklightLevel(int value)
value	型　Int 値　0-255
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

9.10　画面の明るさのレベルを取得

メソッド	GetScreenBacklightLevel()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　ScreenBacklightLevel 値　明るさのレベル 型　Int

9.11　画面の明るさの自動調節モードをオフ / オン

メソッド	SetAutoBrightness(int mode)
mode	型　Int 値　0= オフ、1= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

9.12　画面の自動回転をオフ / オン

メソッド	SetAutoRotation(boolean enabled)
enabled	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

9.13　電池のパーセント表示をオフ / オン

メソッド	setDisplayBatteryPercentage(boolean enable)
enable	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

9.14　グローブモードをオフ / オン

メソッド	setGloveMode(boolean enabled)
enabled	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

9.15　現在のグローブモードの状態を取得

メソッド	getGloveMode()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　GloveMode 値　グローブモードの状態 型　Boolean

第 10 章　機能制限

パッケージ名 : com.unitech.api.dmi
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

10.1　現在の機能の状態を取得

メソッド	DCMO_Get(String cmdData)		
cmdData	型	String	
	値	Touch	DMI_TOUCH
		PhysicalKeyboard	DMI_PHYSICAL_KEYBOARD
		USB	DMI_USB
		UsbMtpMode	DMI_USB_MTP_MODE
		GPS	DMI_GPS
		WWAN	DMI_WWAN
		Scanner	
		Camera	DMI_CAMERA
		VirtualKeyboard	DMI_VIRTUAL_KEYBOARD
		WLAN	DMI_WLAN
		Bluetooth	DMI_BLUETOOTH
		Flashlight	DMI_FLASHLIGHT
		DozeMode	DMI_DOZE_MODE
返値 (Bundle)	キー	errorCode	
	値	0= 成功、1= 失敗	
	型	int	
	キー	errorMsg	
	値	エラーメッセージ	
	型	String	
	キー	Result	
	値	結果	
	型	Boolean	
	キー	Status	
	値	0= オン、1= オフ、2= 未サポート	
	型	Int	

10.2　機能の状態を設定

メソッド	DCMO_Set(String cmdData, int enable)	
cmdData	型	String
	値	Touch　DMI_TOUCH
		PhysicalKeyboard　DMI_PHYSICAL_KEYBOARD
		USB　DMI_USB
		UsbMtpMode　DMI_USB_MTP_MODE
		GPS　DMI_GPS
		WWAN　DMI_WWAN
		Scanner
		Camera　DMI_CAMERA
		VirtualKeyboard　DMI_VIRTUAL_KEYBOARD
		WLAN　DMI_WLAN
		Bluetooth　DMI_BLUETOOTH
		Flashlight　DMI_FLASHLIGHT
		DozeMode　DMI_DOZE_MODE
enable	型	Int
	値	0= オフ、1= オン
返値 (Bundle)	キー	errorCode
	値	0= 成功、1= 失敗
	型	int
	キー	errorMsg
	値	エラーメッセージ
	型	String
	キー	Result
	値	結果
	型	Boolean
	キー	Status
	値	0= オフ、1= オン、2= 未サポート
	型	Int

第 11 章　ファイル操作

パッケージ名 : com.unitech.api.file
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

11.1　インスタンスの取得

メソッド	getInstance(Context context)
context	型　Context 値　コンテキスト
返値	型　FileCtrl 値　FileCtrl のインスタンス

11.2　UTF-8 形式のファイルを作成してデータを書き込む

追記はせず、常に新しいファイルが作成されます。

メソッド	writeUTF8ToFile(String path, String content)
path	型　String 値　ファイルパス
content	型　String 値　内容
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

11.3　ファイルを作成してデータを書き込む

追記はせず、常に新しいファイルが作成されます。

メソッド	writeToFile(String path, byte[] content)
path	型　String 値　ファイルパス
content	型　ByteArray 値　内容
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

11.4　ファイルから内容を読み込む

メソッド	readFromFile(String path)
path	型　String 値　ファイルパス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　Data 値　内容 型　ByteArray

11.5　ファイルコピー

メソッド	copyFile(String fromPath, String toPath)
fromPath	型　String 値　コピー元ファイルパス
toPath	型　String 値　コピー先ファイルパス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

11.6　空のファイルまたはフォルダを作成

メソッド	createFile(String path)
path	型　String 値　ファイルパスまたはフォルダパス ("/" で終わる)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

11.7　ファイルまたはフォルダ削除

メソッド	deleteFile(String path)
path	型　String 値　ファイル名を含むパスまたはフォルダパス ("/" で終わる)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

11.8　ファイルストリームを作成

ファイルにデータを読み書きするために使用できるストリームを作成します。このストリームに対する操作が完了した場合は、closeFileStream を呼び出して、このストリームを閉じます。

メソッド	createFileStream(String filepath, boolean keepContents)
fromPath	型　String 値　ファイルパス
keepContents	型　String 値　コピー先ファイルパス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　ParcelFileDescriptor 値　ディスクリプタ 型　Parcelable キー　FileStreamID 値　ストリーム ID 型　Long

■　サンプルコード

Java

```
public class MyFileCtrl {

	private final FileCtrl fileCtrl;

	public MyFileCtrl(final Context context) {
		fileCtrl = FileCtrl.getInstance(context);
	}

	public ParcelFileDescriptor createFileStream(String filepath, Boolean keepContents) {
		Bundle bundle = fileCtrl.createFileStream(filepath, keepContents);
		long fileStreamId = bundle.getLong("FileStreamID");
		fileCtrl.closeFileStream(fileStreamId);
		return bundle.getParcelable("ParcelFileDescriptor");
	}

	public void writeToFile(ParcelFileDescriptor pfd, byte[] msg) {
		try (FileOutputStream fos = new FileOutputStream(pfd.getFileDescriptor())) {
			fos.write(msg);
			fos.flush();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}

// 呼び出し側
MyFileCtrl myFileCtrl = new MyFileCtrl(getApplicationContext());
String filepath = getExternalFilesDir(null).getAbsolutePath() + "/TestFile.txt";
boolean keepContents = true;
byte[] writeMsg = "Test\n".getBytes();
ExecutorService executor = Executors.newSingleThreadExecutor();
CompletableFuture
		.supplyAsync(() -> myFileCtrl.createFileStream(filepath, keepContents), executor)
		.thenAccept(pfd -> myFileCtrl.writeToFile(pfd, writeMsg));
```

Kotlin

```
class MyFileCtrl(
      private val context: Context,
      private val defaultDispatcher: CoroutineDispatcher = Dispatchers.Default,
      private val ioDispatcher: CoroutineDispatcher = Dispatchers.IO
) {
      private val fileCtrl by lazy { FileCtrl.getInstance(context) }

      suspend fun createFileStream(
           filepath: String,
           keepContents: Boolean
      ): ParcelFileDescriptor? = withContext(defaultDispatcher) {
           val bundle = fileCtrl.createFileStream(filepath, keepContents)
           val fileStreamId = bundle.getLong("FileStreamID")
           // closeFileStream を呼び出す
           fileCtrl.closeFileStream(fileStreamId)
           bundle.getParcelable("ParcelFileDescriptor")
      }

      suspend fun writeToFile(pfd: ParcelFileDescriptor?, msg: ByteArray) =
           withContext(ioDispatcher) { execute(pfd, msg) }

      private fun execute(pfd: ParcelFileDescriptor?, msg: ByteArray) {
           FileOutputStream(pfd?.fileDescriptor).use { fos ->
                fos.write(msg)
                fos.flush()
           }
      }
}

// 呼び出し側
val mainScope = CoroutineScope(Dispatchers.Main)
val myFileCtrl = MyFileCtrl(applicationContext)
val filePath = getExternalFilesDir(null)?.absolutePath + "/TestFile.txt"
val keepContents = true
val writeMsg = "Test\n".toByteArray()
mainScope.launch {
      val pfd = myFileCtrl.createFileStream(filePath, keepContents)
      pfd?.let { myFileCtrl.writeToFile(it, writeMsg) }
}
```

11.9　ファイルストリームを閉じる

メソッド	closeFileStream(long fileStreamId)
fileStreamId	型　Long 値　ストリーム ID
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

11.10　ファイルまたはフォルダの存在を確認

メソッド	exists(String path)
path	型　String 値　ファイルまたはフォルダパス
返値 (Bundle)	キー　errorCode 値　0= 存在する、1= 存在しない 型　int キー　errorMsg 値　メッセージ 型　String

11.11　フォルダかどうか確認

メソッド	isDirectory(String path)
path	型　String 値　フォルダパス
返値 (Bundle)	キー　errorCode 値　0= フォルダです、1= フォルダではありません 型　int キー　errorMsg 値　メッセージ 型　String

11.12　ファイルを開く

メソッド	openFile(String path)
path	型　String 値　ファイルパス
返値 (Bundle)	キー　errorCode 値　0= ファイルオープン成功、1= ファイルオープンできない 型　int キー　errorMsg 値　エラーメッセージ 型　String

11.13　特定のパスのファイル名およびフォルダ名のリストを取得

メソッド	list(String path)
path	型　String 値　フォルダパス
返値 (Bundle)	キー　errorCode 値　0= リストは null ではない、1= リストは null 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　List 値　ファイル名およびフォルダ名のリスト 型　StringArray

11.14　ファイル名の変更

メソッド	renameTo(String path, String newName)
path	型　String 値　ファイルパス
newName	型　String 値　新しいファイル名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

11.15　ファイルが書き込み可能か確認

メソッド	canWrite(String path)
path	型　String 値　ファイルパス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

11.16　ファイルが読み込み可能か確認

メソッド	canRead(String path)
path	型　String 値　ファイルパス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 12 章　サイレント OS アップデート

パッケージ名 : com.unitech.api.fota
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

12.1　サイレント OS アップデート

メソッド	silentOSUpdate(String updateZipPath)
updateZipPath	型　String 値　更新用の ZIP ファイルパス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 13 章　一般的な情報

パッケージ名 : com.unitech.api.general
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

13.1　デバイスの言語の変更

メソッド	setLanguage(String language)
language	型　String 値　言語 (例 : en-us= 英語（米国)、ja-jp= 日本語)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

13.2　デバイスの仮想キーボードの変更

メソッド	setImeCurrentKeyboard(String data)
data	型　String 値　パッケージ名、またはパッケージ名 / クラス名 IME のリストを問い合わせるには、たとえば「adb shell ime list -a」を使用します。 gboard=　com.google.android.inputmethod.latin/com.android.inputmethod.latin.LatinIME Google ピンイン入力 =　com.google.android.inputmethod.pinyin/.PinyinIME
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

13.3　デバイスのシリアル番号を取得

メソッド	getDeviceSerialNumber()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　SerialNumber 値　シリアル番号 型　String

13.4　デバイスの OS ビルド番号を取得

メソッド	getDeviceBuildNumber()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　BuildNumber 値　OS ビルド番号 型　String

第 14 章　キーマップ

パッケージ名 : com.unitech.api.keymap
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

✎ デバイス特有のキー定義については、138 ページの「付録 E デバイスのキーマッピング」を参照してください。

14.1　インスタンスの取得

メソッド	getInstance(Context context)
context	型　Context 値　コンテキスト
返値	型　KeymappingCtrl 値　KeymappingCtrl のインスタンス

14.2　特定のキーにキーマップを追加

メソッド	addKeyMappings(String keyName, String keyCode, boolean wakeup, String broadcastDownAction, Bundle[] broadcastDownParams, String broadcastUpAction, Bundle[] broadcastUpParams, Bundle[] startActivityParams)
keyName	型　String 値　キーマップを追加するキーの名前
keyCode	型　String 値　キーコード＝キー押下時のキーコードを変換 アプリ＝キー押下時にアプリを起動 (クラス名とパッケージ名半角スペースで区切ってセット)
wakeup	型　Boolean 値　キー押下時にスリープを解除 (true= 解錠する、false= 解除しない)
broadcastDownAction	型　String 値　キーダウン時のブロードキャストアクション
broadcastDownParams	型　BundleArray 値　キーダウン時のインテント・エキストラ
broadcastUpAction	型　String 値　キーアップ時のブロードキャストアクション
broadcastUpParams	型　BundleArray 値　キーアップ時のインテント・エキストラ
startActivityParams	型　BundleArray 値　アプリ実行時のインテント・エキストラ
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

■　サンプルコード

ボリュームアップキーを無効化する（ボリュームアップキーに、キーコード「0」をセットする）

Java

```
KeymappingCtrl keymappingCtrl = KeymappingCtrl.getInstance(getApplicationContext());
ExecutorService executor = Executors.newSingleThreadExecutor();
CompletableFuture.runAsync(() -> keymappingCtrl.addKeyMappings(
          "VOLUME_UP", "0", false, null, null, null, null, null), executor);
```

Kotlin

```
val defaultScope = CoroutineScope(Dispatchers.Default)
val keymappingCtrl = KeymappingCtrl.getInstance(applicationContext)
defaultScope.launch {
     keymappingCtrl.addKeyMappings("VOLUME_UP", "0", false, null, null, null, null, null)
}
```

■　サンプルコード

ボリュームアップキーのキーダウンにアクションとエキストラをセットしてブロードキャストする

Java

```
KeymappingCtrl keymappingCtrl = KeymappingCtrl.getInstance(getApplicationContext());
String broadcastDownAction = "VOLUME_UP.down";
Bundle[] broadcastDownParams = new Bundle[1];
Bundle downParamsBundle = new Bundle();
downParamsBundle.putString("Key", "volUpDown");
downParamsBundle.putString("Value", "VolDownValue");
broadcastDownParams[0] = downParamsBundle;
ExecutorService executor = Executors.newSingleThreadExecutor();
CompletableFuture.runAsync(() -> keymappingCtrl.addKeyMappings(
          "VOLUME_UP", "24", false,
          broadcastDownAction, broadcastDownParams, null, null, null), executor);
```

Kotlin

```
val defaultScope = CoroutineScope(Dispatchers.Default)
val keymappingCtrl = KeymappingCtrl.getInstance(applicationContext)
val broadcastDownAction = "VOLUME_UP.down"
val broadcastDownParams = arrayOf(Bundle().apply {
     putString("Key", "volUpDown")
     putString("Value", "VolDownValue")
})
defaultScope.launch {
     keymappingCtrl.addKeyMappings(
          "VOLUME_UP", "24", false,
          broadcastDownAction, broadcastDownParams, null, null, null)
}
```

■　サンプルコード

ボリュームアップキーを UNote アプリの実行ショートカットに設定する

Java

```java
KeymappingCtrl keymappingCtrl = KeymappingCtrl.getInstance(getApplicationContext());
// ショートカットに設定するアプリの "クラス名{半角スペース}パッケージ名"
String keyCode = "com.unitech.unote.UNote com.unitech.unote";
ExecutorService executor = Executors.newSingleThreadExecutor();
CompletableFuture.runAsync(() -> keymappingCtrl.addKeyMappings(
          "VOLUME_UP", keyCode, false, null, null, null, null, null), executor);
```

Kotlin

```kotlin
val defaultScope = CoroutineScope(Dispatchers.Default)
val keymappingCtrl = KeymappingCtrl.getInstance(applicationContext)
// ショートカットに設定するアプリの "クラス名{半角スペース}パッケージ名"
val keyCode = "com.unitech.unote.UNote com.unitech.unote"
defaultScope.launch {
      keymappingCtrl.addKeyMappings(
          "VOLUME_UP", keyCode, false, null, null, null, null, null
      )
}
```

14.3　すべてのキーマップをリセット

メソッド	resetKeyMappings()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

14.4　プログラム可能なキーサービスのオフ / オン

メソッド	enableKeyMapping(boolean enabled)
enabled	型　Boolean 値　true= オン、false= オフ
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

14.5　すべてのキーマップをリセットして初期化

メソッド	resetKeyMappings()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

14.6　キーマップの設定情報をファイルからインポート

メソッド	importKeyMappings(String folderPath)
folderPath	型　String 値　フォルダパス ("/" で終わる)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

14.7　キーマップの設定情報をファイルへエクスポート

メソッド	exportKeyMappings(String folderPath)
folderPath	型　String 値　フォルダパス ("/" で終わる)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

14.8　特定のキーの現在のキーマップを取得

メソッド	enableKeyMapping(boolean enabled)
enabled	型　Boolean 値　true= オン、false= オフ
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 15 章　位置情報

パッケージ名 : com.unitech.api.location
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

15.1　位置情報モードを設定

メソッド	setLocationMode(int mode)
mode	型　Int 値　0= オフ、1= デバイスのみ、2= バッテリーセーブ、3= 高精度
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 16 章　NFC

パッケージ名 : com.unitech.api.nfc
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

16.1　NFC アダプタのオフ / オン

メソッド	setNfcAdapter(boolean enabled)
enabled	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 17 章　電源

パッケージ名 : com.unitech.api.power
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

17.1　デバイスリセット（初期化）

すべての情報をファイルを初期化し、デバイスを工場出荷状態へリセットします。

メソッド	ColdBoot()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　Result 値　false= 失敗、true= 成功 型　Boolean

17.2　デバイスリブート（再起動）

メソッド	WarmBoot()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String キー　Result 値　false= 失敗、true= 成功 型　Boolean

17.3　Wake Lock を取得

PARTIAL_WAKE_LOCK レベルの CPU 動作を保証します。電源ボタンを押しても、画面とキーボードはオフになりますが、CPU は走り続けます。そのため、この API の使用は、デバイスのバッテリー寿命に大きく影響を及ぼす可能性があります。

timeout に 1 以上の値を渡した場合、指定されたタイムアウト時間が経過すると Wake Lock は自動的に解除されます。timeout に 0 を渡した場合、Wake Lock を解除するためには releaseWakeLock を呼び出す必要があります。

メソッド	acquireWakeLock(long timeout)
timeout	型　Long 値　Wake Lock を自動的に解除するためのタイムアウト時間 (ミリ秒)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

17.4　Wake Lock を解除

メソッド	releaseWakeLock()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 18 章　セーフモードロック

パッケージ名 : com.unitech.api.safemodelock
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

18.1　インスタンスの取得

メソッド	getInstance(Context context)
context	型　Context 値　コンテキスト
返値	型　SafeModeLockCtrl 値　SafeModeLockCtrl のインスタンス

18.2　セーフモードロック

セーフモードは、ユーザーによる簡単な電源ボタンの操作によって実行することができます。そのため、ロックダウンアプリによる指定されたアプリ以外の実行制限や設定の変更禁止をセーフモードをバイパスして簡単に実行することができたり、最悪なことにロックダウンアプリそのものをアンインストールすることができてしまいます。

この API は、セーフモードの使用を制限することができます。enabled に true を渡すと、セーフモードロック状態になります。セーフモードロック状態のときに、デバイスがセーフモードを検出すると、SafeMode detected! Device is locked by administrator. Please shutdown the device and then turn it again のメッセージを画面にフルスクリーンで表示し、その他の機能にアクセスできないようにデバイスを制限します。

launcherPackageName に、特定のランチャーアプリのパッケージ名を渡すと、メッセージの代わりに、そのランチャーアプリが実行されます。空文字を渡すと、ランチャーアプリの実行は解除されます。。

メソッド	setSafeModeLock(boolean enabled, String launcherPackageName)
enabled	型　Boolean 値　Wake Lock を自動的に解除するためのタイムアウト時間 (ミリ秒)
launcherPackageName	型　String 値　セーフモード時のランチャーアプリのパッケージ名
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 19 章　バーコードスキャナ

パッケージ名 : com.unitech.api.scanner
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

Unitech Scan Service の設定変更やバーコードデータの受信などは、各モデルの Android プログラミングマニュアルを参照してください。

19.1　Unitech Scan Service の設定をファイルからインポート

メソッド	importSettings(String folderPath)
folderPath	型　String 値　フォルダパス ("/" で終わる)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

19.2　Unitech Scan Service の設定をファイルへエクスポート

メソッド	exportSettings(String folderPath)
folderPath	型　String 値　フォルダパス ("/" で終わる)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 20 章　セキュリティ

パッケージ名 : com.unitech.api.security
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

20.1　USB デバッグのオフ / オン

メソッド	setAdbDebugging(boolean enabled)
enabled	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 21 章　Elauncher

パッケージ名 : com.unitech.api.uapps
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

21.1　リリース情報

リリース日	2021 年 8 月 26 日
最新版	V1.5.30
ダウンロードリンク	http://w3.tw.ute.com/pub/sw/ELauncher/1.5.30/ELauncher_v1.5.30_ea630.apk http://w3.tw.ute.com/pub/sw/ELauncher/1.5.30/ELauncher_v1.5.30_pa760.apk http://w3.tw.ute.com/pub/sw/ELauncher/1.5.30/ELauncher_v1.5.30_ht730.apk http://w3.tw.ute.com/pub/sw/ELauncher/1.5.30/ELauncher_v1.5.30_wd200.apk

21.2　Elauncher をキオスクモードで表示

メソッド	setEnabled(boolean enabled)
enabled	型　Boolean 値　false= 表示しない、true= 表示する
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

21.3　キオスクアプリを設定

メソッド	setKioskApp(String pkg)
pkg	型　String 値　パッケージ名 (空文字はキオスクアプリを解除)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

21.4　Elauncher からの実行を許可するアプリリストを設定

メソッド	setLauncherAppList(String apps)
apps	型　String 値　カンマ区切りのアプリのラベル名 (例 : 設定 ,UNote,Chrome) 言語によってアプリのラベル名が変化する可能性があることに注意してください。たとえば OS の言語設定が日本語のとき、設定アプリのラベル名は " 設定 " ですが、英語のときは "Settings" です。
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

21.5　Elauncher の壁紙を設定

メソッド	setWallpaper(String path)
path	型　String 値　画像ファイルのパス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

21.6　Elauncher の PIN コードを設定

Elauncher の設定変更や、Elauncher の終了を制限するための PIN コードを設定します。

メソッド	setPINCode(String PIN)
PIN	型　String 値　PIN コード
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

21.7　キオスクアプリの実行モードを設定

事前に「21.3 キオスクアプリを設定」をセットしてください。

メソッド	setMode(int mode)
mode	型　Integer 値　0= ランチャーモード、1= キオスクモード
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 22 章　MoboLink クライアント

パッケージ名 : com.unitech.api.uapps
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

22.1　MoboLink サーバーの URL を設定

メソッド	SetServerURL(String serverURL)
serverURL	型　String 値　MoboLink サーバー URL アドレス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

22.2　デバイス再起動時にクライアントの設定を消去

メソッド	SetRebootstrap(Boolean rebootstrap)
rebootstrap	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

22.3　サイレントモードのオフ / オン

メソッド	SetShowUI(Boolean showUI)
showUI	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

22.4　接続 / 切断通知を表示

メソッド	SetConnectionNotification(Boolean connectionNotification)
connectionNotification	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

22.5　イベント通知を表示

メソッド	SetEventNotification(Boolean eventNotification)
eventNotification	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

22.6　ユーザーによる設定変更の制限

メソッド	SetUserConfigurable(Boolean userConfigurable)
userConfigurable	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

22.7　デバイス再起動時の自動実行

メソッド	SetAutoRun(Boolean autoRun)
autoRun	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

22.8　MoboLink コンフィグを設定

メソッド	SetMoboLinkConfig(String serverUrl, Boolean rebootstrap, Boolean showUI, Boolean connectionNotification, Boolean eventNotification, Boolean userConfigurable, Boolean autoRun)
serverURL	型　String 値　MoboLink サーバー URL アドレス
rebootstrap	型　Boolean 値　false= オフ、true= オン
showUI	型　Boolean 値　false= オフ、true= オン
connectionNotification	型　Boolean 値　false= オフ、true= オン
eventNotification	型　Boolean 値　false= オフ、true= オン
userConfigurable	型　Boolean 値　false= オフ、true= オン
autoRun	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 23 章　RFID2KEY

パッケージ名 : com.unitech.api.uapps
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

23.1　設定をファイルからインポート

メソッド	importSettings(String filePath, String passCode)
filePath	型　String 値　ファイル名を含む完全パス
passCode	型　String 値　パスコード (設定されていない場合は空文字)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

23.2　設定をファイルにエクスポート

メソッド	exportSettings(String filePath)
filePath	型　String 値　ファイル名を含む完全パス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

23.3　設定をリセット

メソッド	resetSettings(String passCode)
passCode	型　String 値　パスコード (設定されていない場合は空文字)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

23.4　動作モードを切り替え

メソッド	setMode(Boolean mode, String passCode)
mode	型　Boolean 値　false= スキャナモード、true=RFID モード
passCode	型　String 値　パスコード (設定されていない場合は空文字)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 24 章　Software Update

パッケージ名 : com.unitech.api.uapps
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

24.1　設定更新

メソッド	updateConfig(String serverUrl, int opmode, int policyInstalled, String selectedApps, int policyNew, String passcode)
serverURL	型　String 値　Software Update サーバー URL アドレス
opmode	型　Integer 値　Operation Mode 0=Normal Mode、1=Auto Mode、2=Auto Silent Mode、3=Disable
policyInstalled	型　Integer 値　Application Update Policy 0=All Installed App、1=None、2=Selected
selectedApps	型　String 値　更新対象として選択するカンマ区切りのアプリ名 (例 : KeyRemap,Software Update,USS,UNote,StageGO,OEMConfig)
policyNew	型　Integer 値　New Applications 0=None、1=Install
passcode	型　String 値　パスコード
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

24.2　設定をファイルにエクスポート

メソッド	exportSettings(String filepath)
filepath	型　String 値　ファイル名を含む完全なパス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

24.3　設定をファイルからインポート

メソッド	importSettings(String filepath)
filepath	型　String 値　ファイル名を含む完全なパス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

24.4　設定をリセット

メソッド	resetSettings()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 25 章　StageGO

パッケージ名 : com.unitech.api.uapps
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

25.1　ログファイルの保存場所を設定

メソッド	setReportPath(String filepath)
filepath	型　String 値　フォルダパス (最後は "/" で終わる) 例 : /= ルートフォルダ、/Download/=Download フォルダ
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

25.2　ログファイルの最大保存数を設定

メソッド	setReportMaxNum(int num)
num	型　Integer 値　ログファイルの最大保存数
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

25.3　ブートアップスクリプトのオフ / オン

ブートアップスクリプトがオンになっていると、デバイスの起動後に指定された StageGo スクリプトファイル (*.stagego) を自動的に実行します。

メソッド	enabledBootupScript(boolean enabled)
enabled	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

25.4　ブートアップスクリプトファイルの設定

メソッド	setBootupScript(String filepath)
filepath	型　String 値　ファイルパス (カンマ区切りで複数指定可能) 例 : /StageGO.stagego,/Download/StageGO.stagego
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

25.5　パスコードの設定

メソッド	setPasscode(String passcode)
passcode	型　String 値　パスコード (空文字はパスコードの削除を意味します)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

25.6　スキャンモードの設定

メソッド	setScanMode(String scanMode)
scanMode	型　String 値　CameraMode= カメラモード、ScannerMode= スキャナモード
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 26 章　Unitech Scanner Utility (USU)

パッケージ名 : com.unitech.api.usu
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

✎ 「Unitech Scanner Utility V1.3.x.x ユーザー マニュアル」および「Unitech Bluetooth Service API Programming Guide」についてもご確認ください。

26.1　インスタンスの取得

メソッド	getInstance(String packageName, String passcode, Context context)
packageName	型　String 値　パッケージ名
passcode	型　String 値　パスコード
context	型　Context 値　コンテキスト
返値	型　USUCtrl 値　USUCtrl のインスタンス

26.2　最新のリザルトコードを取得

メソッド	getLastResultCode()
返値	型　Integer 値　0= 成功、1= 失敗

26.3　最新のリザルトメッセージを取得

メソッド	getLastResultMessage()
返値	型　String 値　リザルトメッセージ

26.4　パスコードの変更

メソッド	changePasscode(String passcode)
passcode	型　String 値　パスコード

■　サンプルコード

「26.5 ペアリングバーコードの取得」

Java

```
String pkgName = this.getPackageName();
String passcode = "";
Context context = this.getApplicationContext();
USUCtrl usuCtrl = USUCtrl.getInstance(pkgName, passcode, context);
ExecutorService executor = Executors.newSingleThreadExecutor();

CompletableFuture.runAsync(() -> {
      usuCtrl.GetPairingBarcode();
      int resultCode = usuCtrl.getLastResultCode();
      String resultMessage = usuCtrl.getLastResultMessage();
      Log.d(TAG, String.format("lastResultCode: %s", resultCode));
      Log.d(TAG, String.format("lastResultMessage: %s", resultMessage));
      if (!usuCtrl.replyQueue.isEmpty()) {
           Bundle bundle = usuCtrl.replyQueue.poll();
           if (bundle != null) {
                bundle.keySet().forEach(key ->
                          Log.d(TAG, String.format("%s: %s", key, bundle.get(key))));
           }
      }
}, executor);
```

Kotlin

```
val defaultScope = CoroutineScope(Dispatchers.Default)
val pkgName = this.packageName
val passcode = ""
val context = this.applicationContext
val usuCtrl = USUCtrl.getInstance(pkgName, passcode, context)

defaultScope.launch {
      usuCtrl.GetPairingBarcode()
      val resultCode = usuCtrl.lastResultCode
      val resultMessage = usuCtrl.lastResultMessage
      Log.d(tag, "resultCode: $resultCode")
      Log.d(tag, "resultMessage: $resultMessage")
      if (!usuCtrl.replyQueue.isEmpty()) {
           usuCtrl.replyQueue.poll()?.let { bundle ->
                bundle.keySet().forEach { key -> Log.d(tag, "$key: ${bundle.get(key)}") }
           }
      }
}
```

26.5　ペアリングバーコードの取得

メソッド	void GetPairingBarcode()
レスポンス	キー　PairingBarcodeContent 値　ペアリングバーコード 型　String キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_PAIRING_BARCODE_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.6　スキャナの情報を取得

メソッド	void GetPairingBarcode()
レスポンス	キー　IsConnected 値　false= 切断中、true= 接続中 型　Boolean キー　serialNo 値　シリアル番号 型　String キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_TARGET_SCANNER_STATUS_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.7　スキャナにペアリング解除を要求

メソッド	void AskScannerToUnpair()
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_UNPAIR_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.8　スキャナのシリアル番号を取得

メソッド	void GetScannerSerialNumber()
レスポンス	キー　sn 値　シリアル番号 型　String キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_SCANNER_SERIAL_NUMBER_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.9　スキャナの Bluetooth デバイス名を取得

メソッド	void GetScannerBluetoothName()
レスポンス	キー　name 値　Bluetooth デバイス名 型　String キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_SCANNER_BLUETOOTH_NAME_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.10　スキャナの Bluetooth アドレスを取得

メソッド	void GetScannerBluetoothMacAddress()
レスポンス	キー　address 値　Bluetooth アドレス 型　String キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_SCANNER_BLUETOOTH_MAC_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.11　スキャナのファームウェアバージョンを取得

メソッド	void GetScannerFirmwareVersion()
レスポンス	キー　fw 値　ファームウェアバージョン 型　String キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_SCANNER_FIRMWARE_VERSION_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.12　スキャナのバッテリレベルを取得

メソッド	void GetScannerBatteryLevel()
レスポンス	キー　battery 値　0=charging、1=very low、2=low、3=ok、4=full 型　Integer キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_BATTERY_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.13　スキャナのトリガーキーの状態を取得

メソッド	void GetScannerTriggerKey()
レスポンス	キー　trig 値　false= 無効、true= 有効 型　Boolean キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_TRIG_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.14　スキャナのトリガーキーの状態を設定

メソッド	void SetScannerTriggerKey(boolean enable)
enable	型　Boolean 値　false= 無効、true= 有効
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_SET_TRIG_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.15　デコード開始（読み取り開始）

メソッド	void StartDecode()
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_START_DECODE_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.16　デコード停止（読み取り停止）

メソッド	void StopDecode()
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_STOP_DECODE_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.17　スキャナの BT SPP ACK/NAK の状態を取得

メソッド	void GetDataACK()
レスポンス	キー　ack 値　false= 無効、true= 有効 型　Boolean キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_DATA_ACK_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.18　スキャナの BT SPP ACK/NAK の状態を設定

メソッド	void SetDataACK(boolean enable)
enable	型　Boolean 値　false= 無効、true= 有効
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_SET_DATA_ACK_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.19　スキャナの BT SPP 自動接続の状態を取得

メソッド	void GetAutoConnection()
レスポンス	キー　autoConn 値　false= 無効、true= 有効 型　Boolean キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_AUTO_CONNECTION_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.20　スキャナの BT SPP 自動接続の状態を設定

メソッド	void SetAutoConnection(boolean enable)
enable	型　Boolean 値　false= 無効、true= 有効
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_SET_AUTO_CONNECTION_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.21　スキャナのバーコード設定の取得

メソッド	void GetScannerSymbologyConfiguration()
レスポンス	キー　付録 A(111 ページ) 参照 値　設定値 型　付録 A(111 ページ) 参照 キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_CONFIG_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.22　スキャナのバーコード設定の変更

メソッド	void SetScannerSymbologyConfiguration(String symbologyName, int value)
symbologyName	型　String 値　付録 A(111 ページ) 参照
value	型　Integer 値　付録 A(111 ページ) 参照
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_SET_CONFIG_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.23　スキャナの Bluetooth 信号チェックレベルの取得

メソッド	void GetScannerBtSignalCheckingLevel()
レスポンス	キー　btSignalCheckingLevel 値　0= 標準レベル、1= 高レベル 型　Integer キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_BLUETOOTH_SIGNAL_CHECKING_LEVEL_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.24　スキャナの Bluetooth 信号チェックレベルの設定

メソッド	void SetScannerBtSignalCheckingLevel(int level)
level	型　Integer 値　0= 標準レベル、1= 高レベル
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_SET_BLUETOOTH_SIGNAL_CHECKING_LEVEL_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.25　スキャナのターミネーター情報の取得

メソッド	void GetScannerDataTerminator()
レスポンス	キー　DataTerminator 値　0= なし、1=CR、2=LF、3=CR+LF、4=TAB 型　Integer キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_GET_DATA_TERMINATOR_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.26　スキャナのターミネーター情報の設定

メソッド	void SetScannerDataTerminator(int terminator)
terminator	型　Integer 値　0= なし、1=CR、2=LF、3=CR+LF、4=TAB
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_SET_DATA_TERMINATOR_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.27　ACK とインジケータの送信

メソッド	void SendACKIndicator(boolean withACK, int beepTime, boolean vibrate, String ledColor)
withACK	型　Boolean 値　false= インジケータのみ、true=ACK とインジケータ
beepTime	型　Integer 値　0= ビープなし、1= ビープ 1 回、2= ビープ 2 回、3= ビープ 3 回
vibrate	型　Boolean 値　false= 振動させない、true= 振動させる
ledColor	型　String 値　none　=LED 点灯なし red　= 赤色 LED 点灯 green　= 緑色 LED 点灯 blue　= 青色 LED 点灯
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_SET_INDICATOR_REPLY 型　String キー　packageName 値　パッケージ名 型　String

26.28　設定のエクスポート

メソッド	void ImportSettings(String filepath)
filepath	型　Sting 値　ファイル名を含むフルパス
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_EXPORT_SETTINGS_REPLY 型　String キー　packageName 値　パッケージ名 型　String キー　message 値　エクスポート成功失敗メッセージ 型　String

26.29　設定のインポート

メソッド	void ImportSettings(String filepath)
filepath	型　Sting 値　ファイル名を含むフルパス
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_IMPORT_SETTINGS_REPLY 型　String キー　packageName 値　パッケージ名 型　String キー　message 値　インポート成功失敗メッセージ 型　String

26.30　USU の設定をスキャナにアップロードして更新

メソッド	void UploadSettings()
レスポンス	キー　result 値　0= 成功、その他 = 失敗 型　Integer キー　replyType 値　API_IMPORT_SETTINGS_REPLY 型　String キー　packageName 値　パッケージ名 型　String

第 27 章　WLAN Advanced

パッケージ名 : com.unitech.api.wlan
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

27.1　インスタンスの取得

メソッド	getInstance(Context context)
context	型　Context 値　コンテキスト
返値	型　WlanAdvancedCtrl 値　WlanAdvancedCtrl のインスタンス

27.2　設定をファイルからインポート

メソッド	importSettings(String folderPath)
folderPath	型　String 値　フォルダパス ("/" で終わる)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

27.3　設定をファイルにエクスポート

メソッド	exportSettings(String folderPath)
folderPath	型　String 値　フォルダパス ("/" で終わる)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

27.4　設定を変更

メソッド	setNewSetting(int powerSaveMode, int roamingTrigger, int roamingDelta, int roamingScanPeriod)
powerSaveMode	型　Integer 値　0=Power Save Mode 無効、1=Power Save Mode 無効
roamingTrigger	型　Integer 値　10 ～ 120 (-1dBm 単位。30 を入力すると -30 dBm の意味になります)
roamingDelta	型　Integer 値　0 ～ 30 (1dB 単位
roamingScanPeriod	型　Integer 値　0 ～ 60 (100 ミリ秒単位)
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　int キー　errorMsg 値　エラーメッセージ 型　String

第 28 章　WLAN (WiFi)

パッケージ名 : com.unitech.api.wlan
詳細は、UnitechSDK のパッケージに同梱されている javadoc で確認してください。

28.1　WiFi プロファイルを設定

メソッド	setWifiProfile(String ssid, String password, int securityMode)
ssid	型　String 値　SSID
password	型　String 値　パスワード
securityMode	型　Integer 値　0= 暗号化なし、1=WEP、2=WPA または WPA2
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.2　WiFi プロファイルを取得

メソッド	getWifiProfile(String ssid)
ssid	型　String 値　SSID
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String キー　SecurityMode 値　0= 暗号化なし、1=WEP、2=WPA または WPA2 (パスワードの取得はできません) 型　Integer

28.3　IP アドレス情報を設定

メソッド	setIpAssignmentProfile(String ssid, int ipAssignment, String ipAddress, int prefixLength, String gateway, String dns1, String dns2)
ssid	型　String 値　SSID
ipAssignment	型　Integer 値　0=DHCP、1= 静的
ipAddress	型　String 値　IP アドレス
prefixLength	型　Integer 値　ネットワークプレフィックス長 (例 : 24=255.255.255.0)
gateway	型　String 値　ゲートウェイ
dns1	型　String 値　DNS 1
dns2	型　String 値　DNS 2
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.4　IP アドレス情報を取得

メソッド	getIpAssignmentProfile(String ssid)
ssid	型　String 値　SSID
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String キー　IpAssignment 値　0=DHCP、1= 静的 型　Integer キー　IpAddress 値　IP アドレス 型　String キー　PrefixLength 値　ネットワークプレフィックス長 (例 : 24=255.255.255.0) 型　Integer キー　Gateway 値　ゲートウェイ 型　String キー　Dns1 値　DNS 1 型　String キー　Dns2 値　DNS 2 型　String

28.5　プロキシ情報を設定

メソッド	setProxyProfile(String ssid, int proxySetting, String pacUrl, String host, int port, String excList)
ssid	型　String 値　SSID
proxySetting	型　Integer 値　0= なし、1= 手動、2= プロキシの自動設定
pacUrl	型　String 値　PAC URL
host	型　String 値　プロキシのホスト名
port	型　Integer 値　プロキシポート
excList	型　String 値　プロキシをバイパス
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.6　プロキシ情報を取得

メソッド	getProxyProfile(String ssid)
ssid	型　String 値　SSID
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String キー　ProxySettings 値　0= なし、1= 手動、2= プロキシの自動設定 型　Integer キー　PacFileUrl 値　PAC URL 型　String キー　Host 値　プロキシのホスト名 型　String キー　Port 値　プロキシポート 型　Integer キー　ExcList 値　プロキシをバイパス 型　String

28.7　WiFi プロファイルを削除

接続中の WiFi プロファイルを削除することはできません。

メソッド	deleteProfile(String ssid)
ssid	型　String 値　SSID
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.8　WiFi プロファイルの生存を確認

メソッド	isProfileExisted(String ssid)
ssid	型　String 値　SSID
返値 (Bundle)	キー　errorCode 値　0= 生存する、1= 生存しない 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.9　保存済みの WiFi プロファイルで接続

メソッド	connectProfile(String ssid)
ssid	型　String 値　SSID
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.10　接続中の WiFi プロファイルを取得

メソッド	getConnectedProfile()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String キー　ssid 値　接続中の SSID 型　String

28.11　保存済みのすべての WiFi プロファイルを削除

接続中の WiFi プロファイルも含めてすべて削除されます。

メソッド	removeAllProfiles()
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.12　WiFi プロファイルがデバイスに保存されているか確認

メソッド	isProfilesEmpty()
返値 (Bundle)	キー　errorCode 値　0=1 つ以上の WiFi プロファイルが保存されている 1=WiFi プロファイルが保存されていない 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.13　キャプティブポータルモードの設定

メソッド	setCaptivePortalMode(int mode)
mode	型　Integer 値　0= キャプティブポータルの検出を試みません 1= キャプティブポータルの検出を試み、ユーザーにサインインを促します 2= 直ちにネットワークから切断し、今後そのネットワークに再接続しません
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.14　HTTP キャプティブポータル URL の設定

新規接続時に HTTP キャプティブポータルの検出に使用する URL を設定します。検証には、サーバーからの 204 応答コードが使用されます。

メソッド	setCaptivePortalHttpUrl(String url)
url	型　String 値　HTTP キャプティブポータル URL
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.15　ネットワークの検証に HTTPS を使用するかどうかを設定

ネットワークの検証に HTTPS を使用するかどうかを設定します。この設定を無効にするには、0 に設定する必要があります。キャプティブポータルは実際には HTTPS を使用していないので、この設定は誤った名称ですが、他の設定の名称と一致させています。

メソッド	setCaptivePortalUseHttps(int mode)
mode	型　Integer 値　0= ネットワークの検証に HTTP を使用する 1= ネットワークの検証に HTTPS を使用する
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.16　HTTPS キャプティブポータル URL の設定

新規接続時に HTTPS キャプティブポータルの検出に使用する URL を設定します。検証には、サーバーからの 204 応答コードが使用されます。キャプティブポータルの特性上、HTTPS によるネットワーク検証はほとんど失敗します。

メソッド	setCaptivePortalHttpsUrl(String url)
url	型　String 値　HTTPS キャプティブポータル URL
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.17　WiFi アダプタのオフ / オン

メソッド	setWifiAdapter(boolean enabled)
enabled	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.18　WiFi エンタープライズプロファイルの設定

メソッド	setWifiEnterpriseProfile(String ssid, String certPath, String identity, String password)
ssid	型　String 値　SSID
certPath	型　String 値　CA 証明書
identity	型　String 値　ID
password	型　String 値　パスワード
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.19　ランダム MAC アドレスの状態を取得

Android 10 以降でサポートされています。

メソッド	getWifiRandomizedMacStatus(String ssid)
ssid	型　String 値　SSID
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

28.20　ランダム MAC アドレスをオフ / オン

Android 10 以降でサポートされています。

メソッド	setWifiRandomizedMac(String ssid, boolean enabled)
ssid	型　String 値　SSID
enabled	型　Boolean 値　false= オフ、true= オン
返値 (Bundle)	キー　errorCode 値　0= 成功、1= 失敗 型　Integer キー　errorMsg 値　エラーメッセージ 型　String

付録 A　バーコード設定

付録 A には、93 ページの「26.22 スキャナのバーコード設定の変更」コマンドおよび 93 ページの「26.21 スキャナのバーコード設定の取得」で開発者が設定できる各設定項目のパラメーターを示しています。

A-1　読み取りを許可する文字数について

いくつかのシンボルの規格、たとえば Code 39、Code 128、QR コードなどには可変長の文字列を含めることができます。スキャナは設定された文字数で構成されたシンボルのみを読み取りの対象とすることができます。たとえば、5 文字から 10 文字の範囲を設定すると、スキャナは 4 文字以下および 11 文字以上のシンボルの読み取りを行いません。

コマンドには、「Length Parameter 1」と「Length Parameter 2」の 2 つのパラメータが存在します。これらを以下のルールで組み合わせて設定します。「Name」の「XXX」はシンボル名が入ります。

a　読み取り可能な文字数を制限しない

「Length Parameter 1」および「Length Parameter 2」の両方に「0」が設定されている場合、スキャナは文字数の制限を行いません。

Name	Type	値
XXX Length Parameter 1	Integer	0
XXX Length Parameter 2	Integer	0

b　1 つの固定長

「Length Parameter 1」に「0」以外の値が設定され、「Length Parameter 2」に「0」が設定されている場合、スキャナは「Length Parameter 1」で設定された数値の文字数を持つシンボルのみを読み取りの対象とします。

Name	Type	値
XXX Length Parameter 1	Integer	0 以外の数値
XXX Length Parameter 2	Integer	0

c　2 つの固定長

「Length Parameter 1」および「Length Parameter 2」に「0」以外の値が設定されており、かつ「Length Parameter 1」の値が「Length Parameter 2」の値よりも大きい (Length 1 > Length 2) 場合、スキャナは「Length Parameter 1」および「Length Parameter 2」で設定された、それぞれの数値の文字数を持つシンボルのみを読み取りの対象とします。

Name	Type	値
XXX Length Parameter 1	Integer	0 以外の数値（数値 大）
XXX Length Parameter 2	Integer	0 以外の数値（数値 小）

d　特定範囲

「Length Parameter 1」および「Length Parameter 2」に「0」以外の値が設定されており、かつ「Length Parameter 1」の値が「Length Parameter 2」の値よりも小さい (Length 1 < Length 2) 場合、スキャナは「Length Parameter 1」から「Length Parameter 2」までの値の範囲の文字数を持つシンボルのみを読み取りの対象とします。

Name	Type	値
XXX Length Parameter 1	Integer	0 以外の数値（数値 小）
XXX Length Parameter 2	Integer	0 以外の数値（数値 大）

設定を行うためのコマンドや初期値は、各シンボルの設定ページをご覧ください。

A-2　UPC/EAN/JAN コード

A-2.1　UPC-A 読み取り許可

Name	Type	設定値	初期値	説明
UPC-A	Integer	0		UPC-A の読み取りを許可しない
		1	✓	UPC-A の読み取りを許可する

A-2.2　UPC-E の読み取り許可

Name	Type	設定値	初期値	説明
UPC-E	Integer	0		UPC-E の読み取りを許可しない
		1	✓	UPC-E の読み取りを許可する

A-2.3　UPC-E1 の読み取り許可

Name	Type	設定値	初期値	説明
UPC-E1	Integer	0	✓	UPC-E1 の読み取りを許可しない
		1		UPC-E1 の読み取りを許可する

A-2.4　EAN-8/JAN-8 の読み取り許可

Name	Type	設定値	初期値	説明
EAN-8/JAN-8	Integer	0		EAN-8/JAN-8 の読み取りを許可しない
		1	✓	EAN-8/JAN-8 の読み取りを許可する

A-2.5　EAN-13/JAN-13 の読み取り許可

Name	Type	設定値	初期値	説明
EAN-13/JAN-13	Integer	0		EAN-13/JAN-13 の読み取りを許可しない
		1	✓	EAN-13/JAN-13 の読み取りを許可する

A-2.6　ISBN の読み取り許可

Name	Type	設定値	初期値	説明
Bookland EAN	Integer	0		ISBN の読み取りを許可しない
		1	✓	ISBN の読み取りを許可する

A-2.7　ISBN の出力フォーマット

112 ページの「A-2.6 ISBN の読み取り許可」が「1」に設定されている必要があります。

Name	Type	設定値	初期値	説明
Bookland ISBN Format	Integer	0	✓	**ISBN-10** -「978」から始まる JAN-13 を旧規格の 10 桁コードとして出力します。このフォーマットでは「979」から始まるコードは ISBN として処理されません。
		1		**ISBN-13** -「978」または「979」から始まる JAN-13 を現行規格の 13 桁コードとして出力します。

A-2.8　UPC/EAN/JAN アドオンコードの読み取り許可

Name	Type	設定値	初期値	説明
Decode UPC/EAN/JAN Supplementals	Integer	0	✓	アドオンコードがあっても無視します（無効）
		1		アドオンコードを送信します（必須）
		2		アドオンコードを自動検出します（自動）
		3		378/379/978/979/977/414/419/434/439/491 から始まる場合は、アドオンコードを送信します
		4		378/379 アドオンモード
		5		978/979 アドオンモード
		6		977 アドオンモード
		7		414/419/434/439 アドオンモード
		8		491 アドオンモード

A-2.9　UPC-A チェックデジットの送信許可

Name	Type	設定値	初期値	説明
Transmit UPC-A Check Digit	Integer	0		UPC-A チェックデジットの送信を許可しない
		1	✓	UPC-A チェックデジットの送信を許可する

A-2.10　UPC-E チェックデジットの送信許可

Name	Type	設定値	初期値	説明
Transmit UPC-E Check Digit	Integer	0		UPC-E チェックデジットの送信を許可しない
		1	✓	UPC-E チェックデジットの送信を許可する

A-2.11　UPC-E1 チェックデジットの送信許可

Name	Type	設定値	初期値	説明
Transmit UPC-E1 Check Digit	Integer	0		UPC-E1 チェックデジットの送信を許可しない
		1	✓	UPC-E1 チェックデジットの送信を許可する

A-2.12　UPC-A プリアンブルの送信許可

Name	Type	設定値	初期値	説明
UPC-A Preamble	Integer	0		UPC-A プリアンブルの送信を許可しない（チェックデジットを含めた 11 桁で送信されます）
		1	✓	UPC-A プリアンブルとして「システムキャラクタ」の送信を許可する（チェックデジットを含めた 12 桁で送信されます）
		2		UPC-A プリアンブルとして「カントリーコード」と「システムキャラクタ」の送信を許可します（カントリーコードの「0」を先頭に挿入した、チェックデジットを含めた 13 桁で送信されます）

A-2.13　UPC-E プリアンブルの送信許可

Name	Type	設定値	初期値	説明
UPC-E Preamble	Integer	0		UPC-E プリアンブルの送信を許可しない（チェックデジットを含めた 7 桁で送信されます）
		1	✓	UPC-E プリアンブルとして「システムキャラクタ」の送信を許可する（チェックデジットを含めた 8 桁で送信されます）
		2		UPC-E プリアンブルとして「カントリーコード」と「システムキャラクタ」の送信を許可します（カントリーコードの「0」を先頭に挿入した、チェックデジットを含めた 9 桁で送信されます）

A-2.14　UPC-E1 プリアンブルの送信許可

Name	Type	設定値	初期値	説明
UPC-E1 Preamble	Integer	0		UPC-E1 プリアンブルの送信を許可しない（チェックデジットを含めた 7 桁で送信されます）
		1	✓	UPC-E1 プリアンブルとして「システムキャラクタ」の送信を許可する（チェックデジットを含めた 8 桁で送信されます）
		2		UPC-E1 プリアンブルとして「カントリーコード」と「システムキャラクタ」の送信を許可します（カントリーコードの「0」を先頭に挿入した、チェックデジットを含めた 9 桁で送信されます）

A-2.15　UPC-E を UPC-A に変換

Name	Type	設定値	初期値	説明
Convert UPC-E to UPC-A	Integer	0	✓	UPC-E から UPC-A への変換を行わない
		1		UPC-E から UPC-A への変換を行う（足りない桁は「0」で埋めます）

A-2.16　UPC-E1 を UPC-A に変換

Name	Type	設定値	初期値	説明
Convert UPC-E1 to UPC-A	Integer	0	✓	UPC-E1 から UPC-A への変換を行わない
		1		UPC-E1 から UPC-A への変換を行う（足りない桁は「0」で埋めます）

A-2.17　EAN-8/JAN-8 を EAN-13/JAN-13 に拡張

Name	Type	設定値	初期値	説明
EAN-8/JAN-8 Extend	Integer	0	✓	EAN-8/JAN-8 から EAN-13/JAN-13 への変換を行わない
		1		EAN-8/JAN-8 から EAN-13/JAN-13 への変換を行う（足りない桁は「0」で埋めます）

A-2.18　UCC クーポン拡張コードの読み取り許可

「5」で始まる UPC-A、「99」で始まる JAN-13/UPC-A/GS1-128 をクーポンコードとして出力することができます。すべてのクーポンタイプをサポートするには、UPC-A/JAN-13/GS1-128 の読み取りを許可する必要があります。

Name	Type	設定値	初期値	説明
UCC Coupon Extended Code	Integer	0	✓	UCC クーポン拡張コードの読み取りを許可しない
		1		UCC クーポン拡張コードの読み取りを許可する

A-2.19　ISSN の読み取り許可

Name	Type	設定値	初期値	説明
ISSN EAN	Integer	0	✓	ISSN の読み取りを許可しない
		1		ISSN の読み取りを許可する

A-3　Code 128

A-3.1　Code 128 の読み取り許可

Name	Type	設定値	初期値	説明
Code 128	Integer	0		Code 128 の読み取りを許可しない
		1	✓	Code 128 の読み取りを許可する

A-3.2　Code 128 の読み取りを許可する文字数

このコマンドの詳細は 111 ページの「A-1 読み取りを許可する文字数について」を参照してください。

Name	Type	設定範囲	初期値
Code 128 Length Parameter 1	Integer	0 - 55	0
Code 128 Length Parameter 2	Integer	0 - 55	0

A-3.3　GS1-128 の読み取り許可

Name	Type	設定値	初期値	説明
GS1-128	Integer	0		GS1-128 の読み取りを許可しない
		1	✓	GS1-128 の読み取りを許可する

A-3.4　ISBT 128 の読み取り許可

Name	Type	設定値	初期値	説明
ISBT 128	Integer	0		ISBT 128 の読み取りを許可しない
		1	✓	ISBT 128 の読み取りを許可する

A-4　Code 39

A-4.1　Code 39 の読み取り許可

Name	Type	設定値	初期値	説明
Code 39	Integer	0		Code 39 の読み取りを許可しない
		1	✓	Code 39 の読み取りを許可する

A-4.2　Trioptic Code 39 の読み取り許可

117 ページの「A-4.8 Code 39 のフルアスキーフォーマットへの変換」が [0] に設定されている必要があります。

Name	Type	設定値	初期値	説明
Trioptic Code 39	Integer	0	✓	Trioptic Code 39 の読み取りを許可しない
		1		Trioptic Code 39 の読み取りを許可する

A-4.3　Code 39 を Code 32 に変換

116 ページの「A-4.1 Code 39 の読み取り許可」が「1」に設定されている必要があります。

Name	Type	設定値	初期値	説明
Convert Code 39 to Code 32	Integer	0	✓	Code 39 を Code32 に変換しない
		1		Code 39 を Code32 に変換する

A-4.4　Code 32 プリフィックスの挿入

116 ページの「A-4.3 Code 39 を Code 32 に変換」が「1」に設定されている必要があります。

Name	Type	設定値	初期値	説明
Code 32 Prefix	Integer	0	✓	Code32 プリフィックスを挿入しない
		1		Code32 プリフィックスとして先頭に「A」を挿入する

A-4.5　Code 39 の読み取りを許可する文字数

このコマンドの詳細は 111 ページの「A-1 読み取りを許可する文字数について」を参照してください。

Name	Type	設定範囲	初期値
Code 39 Length Parameter 1	Integer	0 - 55	2
Code 39 Length Parameter 2	Integer	0 - 55	55

A-4.6　Code 39 のチェックデジットの検査

「1」が設定されている場合、チェックデジットの無いバーコードを読取ることはできません。

Name	Type	設定値	初期値	説明
Code 39 Check Digit Verification	Integer	0	✓	Code 39 のチェックデジットを検査しない
		1		Code 39 のチェックデジットを検査する

A-4.7　Code 39 のチェックデジットの送信許可

117 ページの「A-4.6 Code 39 のチェックデジットの検査」が「1」に設定されている必要があります。

Name	Type	設定値	初期値	説明
Transmit Code 39 Check Digit	Integer	0	✓	検査に合格した Code 39 チェックデジットの送信を許可しない
		1		検査に合格した Code 39 チェックデジットの送信を許可する

A-4.8　Code 39 のフルアスキーフォーマットへの変換

フルアスキーフォーマットは、2 つの文字の組み合わせでアスキーキャラクタを表現する特別なフォーマットです。

Name	Type	設定値	初期値	説明
Code 39 Full ASCII Conversion	Integer	0	✓	Code 39 フルスキーフォーマットに変換しない
		1		Code 39 フルスキーフォーマットに変換する

A-5　Code 93

A-5.1　Code 93 の読み取り許可

Name	Type	設定値	初期値	説明
Code 93	Integer	0	✓	Code 93 の読み取りを許可しない
		1		Code 93 の読み取りを許可する

A-5.2　Code 93 の読み取りを許可する文字数

このコマンドの詳細は 111 ページの「A-1 読み取りを許可する文字数について」を参照してください。

Name	Type	設定範囲	初期値
Code 93 Length Parameter 1	Integer	0 - 55	4
Code 93 Length Parameter 2	Integer	0 - 55	55

A-6　Code 11

A-6.1　Code 11 の読み取り許可

Name	Type	設定値	初期値	説明
Code 11	Integer	0	✓	Code 11 の読み取りを許可しない
		1		Code 11 の読み取りを許可する

A-6.2　Code 11 の読み取りを許可する文字数

このコマンドの詳細は 111 ページの「A-1 読み取りを許可する文字数について」を参照してください。

Name	Type	設定範囲	初期値
Code 11 Length Parameter 1	Integer	0 - 55	4
Code 11 Length Parameter 2	Integer	0 - 55	55

A-6.3　Code 11 のチェックデジットの検査

「1」または「2」が設定されている場合、チェックデジットの無いバーコードを読取ることはできません。

Name	Type	設定値	初期値	説明
Code 11 Check Digit Verification	Integer	0	✓	Code 11 のチェックデジットを検査しない
		1		Code 11 のチェックデジット (1 文字) を検査する
		2		Code 11 のチェックデジット (2 文字) を検査する

A-6.4　Code 11 のチェックデジットの送信許可

118 ページの「A-6.3 Code 11 のチェックデジットの検査」が「1」または「2」に設定されている必要があります。

Name	Type	設定値	初期値	説明
Transmit Code 11 Check Digit	Integer	0	✓	検査に合格した Code 11 チェックデジットの送信を許可しない
		1		検査に合格した Code 11 チェックデジットの送信を許可する

A-7　Interleaved 2 of 5（ITF）

A-7.1　interleaved 2 of 5 の読み取り許可

Name	Type	設定値	初期値	説明
Interleaved 2 of 5	Integer	0	✓	Interleaved 2 of 5 の読み取りを許可しない
		1		Interleaved 2 of 5 の読み取りを許可する

A-7.2　Interleaved 2 of 5 の読み取りを許可する文字数

このコマンドの詳細は 111 ページの「A-1 読み取りを許可する文字数について」を参照してください。

✎ 読み取りを許可する文字列を制限しない場合、Interleaved 2 of 5 は誤読に対して非常に脆弱になります。

Name	Type	設定範囲	初期値
Interleaved 2 of 5 Length Parameter 1	Integer	0 - 55	14
Interleaved 2 of 5 Length Parameter 2	Integer	0 - 55	0

A-7.3　Interleaved 2 of 5 のチェックデジットの検査

「1」または「2」が設定されている場合、チェックデジットの無いバーコードを読取ることはできません。

Name	Type	設定値	初期値	説明
Interleaved 2 of 5 Check Digit Verification	Integer	0	✓	Interleaved 2 of 5 のチェックデジットを検査しない
		1		Interleaved 2 of 5 のチェックデジット (USS) を検査する
		2		Interleaved 2 of 5 のチェックデジット (OPCC) を検査する

A-7.4　Interleaved 2 of 5 のチェックデジットの送信許可

119 ページの「A-7.3 Interleaved 2 of 5 のチェックデジットの検査」が「1」または「2」に設定されている必要があります。

Name	Type	設定値	初期値	説明
Transmit Interleaved 2 of 5 Check Digit	Integer	0	✓	検査に合格した Interleaved 2 of 5 チェックデジットの送信を許可しない
		1		検査に合格した Interleaved 2 of 5 チェックデジットの送信を許可する

A-7.5　Interleaved 2 of 5 を EAN/JAN-13 に変換

14 文字で構成された Interleaved 2 of 5 を EAN/JAN-13 に変換して出力します。変換を正しく行うためには、データの先頭に「0」と、データの末尾に EAN/JAN-13 用の正しいチェックデジットが必要です。

Name	Type	設定値	初期値	説明
Convert I 2 of 5 to EAN-13	Integer	0	✓	Interleaved 2 of 5 から EAN/JAN-13 への変換を行わない
		1		Interleaved 2 of 5 から EAN/JAN-13 への変換を行う

A-8　Discrete 2 of 5

A-8.1　Discrete 2 of 5 の読み取り許可

Name	Type	設定値	初期値	説明
Discrete 2 of 5	Integer	0	✓	Discrete 2 of 5 の読み取りを許可しない
		1		Discrete 2 of 5 の読み取りを許可する

A-8.2　Discrete 2 of 5 の読み取りを許可する文字数

このコマンドの詳細は 111 ページの「A-1 読み取りを許可する文字数について」を参照してください。

✎ 読み取りを許可する文字列を制限しない場合、Discrete 2 of 5 は誤読に対して非常に脆弱になります。

Name	Type	設定範囲	初期値
Discrete 2 of 5 Length Parameter 1	Integer	0 - 55	12
Discrete 2 of 5 Length Parameter 2	Integer	0 - 55	0

A-9　Codabar (NW-7)

A-9.1　Codabar の読み取り許可

Name	Type	設定値	初期値	説明
Codabar	Integer	0	✓	Codabar の読み取りを許可しない
		1		Codabar の読み取りを許可する

A-9.2　Codabar の読み取りを許可する文字数

このコマンドの詳細は 111 ページの「A-1 読み取りを許可する文字数について」を参照してください。

Name	Type	設定範囲	初期値
Codabar Length Parameter 1	Integer	0 - 55	5
Codabar Length Parameter 2	Integer	0 - 55	55

A-9.3　Codabar の CLSI 編集

[1] が設定されている場合、スタート・ストップキャラクタを取り除き、1 文字目、5 文字目、10 文字目の後にスペースを挿入します。

- このパラメータは、読み取り対象の Codabar が 14 文字で構成されている場合にのみ適用されます。
- スタート・ストップキャラクタは 14 文字には含まれません。

Name	Type	設定値	初期値	説明
CLSI Editing	Integer	0	✓	CLSI 編集を行わない
		1		CLSI 編集を行う

A-9.4　Codabar の NOTIS 編集

[1] が設定されている場合、スタート・ストップキャラクタを取り除いて送信します。

Name	Type	設定値	初期値	説明
NOTIS Editing	Integer	0	✓	NOTIS 編集を行わない
		1		NOTIS 編集を行う

A-10　MSI

A-10.1　MSI の読み取り許可

Name	Type	設定値	初期値	説明
MSI	Integer	0	✓	MSI の読み取りを許可しない
		1		MSI の読み取りを許可する

A-10.2　MSI の読み取りを許可する文字数

このコマンドの詳細は 111 ページの「A-1 読み取りを許可する文字数について」を参照してください。

Name	Type	設定範囲	初期値
MSI Length Parameter 1	Integer	0 - 55	4
MSI Length Parameter 2	Integer	0 - 55	55

A-10.3　MSI のチェックデジットの検査

Name	Type	設定値	初期値	説明
MSI Check Digit	Integer	0	✓	MSI のチェックデジット (1 文字) を検査する
		1		MSI のチェックデジット (2 文字) を検査する

A-10.4　MSI のチェックデジットの送信許可

Name	Type	設定値	初期値	説明
Transmit MSI Check Digit	Integer	0	✓	検査に合格した MSI チェックデジットの送信を許可しない
		1		検査に合格した MSI チェックデジットの送信を許可する

A-10.5　MSI チェックデジットの検証アルゴリズム

121 ページの「A-10.3 MSI のチェックデジットの検査」に [1] が設定されている場合の、2 番目のチェックデジットの検証アルゴリズムを設定します。

Name	Type	設定値	初期値	説明
MSI Check Digit Algorithm	Integer	0		MOD 10/MOD 11
		1	✓	MOD 10/MOD 10

A-11　Chinese 2 of 5

A-11.1　Chinese 2 of 5 の読み取り許可

Name	Type	設定値	初期値	説明
Chinese 2 of 5	Integer	0	✓	Chinese 2 of 5 の読み取りを許可しない
		1		Chinese 2 of 5 の読み取りを許可する

A-12　Matrix 2 of 5

A-12.1　Matrix 2 of 5 の読み取り許可

Name	Type	設定値	初期値	説明
Matrix 2 of 5	Integer	0	✓	Matrix 2 of 5 の読み取りを許可しない
		1		Matrix 2 of 5 の読み取りを許可する

A-12.2　Matrix 2 of 5 の読み取りを許可する文字数

このコマンドの詳細は 111 ページの「A-1 読み取りを許可する文字数について」を参照してください。

Name	Type	設定範囲	初期値
Matrix 2 of 5 Length Parameter 1	Integer	0 - 55	14
Matrix 2 of 5 Length Parameter 2	Integer	0 - 55	0

A-12.3　Matrix 2 of 5 のチェックデジットの検査

「1」が設定されている場合、チェックデジットの無いバーコードを読取ることはできません。

Name	Type	設定値	初期値	説明
Matrix 2 of 5 Check Digit	Integer	0	✓	Matrix 2 of 5 のチェックデジットを検査しない
		1		Matrix 2 of 5 のチェックデジットを検査する

A-12.4　Matrix 2 of 5 のチェックデジットの送信許可

122 ページの「A-12.3 Matrix 2 of 5 のチェックデジットの検査」が「1」または「2」に設定されている必要があります。

Name	Type	設定値	初期値	説明
Transmit Matrix 2 of 5 Check Digit	Integer	0	✓	検査に合格した Matrix 2 of 5 チェックデジットの送信を許可しない
		1		検査に合格した Matrix 2 of 5 チェックデジットの送信を許可する

A-13　反転した 1 次元シンボル

A-13.1　反転した 1 次元シンボルの読み取り

Name	Type	設定値	初期値	説明
Inverse 1D	Integer	0	✓	反転した 1 次元シンボルの読み取りを許可しない（標準のみ）
		1		反転した 1 次元シンボルのみ読み取りを許可する（反転のみ）
		2		自動検出する（標準および反転）

A-14　郵便コード

A-14.1　US Postnet の読み取り許可

Name	Type	設定値	初期値	説明
US Postnet	Integer	0	✓	US Postnet の読み取りを許可しない
		1		US Postnet の読み取りを許可する

A-14.2　US Planet の読み取り許可

Name	Type	設定値	初期値	説明
US Planet	Integer	0	✓	US Planet の読み取りを許可しない
		1		US Planet の読み取りを許可する

A-14.3　US Postal のチェックデジットの送信許可

✎ US Postnet および US Planet の両方に適用されます。

Name	Type	設定値	初期値	説明
Transmit US Postal Check Digit	Integer	0		検査に合格した US Postal チェックデジットの送信を許可しない
		1	✓	検査に合格した US Postal チェックデジットの送信を許可する

A-14.4　UK Postal の読み取り許可

Name	Type	設定値	初期値	説明
UK Postal	Integer	0	✓	UK Postal の読み取りを許可しない
		1		UK Postal の読み取りを許可する

A-14.5　UK Postal のチェックデジットの送信許可

Name	Type	設定値	初期値	説明
Transmit UK Postal Check Digit	Integer	0		検査に合格した UK Postal チェックデジットの送信を許可しない
		1	✓	検査に合格した UK Postal チェックデジットの送信を許可する

A-14.6　日本郵便カスタマバーコードの読み取り許可

Name	Type	設定値	初期値	説明
Japan Postal	Integer	0	✓	日本郵便カスタマバーコードの読み取りを許可しない
		1		日本郵便カスタマバーコードの読み取りを許可する

A-14.7　Australia Post の読み取り許可

Name	Type	設定値	初期値	説明
Australia Post	Integer	0	✓	Australia Post の読み取りを許可しない
		1		Australia Post の読み取りを許可する

A-14.8　Netherlands KIX Code の読み取り許可

Name	Type	設定値	初期値	説明
Netherlands KIX Code	Integer	0	✓	Netherlands KIX Code の読み取りを許可しない
		1		Netherlands KIX Code の読み取りを許可する

A-14.9　USPS 4CB/One Code/Intelligent Mail の読み取り許可

Name	Type	設定値	初期値	説明
USPS 4CB/One Code/Intelligent Mail	Integer	0	✓	USPS 4CB/One Code/Intelligent Mail の読み取りを許可しない
		1		USPS 4CB/One Code/Intelligent Mail の読み取りを許可する

A-14.10　UPU FICS Postal の読み取り許可

Name	Type	設定値	初期値	説明
UPU FICS Postal	Integer	0	✓	UPU FICS Postal の読み取りを許可しない
		1		UPU FICS Postal の読み取りを許可する

A-15　GS1 Databar

A-15.1　GS1 Databar の読み取り許可

このコマンドは次のすべての種類に適用されます。

- GS1 DataBar Omnidirectional
- GS1 DataBar Truncated
- GS1 DataBar Stacked
- GS1 DataBar Stacked Omnidirectional

Name	Type	設定値	初期値	説明
GS1 DataBar	Integer	0		GS1 DataBar の読み取りを許可しない
		1	✓	GS1 DataBar の読み取りを許可する

A-15.2　GS1 Databar Limited の読み取り許可

Name	Type	設定値	初期値	説明
GS1 DataBar Limited	Integer	0	✓	GS1 DataBar Limited の読み取りを許可しない
		1		GS1 DataBar Limited の読み取りを許可する

A-15.3　GS1 Databar Expanded の読み取り許可

Name	Type	設定値	初期値	説明
GS1 DataBar Expanded	Integer	0		GS1 DataBar Expanded の読み取りを許可しない
		1	✓	GS1 DataBar Expanded の読み取りを許可する

A-15.4　GS1 Databar を UPC/EAN/JAN に変換

[1] が設定されている場合、「0」から開始される GS1 Databar または GS1 Databar Limited をスキャンしたとき、先頭の「010」を削除して 13 桁の EAN/JAN-13 として送信します。2 個以上 6 個未満の「0」から開始される GS1 Databar または GS1 Databar Limited をスキャンしたとき、「0100」を削除して 12 桁の UPC-A として送信します。

Name	Type	設定値	初期値	説明
Convert GS1 DataBar to UPC/EAN	Integer	0	✓	GS1 Databar から UPC/EAN/JAN への変換を行わない
		1		GS1 Databar から UPC/EAN/JAN への変換を行う

A-16　合成シンボル

A-16.1　合成シンボル CC-C の読み取り許可

Name	Type	設定値	初期値	説明
Composite CC-C	Integer	0	✓	合成シンボル CC-C の読み取りを許可しない
		1		合成シンボル CC-C の読み取りを許可する

A-16.2　合成シンボル CC-A/B の読み取り許可

✎ このパラメータに [1] が選択されているとき、125 ページの「A-16.4 UPC 合成モード」の値も適切に設定する必要があります。そうしないと、JAN コードが読み取れなくなるかもしれません。

Name	Type	設定値	初期値	説明
Composite CC-A/B	Integer	0	✓	合成シンボル CC-A/B の読み取りを許可しない
		1		合成シンボル CC-A/B の読み取りを許可する

A-16.3　合成シンボル TLC-39 の読み取り許可

Name	Type	設定値	初期値	説明
Composite TLC-39	Integer	0	✓	合成シンボル TLC-39 の読み取りを許可しない
		1		合成シンボル TLC-39 の読み取りを許可する

A-16.4　UPC 合成モード

✎ [1] が選択されているとき、単独の UPC/EAN/JAN コードが読取れない場合があります。

Name	Type	設定値	初期値	説明
UPC Composite Mode	Integer	0		UPC/EAN/JAN コードを常に単独のコードとして検出する
		1	✓	UPC/EAN/JAN コードを常に合成シンボルの一部として検出する
		2		単独の UPC/EAN/JAN コードか合成シンボルか自動検出する

A-17　PDF417

A-17.1　PDF417 の読み取り許可

Name	Type	設定値	初期値	説明
PDF417	Integer	0		PDF417 読み取りを許可しない
		1	✓	PDF417 の読み取りを許可する

A-17.2　マイクロ PDF417 の読み取り許可

Name	Type	設定値	初期値	説明
MicroPDF417	Integer	0	✓	マイクロ PDF417 読み取りを許可しない
		1		マイクロ PDF417 の読み取りを許可する

A-17.3　マイクロ PDF417 の Code 128 エミュレーション

[1] が設定されているとき、特定の値から始まるマイクロ PDF417 を Code 128 として送信します。このパラメータを適用するにはを使用するためには 128 ページの「A-22.4 コード ID の送信」で AIM コード ID を送信するように設定する必要があります。

以下の表は、この機能の設定値と特定のマイクロ PDF417 を読み取った時の AIM コード ID の確認表です。

先頭 3 桁	AIM コード ID	
	無効	有効
903-905	]L3	]C1
908-909	]L4	]C2
910-911	]L5	]C0

✎ リンクされたマイクロ PDF に使用されるコードワード 906、907、912、914 および 915 はサポートしていません。125 ページの「A-16 合成シンボル」をご利用ください。

Name	Type	設定値	初期値	説明
Code 128 Emulation	Integer	0	✓	マイクロ PDF417 の Code 128 エミュレーションを無効にする
		1		マイクロ PDF417 の Code 128 エミュレーションを有効にする

A-18　Data Matrix

A-18.1　Data Matrix の読み取り許可

Name	Type	設定値	初期値	説明
Data Matrix	Integer	0		Data Matrix 読み取りを許可しない
		1	✓	Data Matrix の読み取りを許可する

A-18.2　反転した Data Matrix の読み取り

Name	Type	設定値	初期値	説明
Data Matrix Inverse	Integer	0	✓	反転した Data Matrix の読み取りを許可しない（標準のみ）
		1		反転した Data Matrix のみ読み取りを許可する（反転のみ）
		2		自動検出する（標準および反転）

A-18.3　Maxicode の読み取り許可

Name	Type	設定値	初期値	説明

Maxicode	Integer	0	✓	Maxicode 読み取りを許可しない
		1		Maxicode の読み取りを許可する

A-19　QR コード

A-19.1　QR コードの読み取り許可

Name	Type	設定値	初期値	説明
QR Code	Integer	0		QR コード読み取りを許可しない
		1	✓	QR コードの読み取りを許可する

A-19.2　マイクロ QR コードの読み取り許可

Name	Type	設定値	初期値	説明
MicroQR	Integer	0		マイクロ QR コード読み取りを許可しない
		1	✓	マイクロ QR コードの読み取りを許可する

A-20　Aztec

A-20.1　Aztec コードの読み取り許可

Name	Type	設定値	初期値	説明
Aztec	Integer	0		Aztec コード読み取りを許可しない
		1	✓	Aztec コードの読み取りを許可する

A-20.2　反転した Aztec コードの読み取り

Name	Type	設定値	初期値	説明
Aztec Inverse	Integer	0		反転した Aztec コードの読み取りを許可しない（標準のみ）
		1		反転した Aztec コードのみ読み取りを許可する（反転のみ）
		2	✓	自動検出する（標準および反転）

A-21　Han Xin

A-21.1　Han Xin の読み取り許可

Name	Type	設定値	初期値	説明
Han Xin	Integer	0	✓	Han Xin 読み取りを許可しない
		1		Han Xin の読み取りを許可する

A-21.2　反転した Han Xin の読み取り

Name	Type	設定値	初期値	説明
Han Xin Inverse	Integer	0	✓	反転した Han Xin の読み取りを許可しない（標準のみ）
		1		反転した Han Xin のみ読み取りを許可する（反転のみ）
		2		自動検出する（標準および反転）

A-22　その他

A-22.1　ピックリストモード

[2] が設定されているとき、照準パターンが重なったバーコードのみ読取りを行います。

Name	Type	設定値	初期値	説明
Picklist Mode	Integer	0	✓	ピックリストモードは常に無効です
		2		ピックリストモードは常に有効です

A-22.2　デコードセッションタイムアウト

読取りセッションを継続する時間の最大値を 0.1 秒単位で変更します。

Name	Type	設定範囲	初期値
Decode Session Timeout	Integer	05 - 99 (0.5 秒 - 9.9 秒)	99 (9.9 秒)

A-22.3　低照度環境でのバーコード検出支援

薄暗い環境下におけるプレゼンテーション動作中のバーコードの検出支援を行います。

Name	Type	設定値	初期値	説明
Low Light Motion Detection Assist	Integer	0	✓	低照度環境でのバーコード検出支援を利用しない
		1		低照度環境でのバーコード検出支援のため、常に照準パターンを照射し続ける
		2		低照度環境でのバーコード検出支援のため、常に照明を点灯し続ける

A-22.4　コード ID の送信

[1] または [2] が設定されているとき、あらかじめ定められているコード ID を、Preffix とスキャンしたバーコードデータの間に挿入します。

Name	Type	設定値	初期値	説明
Transmit Code ID Character	Integer	0	✓	コード ID を挿入しない
		1		AIM コード ID を挿入する
		2		シンボルコード ID を挿入する

付録 B　シンボルコード ID

■　コード ID 一覧

コード ID	コードタイプ
A	UPC/EAN/JAN
B	Code 39、Code32
C	Codabar（NW-7）
D	Code 128、ISBT 128
E	Code 93
F	Interleaved 2 of 5（ITF）
G	Discrete 2 of 5、IATA 2 of 5
H	Code 11
J	MSI
K	GS1-128
L	Bookland EAN
M	Trioptic Code 39
N	Coupon Code
R	GS1 Databar 系
S	Matrix 2 of 5
T	UCC コンポジット、TLC 39
U	Chinese 2 of 5
V	Korean 3 of 5
X	ISSN、PDF417、Micro PDF417
z	Aztec、Aztec Rune
P00	Data Matrix
P01	QR コード、Micro QR コード
P02	Maxicode
P03	US Postnet
P04	US Planet
P05	日本郵便カスタマバーコード
P06	UK Postal
P08	Netherlands KIX Code
P09	Australia Post
P0A	USPS 4CB/One Code/Intelligent Mail
P0B	UPU FICS Postal
P0H	Han Xin
P0X	Signature Capture

付録 C　AIM コード ID

AIM コード ID は 3 つの識別子（]cm ）を含んで表示されます。

-]　= フラグ文字
- c　= コード文字
- m　= 修飾文字

■　AIM コード文字一覧

コード文字	コードタイプ
A	Code 39、Code 39 フル ASCII、Code32
C	Code 128、ISBT 128、GS1-128、Coupon
d	Data Matrix
E	UPC/EAN/JAN
e	GS1 Databar 系
F	Codabar（NW-7）
G	Code 93
H	Code 11
h	Han Xin
I	Interleaved 2 of 5（ITF）
L	PDF417、Micro PDF417
L2	TLC 39
M	MSI
Q	QR コード、Micro QR コード
S	Discrete 2 of 5、IATA 2 of 5
U	Maxicode
z	Aztec、Aztec Rune
X	Code 39 Trioptic、Bookland EAN、Matrix 2 of 5、Chinese 2 of 5、Korean 3 of 5、ISSN、日本郵便カスタマバーコード、US Postnet、US Planet、UK Postal、Australia Post、Netherlands KIX Code、USPS 4CB/One Code/Intelligent Mail、UPU FICS Postal、Signature Capture

■　AIM 修飾文字一覧

修飾文字はオプション値の合計です。

オプション値	オプション
Code 39	
0	チェックデジットも、フル ASCII 変換もなし
1	チェックデジットを検査した
3	チェックデジットを検査し送信しなかった
4	フル ASCII 変換を行った

AIM 修飾文字一覧 (続き)

オプション値	オプション
5	フル ASCII 変換を行い、チェックデジットを検査した
7	フル ASCII 変換を行い、チェックデジット検査し送信しなかった
例　フル ASCII バーコードのチェックデジットを検査し送信しなかった場合の AIM コード ID は、]A7 となります。	
Trioptic Code 39	
0	オプションが設定されておらず、常に 0 を送信します
例　Trioptic Code 39 のコード ID は、]X0 となります。	
Code 128	
0	FNC1 コードが最初の文字の位置に無い
1	FNC1 コードが最初の文字の位置にある
2	FNC1 コードが 2 番目の文字の位置にある
例　Code128 または EAN128 バーコードの最初の文字の位置に FNC1 コードがある場合の AIM コード ID は、]C1 となります。	
Interleaved 2 of 5（ITF）	
0	チェックデジットを検査していない
1	チェックデジットを検査した
3	チェックデジットを検査し送信しなかった
例　Interleaved 2 of 5 のチェックデジットを検査しない場合の AIM コード ID は、]I0 となります。	
Codabar（NW-7）	
0	チェックデジットを検査していない
1	チェックデジットを検査した
例　Codabar のチェックデジットを検査しない場合の AIM コード ID は、]F0 となります。	
Code 93	
0	オプションが設定されておらず、常に 0 を送信します
例　Code93 バーコードのコード ID は、]G0 となります。	
MSI	
0	チェックデジットが送信されます
1	チェックデジットは送信されません
例　MSI のコード ID は、]M0 となります。	
Discrete 2 of 5	
0	オプションが設定されておらず、常に 0 を送信します
例　Discrete 2 of 5 のコード ID は、]S0 となります。	
UPC/EAN/JAN	
0	アドオンコードを含まない UPC-A、UPC-E および JAN-13
1	2 桁アドオンコードのみ

AIM 修飾文字一覧 (続き)

オプション値	オプション
2	5 桁アドオンコードのみ
3	アドオンコードを含む UPC-A、UPC-E および JAN-13
4	JAN-8
例　UPC-A のコード ID は、]E0 となります。	
Bookland EAN	
0	オプションが設定されておらず、常に 0 を送信します
例　Bookland EAN のコード ID は、]X0 となります。	
ISSN	
0	オプションが設定されておらず、常に 0 を送信します
例　ISSN のコード ID は、]X0 となります。	
Code11	
0	1 つのチェックデジット
1	2 つのチェックデジット
3	チェックデジットを検査し送信しなかった
例　Code 11 のコード ID は]H0 となります。	
GS1 Databar 系	
0	オプションが設定されておらず、常に 0 を送信します
例　GS1 Databar のコード ID は、]e0 となります。	
合成コード	
	ネイティブモード
0	標準データパケット
1	エンコードされたシンボル区切り文字に続くデータを含むパケット
2	エスケープ文字に続くデータを含むパケット。データパケットは ECI プロトコルをサポートしません。
3	エスケープ文字に続くデータを含むパケット。データパケットは ECI プロトコルをサポートします。
	GS1-128 エミュレーション
1	データパケットは GS1-128 です（先頭に]JC1 がつく）
PDF417、Micro PDF417	
0	リーダーは 1994 年の PDF417 シンボル体系仕様で定義されたプロトコルに準拠するように設定されています。 ✎ このオプションが送信されると、受信側は ECI が呼び出されたかどうか、またはデータバイト 92DEC が送信時に 2 倍になったかどうかを確実に判断できません。
1	リーダーは ECI プロトコルに従うように設定されています。すべてのデータ文字 92DEC は 2 倍になります。

AIM 修飾文字一覧 (続き)

オプション値	オプション
2	リーダーは基本チャネル操作用に設定されています（エスケープ文字伝送プロトコルなし)。 データ文字 92DEC は 2 倍になりません。 ✎ デコーダがこのモードに設定されている場合、バッファされていないマクロシンボルおよびデコーダに ECI エスケープシーケンスの伝達を要求するシンボルは送信できません。
3	バーコードには GS1-128 記号が含まれており、最初のコードワードは 903-907、912、914、915 です。
4	バーコードには GS1-128 記号が含まれており、最初のコードワードは 908-909 です。
5	バーコードには GS1-128 記号が含まれており、最初のコードワードは 910-911 です。
例　PDF417 のコード ID は、]L0 となります。	
Data Matrix	
0	ECC 000 – ECC 140（未サポート）
1	ECC 200
2	最初の位置または 5 番目の位置に FNC1 がある ECC 200
3	2 番目の位置または 6 番目の位置に FNC1 がある ECC 200
4	ECI プロトコルを実装した ECC 200
5	ECI プロトコルを実装しており、最初に位置または 5 番目の位置に FNC1 がある ECC 200
6	ECI プロトコルを実装しており、2 番目の位置または 6 番目の位置に FNC1 がある ECC 200
Maxicode	
0	モード 4 またはモード 5
1	モード 2 またはモード 3
2	ECI プロトコルを実装したモード 4 またはモード 5
3	セカンダリメッセージに ECI プロトコルを実装したモード 2 またはモード 3
QR Code	
0	モデル 1
1	ECI プロトコルを実装していないモデル 2、または MicroQR コード
2	ECI プロトコルを実装したモデル 2
3	ECI プロトコルを実装しておらず、最初の位置に FNC1 があるモデル 2
4	ECI プロトコルを実装しており、最初の位置に FNC1 があるモデル 2
5	ECI プロトコルを実装しておらず、2 番目の位置に FNC1 があるモデル 2
6	ECI プロトコルを実装しており、2 番目の位置に FNC1 があるモデル 2
Aztec	
0	Aztec
C	Aztec Runes
HAN XIN	
0	一般的なデータ。特別な機能は設定されていません。 送信データは AIM ECI プロトコルに準拠していません。

AIM 修飾文字一覧 (続き)

オプション値	オプション
1	ECI プロトコルが有効。エンコードされた ECI モードが少なくとも 1 つあります。送信データは AIM ECI プロトコルに従う必要があります。

付録 D　コードタイプ SSI ID

■　SSI ID 一覧

SSI ID	コードタイプ
0x01	Code 39
0x02	Codabar
0x03	Code 128
0x04	D25
0x05	IATA
0x06	ITF
0x07	Code 93
0x08	UPCA
0x09	UPCE 3
0x0A	EAN-8
0x0B	EAN-13
0x0C	Code 11
0x0D	Code 49
0x0E	MSI
0x0F	GS1-128
0x10	UPCE1
0x11	PDF-417
0x12	Code 16K
0x13	Code 39 Full ASCII
0x14	UPCD
0x15	Trioptic
0x16	Bookland
0x17	Coupon Code
0x18	NW7
0x19	ISBT-128
0x1A	Micro PDF
0x1B	Data Matrix
0x1C	QR Code
0x1d	Micro PDF CCA
0x1E	Postnet (US)
0x1F	Planet (US)
0x20	Code 32
0x21	ISBT-128 Concat.

SSI ID 一覧 (続き)

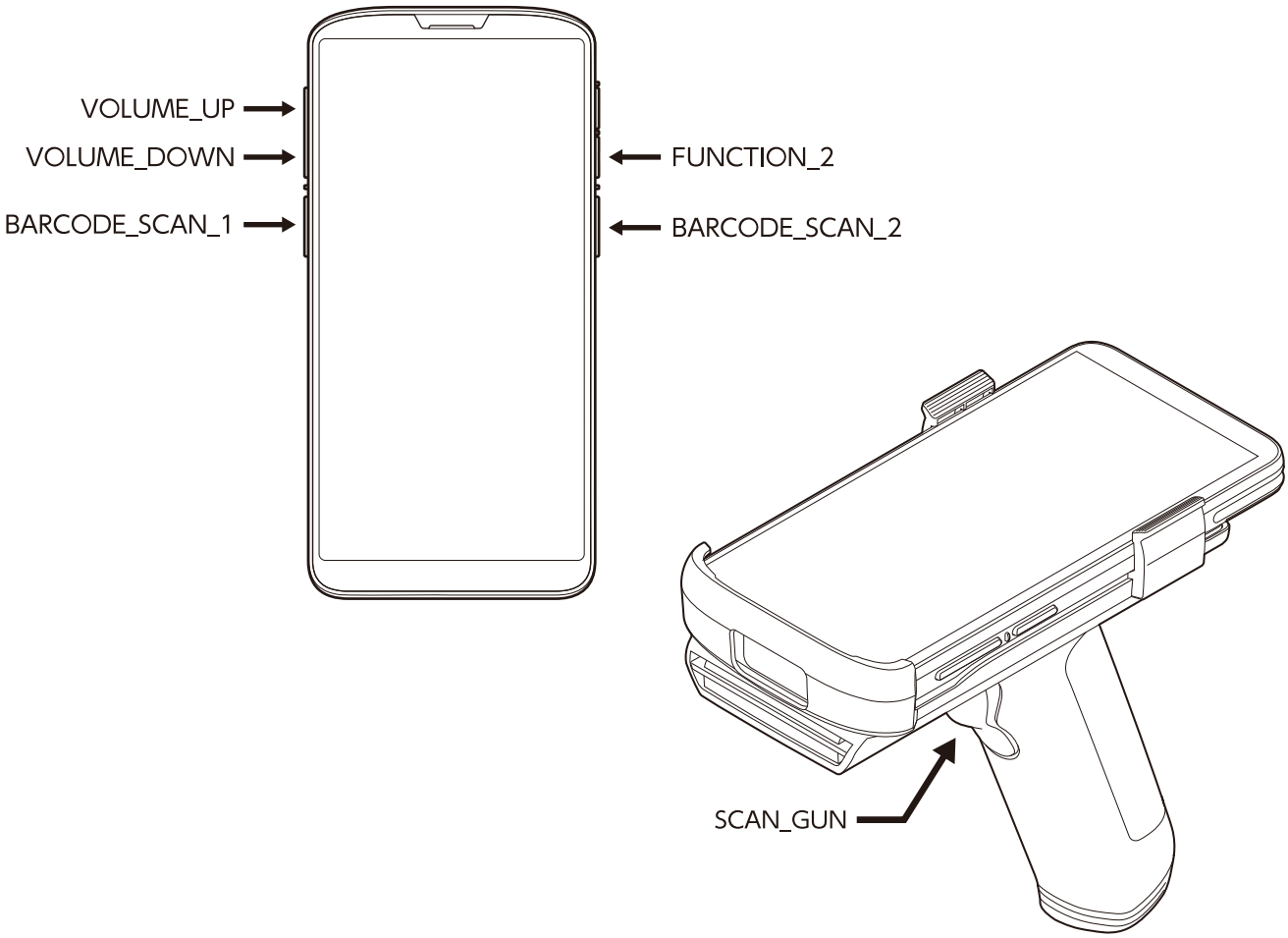
SSI ID	コードタイプ
0x22	Postal (Japan)
0x23	Postal (Australia)
0x24	Postal (Dutch)
0x25	Maxicode
0x26	Postbar (CA)
0x27	Postal (UK)
0x28	Macro PDF-417
0x29	Macro QR Code
0x2C	Micro QR Code
0x2D	Aztec Code
0x2E	Aztec Rune Code
0x2F	French Lottery
0x30	GS1 DataBar-14
0x31	GS1 DataBar Limited
0x32	GS1 DataBar Expanded
0x33	Parameter (FNC3)
0x34	4State US
0x35	4State US4
0x37	Scanlet Webcode
0x38	Cue CAT Code
0x48	UPCA + 2
0x49	UPCE + 2
0x4A	EAN-8 + 2
0x4B	EAN-13 + 2
0x50	UPCE1 + 2
0x51	Composite(CC-A + GS1-128)
0x52	Composite(CC-A + EAN-13)
0x53	Composite(CC-A + EAN-8)
0x54	Composite (CC-A+ GS1 DataBarExpanded)
0x55	Composite (CC-A+ GS1 DataBarLimited)
0x56	Composite (CC-A + GS1DataBar-14)
0x57	Composite(CC-A + UPC-A)
0x58	Composite(CC-A + UPC-E)
0x59	Composite(CC-C + GS1-128)

SSI ID 一覧 (続き)

SSI ID	コードタイプ
0x5A	TLC-39
0x61	Composite(CC-B + GS1-128)
0x62	Composite(CC-B + EAN-13)
0x63	Composite(CC-B + EAN-8)
0x64	Composite (CC-B+ GS1 DataBarExpanded)
0x65	Composite (CC-B+ GS1 DataBarLimited)
0x66	Composite (CC-B + GS1DataBar-14)
0x67	Composite(CC-B + UPC-A)
0x68	Composite(CC-B + UPC-E)
0x69	Signature
0x71	Matrix 2 of 5
0x72	C 2 of 5
0x73	Korean 3 of 5
0x88	UPCA + 5
0x89	UPCE + 5
0x8A	EAN-8 + 5
0x8B	EAN-13 + 5
0x90	UPCE1 + 5
0x99	Multipacket Format
0x9A	Macro Micro PDF
0xA0	OCRB
0xB4	RSS (GS1Databar) ExpandedCoupon
0xB7	Han Xin
0xC1	GS1 Datamatrix
0xE0	RFID Raw
0xE1	RFID URI

付録 E　デバイスのキーマッピング

■　EA630 のキー定義

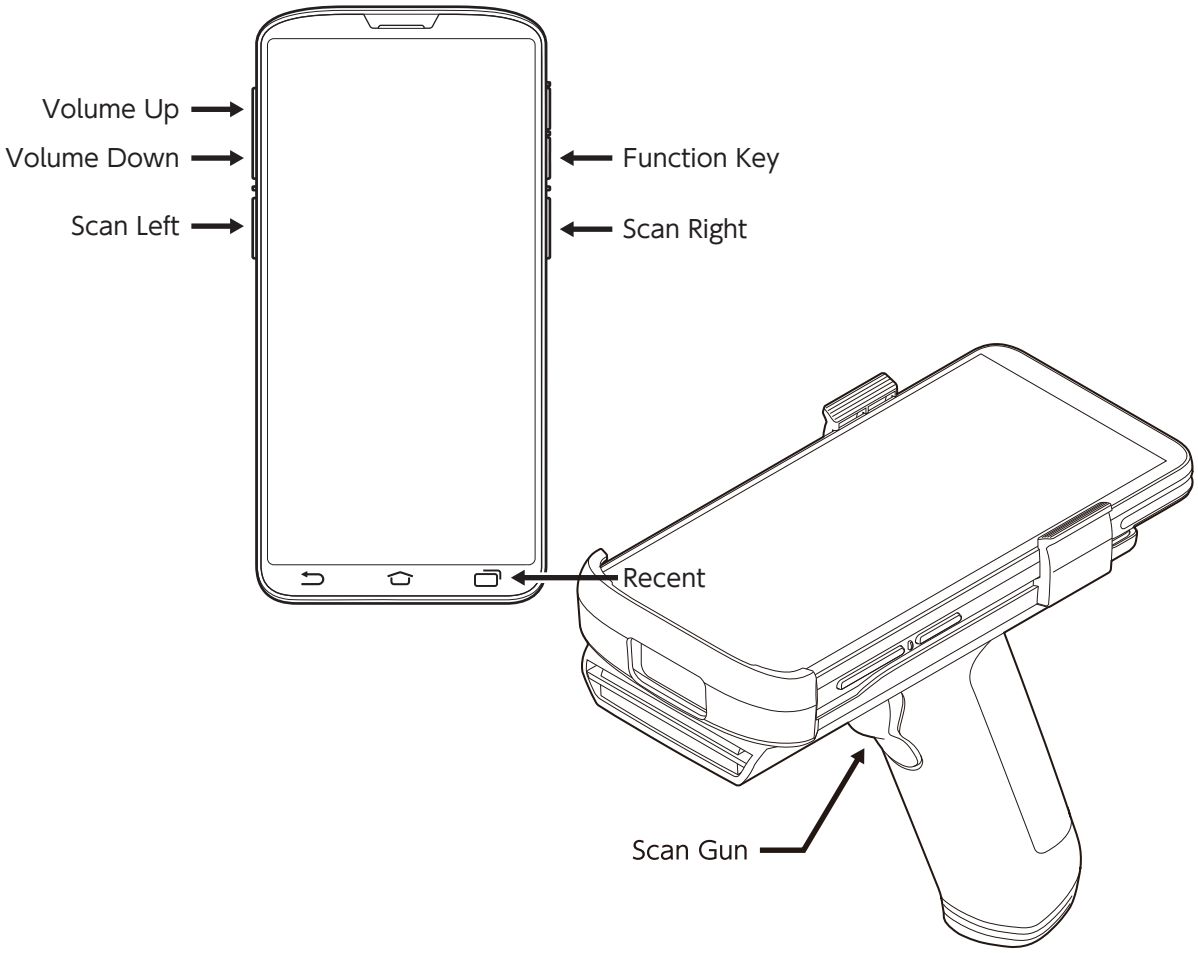


EA630 拡張キーコード

キー	キーコード	説明
SCAN_GUN	289	内部スキャンエンジンの読み取り開始 / 停止
FUNCTION_2	290	ファンクションキー
BARCODE_SCAN_1	291	内部スキャンエンジンの読み取り開始 / 停止
BARCODE_SCAN_2	292	内部スキャンエンジンの読み取り開始 / 停止
SCAN_EXTENDED	293	外部スキャンエンジンの読み取り開始 / 停止 ・2D ロングレンジガングリップ (PN# 5500-900077G) ・RG630 UHF ガングリップ (PN# 5500-900081G)

- 内部スキャンエンジンは、本体に搭載されているバーコードスキャナエンジンです。
- 外部スキャンエンジンは、オプションの 2D ロングレンジガングリップに搭載されているスキャンエンジンまたは、RG630 UHF ガングリップに搭載されている RFID エンジンです。

■　PA760 のキー定義

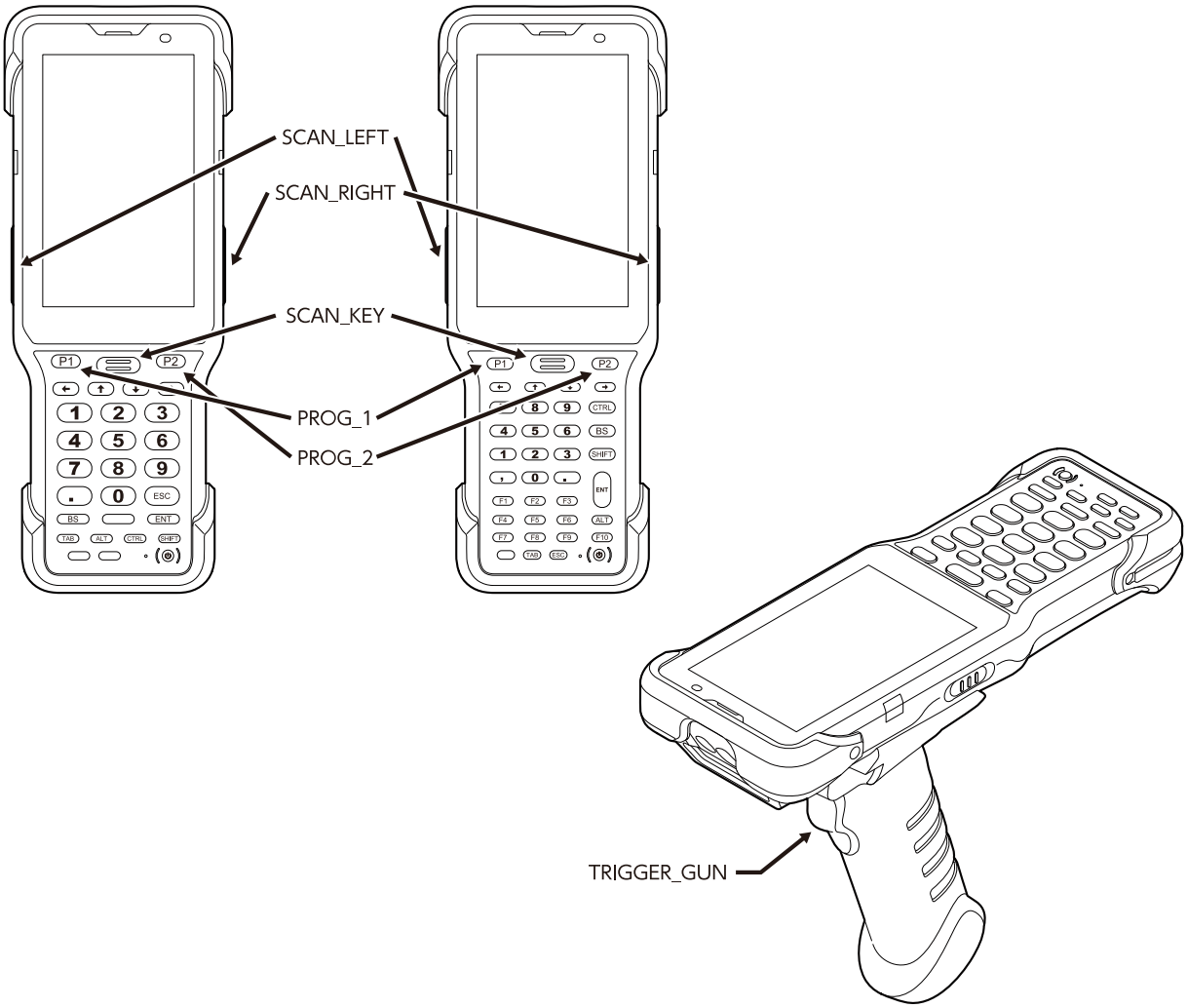


PA760 拡張キーコード

キー	キーコード	説明
SCAN_GUN	289	内部スキャンエンジンの読み取り開始 / 停止
FUNCTION_KEY	290	ファンクションキー
SCAN_LEFT	291	内部スキャンエンジンの読み取り開始 / 停止
SCAN_RIGHT	292	内部スキャンエンジンの読み取り開始 / 停止
SCAN_EXTENDED	293	外部スキャンエンジンの読み取り開始 / 停止 ・2D ロングレンジガングリップ (PN# 5500-900065G) ・RG760 UHF ガングリップ (PN# 5500-900063G)

- 内部スキャンエンジンは、本体に搭載されているバーコードスキャナエンジンです。
- 外部スキャンエンジンは、オプションの 2D ロングレンジガングリップに搭載されているスキャンエンジンまたは、RG760 UHF ガングリップに搭載されている RFID エンジンです。

■　HT730 のキー定義

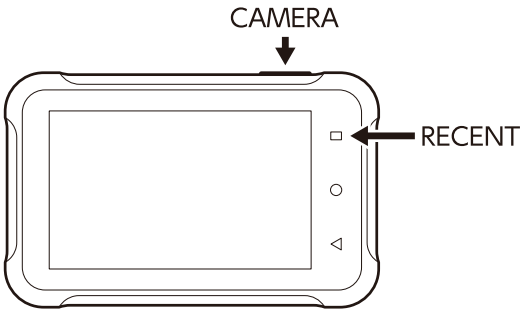


HT730 拡張キーコード

キー	キーコード	説明
F13	289	F13 キー
SCAN_KEY	290	内部スキャンエンジンの読み取り開始 / 停止
SCAN_LEFT	291	内部スキャンエンジンの読み取り開始 / 停止
SCAN_RIGHT	292	内部スキャンエンジンの読み取り開始 / 停止
VKB_TOGGLE	294	仮想キーボードの表示の切り替え
FUNCTION	295	ファンクションキー
PROG_1	296	P1 キー
PROG_2	297	P2 キー
TRIGGER_GUN	298	内部スキャンエンジンの読み取り開始 / 停止
SCAN_EXTENDED	301	外部スキャンエンジンの読み取り開始 / 停止
SCAN_GUN_SWITCH	302	予約

- 内部スキャンエンジンは、本体に搭載されているバーコードスキャナエンジンです。
- 外部スキャンエンジンは、オプションの 2D ロングレンジガングリップに搭載されているスキャンエンジンまたは、RG760 UHF ガングリップに搭載されている RFID エンジンです。

■　WD200 のキー定義



■　Android 標準キーコード

https://developer.android.com/reference/android/view/KeyEvent

unitech

because we care

ユニテック・ジャパン株式会社

〒 104-0033 東京都中央区新川 1-5-19 茅場町長岡ビル 8F

https://www.ute.com/jp