

ePOS-Print SDK for iOS

ユーザーズマニュアル

概要

特徴および環境について説明します。

サンプルプログラム

サンプルプログラムの使い方について説明します。

プログラミングガイド

アプリケーション開発のプログラミング方法について説明します。

API リファレンス

ePOS-Print SDK for iOSで提供しているAPIについて説明します。

コマンドの送受信

コマンドを送受信するためのAPIについて説明します。

付録

ePOS-Print SDK for iOSで使用するプリンター仕様について説明します。

ご注意

- 本書の内容の一部または全部を無断で転載、複写、複製、改ざんすることは固くお断りします。
- 本書の内容については、予告なしに変更することがあります。最新の情報はお問い合わせください。
- 本書の内容については、万全を期して作成いたしました但、万が一不審な点や誤り、記載もれなど、お気づきの点がありましたらご連絡ください。
- 運用した結果の影響については、上項にかかわらず責任を負いかねますのでご了承ください。
- 本製品がお客様により不適切に使用されたり、本書の内容に従わずに取り扱われたり、またはエプソンおよびエプソン指定の者以外の第三者により修理・変更されたことなどに起因して生じた損害などにつきましては、責任を負いかねますのでご了承ください。
- エプソン純正品およびエプソン品質認定品以外のオプションまたは消耗品を装着してトラブルが発生した場合には、責任を負いかねますのでご了承ください。

商標について

EPSON、EXCEED YOUR VISION および ESC/POS はセイコーエプソン株式会社の登録商標です。

Xcode[®]、iPhone[®]、iPod touch[®]、iPad[®]、iTunes[®] は、米国 Apple, Inc. の米国およびその他の国で登録された商標です。

iOS[®] は、Cisco の米国およびその他の国のライセンスに基づき使用されています。

Wi-Fi[®] は、Wi-Fi Alliance[®] の登録商標です。

Bluetooth[®] のワードマークおよびロゴは、Bluetooth SIG, Inc. が所有する登録商標であり、セイコーエプソン株式会社はこれらのマークをライセンスに基づいて使用しています。

QR コードは株式会社デンソーウェーブの登録商標です。

その他の製品名および会社名は、各社の商標または登録商標です。



ESC/POS[®] コマンドシステム

エプソンは、独自の POS プリンターコマンドシステム、ESC/POS により業界のイニシアティブをとってきました。ESC/POS は特許取得済及び特許出願中のものを含む数多くの独自のコマンドを持ち、高い拡張性で多才な POS システムの構築を実現します。エプソン POS プリンターとディスプレイの全タイプに互換性を持つほか、この独自の制御システムにはフレキシビリティもあるため、将来アップグレードが行いやすくなります。その機能と利便性は世界中で評価されています。

安全のために

記号の意味

本書では以下の記号が使われています。それぞれの記号の意味をよく理解してから製品を取り扱ってください。

	ご使用上、必ずお守りいただきたいことを記載しています。この表示を無視して誤った取り扱いをすると、製品の故障や動作不良の原因になる可能性があります。
	補足説明や知っておいていただきたいことを記載しています。

使用制限

本製品を航空機・列車・船舶・自動車などの運行に直接関わる装置・防災防犯装置・各種安全装置など機能・精度などにおいて高い信頼性・安全性が必要とされる用途に使用される場合は、これらのシステム全体の信頼性および安全維持のためにフェールセーフ設計や冗長設計の措置を講じるなど、システム全体の安全設計にご配慮いただいた上で当社製品をご使用いただくようお願いいたします。

本製品は、航空宇宙機器、幹線通信機器、原子力制御機器、医療機器など、きわめて高い信頼性・安全性が必要とされる用途への使用を意図しておりませんので、これらの用途には本製品の適合性をお客様において十分ご確認の上、ご判断ください。

本書について

本書の目的

本書は、ePOS-Print SDK を使った、印刷システムの構築、設計またはプリンターアプリケーションの開発、設計に必要なすべての情報を開発技術者に提供することを、その目的としています。

本書の構成

本書は次のように構成されています。

第 1 章	概要
第 2 章	サンプルプログラム
第 3 章	プログラミングガイド
第 4 章	API リファレンス
第 5 章	コマンドの送受信
付録	プリンターごとのサポート API 一覧 プリンター別サポート情報 注意事項

もくじ

■ 安全のために	3
記号の意味	3
■ 使用制限	3
■ 本書について	4
本書の目的	4
本書の構成	4
■ もくじ	5

概要

7

■ ePOS-Print SDK の概要	7
特徴	7
機能	8
Bluetooth 通信を行うアプリケーションの開発	8
■ 動作環境	9
iOS バージョン	9
iOS デバイス	9
プリンター	9
開発環境	10
■ 提供物	11
パッケージ	11
マニュアル	11
サンプルプログラム	11
ダウンロード	11
■ 制限事項	12

サンプルプログラム

13

■ 概要	13
■ 使用環境	14
開発環境	14
プリンター	14
ターゲットデバイス	14
■ 環境構築	15
■ 使用方法	16
プリンターを検索して印刷する	16
プリンターの機種名を取得する	23
QR コードを使ってプリンターを選択する	24
QR コードを印刷する	24

プログラミングガイド

25

■ ePOS-Print SDK for iOS の組み込み方法	25
■ ePOS-Print SDK	27
印刷モードについて	27
プログラミングフロー	27
プリンターの選択	28
印刷ドキュメントの作成	30
印刷ドキュメントの送信	32
プリンターの状態を確認してから印刷する	34
■ プリンターステータスを自動で取得	36
イベント一覧	37
■ ステータス	38
エラーステータスと対処方法	38
プリンターステータスと対処方法	40
バッテリーステータス	41

API リファレンス

43

■ ePOS-Print API	43
initWithPrinterModel	46
clearCommandBuffer	47
addTextAlign	48
addTextLineSpace	49
addTextRotate	50
addText	51
addTextLang	52
addTextFont	53
addTextSmooth	54
addTextDouble	55
addTextSize	56
addTextStyle	57
addTextPosition	59
addFeedUnit	60
addFeedLine	61
addImage	62
addImage(旧フォーマット)	65
addImage(旧フォーマット)	68
addLogo	70
addBarcode	71
addSymbol	76
addPageBegin	81
addPageEnd	82
addPageArea	83
addPageDirection	84
addPagePosition	85
addPageLine	86
addPageRectangle	88
addCut	89

addPulse.....	90
addSound.....	91
addSound(旧フォーマット).....	93
addFeedPosition.....	95
addLayout.....	96
addCommand.....	98
init.....	99
openPrinter.....	100
openPrinter(旧フォーマット).....	102
openPrinter(旧フォーマット).....	104
closePrinter.....	106
sendData.....	107
sendData(旧フォーマット).....	109
setStatusChangeEventCallback.....	111
setOnlineEventCallback.....	112
setOfflineEventCallback.....	113
setPowerOffEventCallback.....	114
setCoverOkEventCallback.....	115
setCoverOpenEventCallback.....	116
setPaperOkEventCallback.....	117
setPaperNearEndEventCallback.....	118
setPaperEndEventCallback.....	119
setDrawerClosedEventCallback.....	120
setDrawerOpenEventCallback.....	121
setBatteryLowEventCallback.....	122
setBatteryOkEventCallback.....	123
setBatteryStatusChangeEventCallback.....	124
■ プリンター検索 API.....	125
start.....	125
stop.....	127
getDeviceInfoList.....	128
getResult(旧フォーマット).....	130
■ プリンター簡単選択 API.....	131
parseQR.....	131
createQR.....	132
deviceType.....	133
printerName.....	133
macAddress.....	133
■ ログ設定 API.....	134
setLogSettings.....	134

コマンドの送受信..... 137

■ プログラミング.....	137
プログラミングフロー.....	137
EpsonIo クラスの初期化.....	138
デバイスポートのオープン.....	138
データの送信.....	138
データの受信.....	139
デバイスポートのクローズ.....	139
■ エラー値一覧.....	140

■ コマンド送受信 API リファレンス.....	141
init.....	141
open.....	142
close.....	143
write.....	144
read.....	146

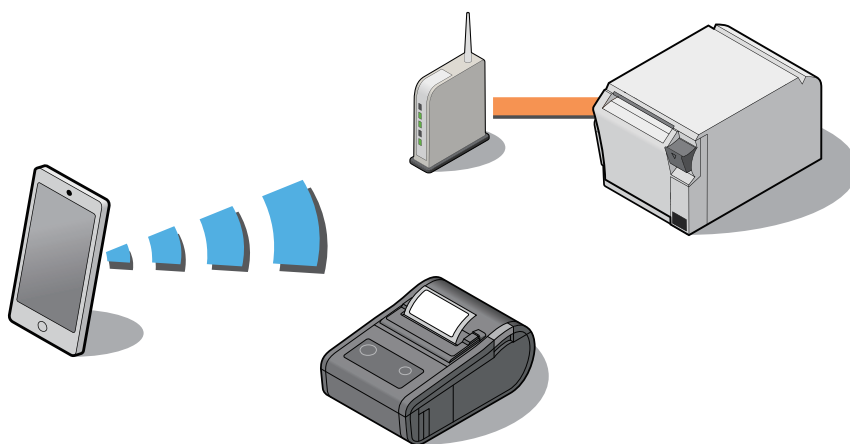
付録..... 147

■ プリンターごとのサポート API 一覧.....	147
■ プリンター別サポート情報.....	149
TM-P20.....	149
TM-P20 iOS Bluetooth モデル.....	151
TM-P60II.....	153
TM-P60II iOS Bluetooth モデル.....	155
TM-T20II iOS Bluetooth モデル.....	157
TM-T70.....	158
TM-T70II.....	159
TM-T70II iOS Bluetooth モデル.....	160
TM-T88V.....	161
TM-T88V iOS Bluetooth モデル.....	162
TM-T90II.....	163
■ 注意事項.....	164
一台のプリンターを複数のモバイル端末から 使用する場合.....	164

概要

本章では、ePOS-Print SDK for iOS の特徴および仕様について説明しています。

ePOS-Print SDK の概要



ePOS-Print SDK for iOS は、エプソン TM プリンターに印刷するための iOS アプリケーションを開発する、開発者向け SDK です。ePOS-Print SDK で提供する API を使用してアプリケーションを開発します。

ePOS-Print SDK には、Android アプリケーション向けの ePOS-Print SDK for Android も用意されています。



TM プリンターにコマンドを送受信するための API も用意されています。
コマンド送受信 API は、ePOS-Print API の EposPrint クラスと同時に使用できません。
コマンド送受信 API の詳細は[コマンドの送受信 \(137 ページ\)](#)を参照してください。

特徴

- ❑ iOS アプリケーションから、TM プリンターに印刷できます。
- ❑ iOS アプリケーションから、TM プリンターのステータスを取得できます。

ePOS-Print API

- 印字の設定（位置揃え / 改行量 / 倒立印字 / ページモード）
- 文字データの設定（言語 / フォント（デバイスフォント）/ 倍角 / 倍率 / スムージング / 印字位置）
- 文字書式の設定（白黒反転 / アンダーライン / 太字）
- 紙送り設定（ドット単位 / 行単位）
- 画像の印字（ラスターイメージ / NV グラフィック）
- バーコードの印字
（機種ごと印字できるバーコードは、[プリンター別サポート情報（149 ページ）](#)を参照してください。）
- 2 次元シンボルの印字
（機種ごと印字できる 2 次元シンボルは、[プリンター別サポート情報（149 ページ）](#)を参照してください。）
- ドロアーキック機能
- ブザー機能
- 用紙レイアウトの設定
- ラベル / ブラックマーク紙の紙送り設定
- ESC/POS コマンドの送信
- プリンターからの応答を取得（印字結果 / プリンターの状態 / バッテリーの状態）

プリンター検索 API

- プリンターの検索

プリンター簡単選択 API

- プリンターの簡単選択
（QR コードを使ってプリンターを簡単選択できます。）

ログ設定 API

- ログ出力の設定
（iOS 端末のストレージや、TCP 接続できるサーバーに出力できます。）



iOS 端末に出力したログは、USB 接続で他のコンピューターにも保存できます。

Bluetooth[®] 通信を行うアプリケーションの開発

Bluetooth を使用するアプリケーションソフトを App Store に登録する場合、エプソンからアップルに事前申請が必要になります。App Store に登録するアプリケーションソフトごとに、以下の URL から申請してください。

<https://c4b.epson-biz.com/ais/J>

動作環境

iOS バージョン

- ❑ iOS Version 4.2 ~ 4.3.5
- ❑ iOS Version 5.0 ~ 5.1.1
- ❑ iOS Version 6.0 ~ 6.1.4
- ❑ iOS Version 7.0 ~ 7.1.2
- ❑ iOS Version 8.0 ~ 8.0.2
- ❑ iOS Version 8.2



最新のバージョンは、README ファイルを参照してください。

iOS デバイス

- ❑ iPhone 3G/ iPhone 3GS/ iPhone 4/ iPhone 4s/ iPhone 5/ iPhone 5s/ iPhone 5c/ iPhone 6/ iPhone 6 Plus
- ❑ iPod touch (2nd generation)/ iPod touch (3rd generation)/ iPod touch (4th generation)/ iPod touch (5th generation)
- ❑ iPad/ iPad 2/ iPad (3rd generation)/ iPad 4/ iPad Air/ iPad Air 2/ iPad mini/ iPad mini 2 (iPad mini with Retina display)/ iPad mini 3

プリンター

TM プリンター	インターフェイス		
	有線 LAN	無線 LAN	Bluetooth
TM-P20	-	○	-
TM-P20 iOS <i>Bluetooth</i> モデル	-	-	○
TM-P60II	-	○	-
TM-P60II iOS <i>Bluetooth</i> モデル	-	-	○
TM-T20II iOS <i>Bluetooth</i> モデル	-	-	○
TM-T70	○	○	-
TM-T70II	○	○	-
TM-T70II iOS <i>Bluetooth</i> モデル	-	-	○
TM-T88V	○	○	-
TM-T88V iOS <i>Bluetooth</i> モデル	-	-	○
TM-T90II	○	○	-



- TM プリンター設定で、Busy となる条件は受信バッファフルのみに設定してください。設定についてはプリンターの詳細取扱説明書を参照してください。
- 無線 LAN の場合、インフラストラクチャモード (Infrastructure mode) とアドホックモード (Ad hoc mode) を使用できます。

開発環境

iOS アプリケーションを開発するには、以下が必要です。

- Xcode Version 4.2 以降

提供物

パッケージ

ファイル名	説明
ePOS-Print.h	クラス定義、エラー値 / デバイスタイプの定数定義を含むヘッダーファイルです。
libeposprint.a	機能実行用ライブラリーです。 (armv6, armv7, armv7s, arm64, i386, x86_64 に対応)
ePOSEasySelect.h	簡単にプリンターを選択するためのヘッダーファイルです。
libeposeasyselect.a	簡単にプリンターを選択するためのライブラリーです。 (armv7, armv7s, arm64, i386, x86_64)
ePOS-Print_Sample_iOS.zip	サンプルプログラムファイルです。
README.jp.txt	README ファイルです。
README.en.txt	README ファイル (英語版) です。
EULA.jp.txt	SOFTWARE LICENSE AGREEMENT が記載されています。
EULA.en.txt	SOFTWARE LICENSE AGREEMENT (英語版) が記載されています。
ePOS-Print_SDK_iOS_ja_revx.pdf	本書です。
ePOS-Print_SDK_iOS_en_revx.pdf	本書 (英語版) です。
ePOS-Print_SDK_iOS_AppDevGuide_ja_revx.pdf	開発環境を構築する手順が記載されています。
ePOS-Print_SDK_iOS_AppDevGuide_en_revx.pdf	開発環境を構築する手順が記載されています。(英語版)

マニュアル

ePOS-Print SDK for iOS のマニュアルには以下のものがあります。

- ePOS-Print SDK for iOS ユーザーズマニュアル (本書)
- ePOS-Print SDK for iOS アプリケーション開発環境 - セットアップガイド

サンプルプログラム

ePOS-Print SDK を使用した、TM プリンター用 iOS アプリケーションのサンプルプログラムがあります。

- ePOS-Print_Sample_iOS.zip
 - 基本機能用サンプル (ePOSPrintSample)
 - 簡単選択用サンプル (ePOSEasySelectSample)

ダウンロード

提供物は、下記エプソン販売ホームページからダウンロードできます。

<http://www.epson.jp/support/sd/>

制限事項

- ePOS-Print APIの通信API(45ページ)とコマンド送受信API(137ページ)は、同一デバイスに対して同時に使用することはできません。
- 同じアプリケーション内で同時にオープンできるデバイスポート数は 16 個です。
- *Bluetooth* 接続でプリンターとの通信中に、端末がスリープ状態になると、通信が切断されてしまいます。
- *Bluetooth* 接続の場合、製品が印刷できないときに印刷データを送ると、iOS の仕様によっては、その印刷データが削除される場合があります。製品が以下のような状態では印刷できません。
 - ロール紙カバーが開いている
 - 用紙なし
 - 印刷済みラベル剥離待ち

サンプルプログラム

本章では、サンプルプログラムの使い方について説明しています。



- サンプルプログラムは iOS アプリケーション開発者向けの、ePOS-Print for iOS API を使用した iOS アプリケーションの実装サンプルとして提供します。
- サンプルプログラムのパッケージは、Objective-C ソースファイルを含む Xcode 用 iOS アプリケーションプロジェクトとして提供しています。

概要

提供しているサンプルプログラムには、以下の機能があります。

サンプルプログラム	説明
<ePOSPrintSample> 	<p>サンプルプログラムは、以下の機能を実装しています。 (サンプルプログラムでは、文字 / 画像 / バーコードなどを回転させる機能は、実装していません。)</p> <ul style="list-style-type: none"> • プリンターの検索 • ポートオープン • ポートクローズ • テキスト印刷 • グラフィック印刷 (画像ファイルの印刷) • バーコード印刷 • 2次元シンボル印刷 • ページモード印刷 • 用紙カット • プリンターのステータス取得 • プリンターの機種名 / 言語情報取得 • ログ出力設定 • ステータスイベントの表示 • バッテリーステータスイベントの表示
<ePOSEasySelectSample> 	<p>QRコードを使用して簡単にプリンターに接続します。</p> <ul style="list-style-type: none"> • QRコードからプリンター情報の取得 • 取得したプリンター情報の解析 • 解析結果を用いたポートオープン • プリンターの検索結果からプリンター情報の QRコード作成

使用環境

開発環境

- Xcode Version 4.2 以降



開発環境構築の詳細は、「ePOS-Print SDK for iOS アプリケーション開発環境 - セットアップガイド」を参照してください。

プリンター

- ePOS-Print SDK でサポートしている TM プリンター

ターゲットデバイス

- コンピューターに USB 接続されているデバイス

環境構築

以下の手順でサンプルプログラムの環境を構築します。

- 1 サンプルプログラムの ZIP ファイルを任意のディレクトリーに展開します。
- 2 展開したディレクトリー内に格納されている、“ePOSPrintSample.xcodeproj”、または“ePOSEasySelectSample.xcodeproj” をダブルクリックします。
- 3 Xcode が起動されます。「Scheme」としてターゲットデバイスを選択します。
- 4 左上の [Run] ボタンをクリックします。
- 5 ターゲットデバイスにサンプルプログラムがインストールされ、サンプルプログラムが起動します。

使用方法

ここでは、以下の使い方を説明します。

□ ePOSPrintSample

- [プリンターを検索して印刷する \(16 ページ\)](#)
- [プリンターの機種名を取得する \(23 ページ\)](#)

□ ePOSEasySelectSample

- [QR コードを使ってプリンターを選択する \(24 ページ\)](#)
- [QR コードを印刷する \(24 ページ\)](#)

プリンターを検索して印刷する

以下の手順で使用します。

- 1 サンプルプログラムを起動します。詳細は、[環境構築 \(15 ページ\)](#) を参照してください。
- 2 プリンターを検索します。メイン画面で [Printer Discovery] をタップします。
[Device Type] を選択すると、検索されたプリンターの IP アドレス / Mac アドレス / プリンター名がリスト表示されます。
- 3 表示されたリストの中から、使用するプリンターをタップします。
- 4 プリンターのポートをオープンします。メイン画面で [Open] をタップします。
手順 3 で選択したプリンターの “Device Type” と “IP アドレス / Mac アドレス” が表示されます。[Printer Name] と [Language] を選択します。
- 5 [Status Monitor] を設定します。

項目	説明
Enabled	<ul style="list-style-type: none">• ON: ステータスマニターを有効にし、プリンターステータスの監視を行います。• OFF: ステータスマニターを無効にします。
Interval	(Enabled) を ON に設定した場合、ステータスの監視間隔をミリ秒単位で設定します。

- 6 [Open] をタップします。

7 以下の処理を実行します。

処理	説明
テキスト印刷	メイン画面の (Text) をタップしてください。 詳細は テキスト印刷 (18 ページ) を参照してください。
グラフィック印刷	メイン画面の (Image) をタップしてください。 詳細は グラフィック印刷 (18 ページ) を参照してください。
バーコード印刷	メイン画面の (Barcode) をタップしてください。 詳細は バーコード印刷 (19 ページ) を参照してください。
2次元シンボル印刷	メイン画面の (2D Code) をタップしてください。 詳細は 2次元シンボル印刷 (19 ページ) を参照してください。
ページモード印刷	メイン画面の (Page Mode) をタップしてください。 詳細は ページモード印刷 (19 ページ) を参照してください。
用紙カット	メイン画面の (Cut) をタップしてください。 詳細は 用紙カット (20 ページ) を参照してください。
プリンタステータスの取得	メイン画面の (Get Status) をタップしてください。
ログ出力設定	メイン画面の (Log Settings) をタップしてください。 詳細は ログ出力設定 (20 ページ) を参照してください。

8 以下の処理の実行結果が表示されます。

- 処理実行結果 (エラーステータス / プリンタステータス / バッテリーステータス)
詳細は、[処理実行結果 \(21 ページ\)](#) を参照してください。
- メソッド (API) 実行エラー
詳細は、[メソッド \(API\) 実行エラー \(22 ページ\)](#) を参照してください。

9 処理をすべて終わったら、メイン画面の [Close] をタップし、プリンターのポートをクローズします。

テキスト印刷

以下の手順で実行します。

- 1 [Print Characters] に印字文字列を入力します。
- 2 印字文字列に文字の属性を指定します。以下の属性を指定できます。

属性	説明
Font	文字フォントを設定します。
Align	位置揃えを設定します。
Line Spacing	改行量を設定します。
Language	言語を設定します。
Size	文字の倍率（縦倍 / 横倍）を設定します。
Style	文字修飾（ボールド / アンダーライン）を設定します。
X Position	横位置の開始位置を設定します。
Feed Unit	紙送り量を設定します。

- 3 [Print] をタップし、印刷します。

グラフィック印刷

以下の手順で実行します。

- 1 [Select Image] をタップし、印刷する画像ファイルを選択します。
- 2 [Color Mode] をタップし、階調を選択します。



[Gray 16]（多階調印刷）を選択できる機種は、TM-T70II / TM-T88V / TM-T90II です。

- 3 [Halftone Method] をタップし、ハーフトーンの処理方法を選択します。
- 4 [Brightness] にタップし、数値を入力して明るさを指定します。
- 5 [Print] をタップし、印刷します。

バーコード印刷

以下の手順で実行します。

1 以下のバーコードの設定をします。

設定	説明
Type	バーコードの種類を選択します。
Data	バーコードデータを入力します。
HRI	HRI の位置を設定します。
Font	HRI フォントを設定します。
Module Size(Width, Height)	バーコードのモジュールサイズ(幅 / 高さ)を設定します。

2 [Print] をタップし、印刷します。

2 次元シンボル印刷

以下の手順で実行します。

1 [Type] から 2 次元シンボルの種類を選択します。

2 [Data] に 2 次元シンボルデータを入力します。

3 2 次元シンボルの種類ごとに、以下を設定します。

設定	説明
Error Correction Level (PDF417, QR Code, Aztec Code, DataMatrix)	エラー訂正レベルを設定します。
Module Size(Width, Height)	2 次元シンボルのモジュールサイズ(幅 / 高さ)を設定します。
Max Size	2 次元シンボルの最大サイズを設定します。

4 [Print] をタップし、印刷します。

ページモード印刷

以下の手順で実行します。

1 [Print Characters] に印字文字列を入力します。

2 [Print Area] で印字領域の設定をします。

設定	説明
X	横方向の原点を設定します。
Y	縦方向の原点を設定します。
Width	印字領域の幅を設定します。
Height	印字領域の高さを設定します。

3 [Print] をタップし、印刷します。

用紙カット

以下の手順で実行します。

- 1 [Type] で紙送りしてカットするか設定します。
- 2 [Print] をタップし、カットを実行します。

ログ出力設定

以下の手順で設定します。

- 1 [Enabled] に、ログ出力の有無と、ログの出力先を設定します。
- 2 ログの出力先に合わせて、以下を設定します。

設定	説明
IP Address	TCP 通信の IP アドレスを指定します。
Port	TCP 通信用のポート番号を指定します。
Log Size	端末のストレージに保存する、ログの最大容量を指定します。
Log Level	出力するログのレベルを設定します。

- 3 [Save Settings Permanently] に、設定の保存方法を設定します。
- 4 [Setting] をタップし、ログ出力の設定を有効にします。
- 5 印刷実行後、ログファイルを確認します。
詳細は、[setLogSettings \(134 ページ\)](#) を参照してください。

実行結果

処理実行結果

以下の表示がされます。

- Result: 以下のエラーステータスが表示されます。

表示文字列	説明
SUCCESS	成功
ERR_PARAM	不正なパラメーターが渡された
ERR_ILLEGAL	不適切な方法で使用された
ERR_PROCESSING	処理を実行できなかった
ERR_TIMEOUT	処理がタイムアウトされた
ERR_CONNECT	デバイスとの通信に失敗した
ERR_MEMORY	処理に必要なメモリーが確保できなかった
ERR_OFF_LINE	オフライン状態
ERR_FAILURE	その他のエラーが発生した

- Status: 以下のプリンターステータスが表示されます。

表示文字列	説明
NO_RESPONSE	プリンター無応答
PRINT_SUCCESS	印刷終了
DRAWER_KICK	ドロアーキックコネクタ 3 番ピンの状態 = "H" (TM-P60II 以外)
BATTERY_OFFLINE	バッテリーオフライン状態 (TM-P60II)
OFF_LINE	オフライン状態
COVER_OPEN	カバーが開いている
PAPER_FEED	紙送りスイッチによる紙送り中
WAIT_ON_LINE	オンライン復帰待ち中
PANEL_SWITCH	紙送りスイッチが押されている
MECHANICAL_ERR	メカニカルエラー発生
AUTOCUTTER_ERR	オートカッターエラー発生
UNRECOVER_ERR	復帰不可能エラー発生
AUTORECOVER_ERR	自動復帰エラー発生
RECEIPT_NEAR_END	ロール紙ニアエンド検出器に用紙なし
RECEIPT_END	ロール紙エンド検出器に用紙なし

- Battery Status: 以下が表示されます。

表示文字列	説明
0xnxxx	バッテリーステータス 詳細は、 バッテリーステータス (41 ページ) を参照してください。

メソッド (API) 実行エラー

以下の表示がされます。

- Error Code: 以下のエラーステータスが表示されます。

表示文字列	説明
ERR_PARAM	不正なパラメーターが渡された
ERR_OPEN	オープン処理に失敗した
ERR_CONNECT	デバイスとの通信に失敗した
ERR_TIMEOUT	指定された時間内にすべてのデータを送信できなかった
ERR_MEMORY	処理に必要なメモリーが確保できなかった
ERR_ILLEGAL	不適切な方法で使用された
ERR_PROCESSING	処理を実行できなかった
ERR_UNSUPPORTED	サポートしていない機種名または言語使用が指定された
ERR_OFF_LINE	プリンターがオフライン状態だった
ERR_FAILURE	その他のエラーが発生した

- Method: メソッド実行エラーになった API が表示されます。

プリンターの機種名を取得する



プリンターの機種名の取得は、コマンド送受信 API を使用しています。詳細は、[コマンドの送受信 \(137 ページ\)](#) を参照してください。

以下の手順で 사용합니다。

- 1 サンプルプログラムを起動します。詳細は、[環境構築 \(15 ページ\)](#) を参照してください。
- 2 プリンターを検索します。メイン画面で [Printer Discovery] をタップします。
[Device Type] を選択すると、検索されたプリンターの IP アドレス / Mac アドレス / プリンター名がリスト表示されます。
- 3 表示された [Printer List] の中から、使用するプリンターをタップします。
- 4 メイン画面の [Get Printer Name] をタップします。
- 5 [Get Printer Name] をタップします。
- 6 以下が表示されます。

表示内容	説明
Printer Name	プリンターの機種名が表示されます。
Language	プリンターの言語仕様が表示されます。

QRコードを使ってプリンターを選択する

以下の手順で 사용합니다。

- 1 サンプルプログラムを起動します。詳細は、[環境構築 \(15 ページ\)](#) を参照してください。
- 2 メイン画面の [Quick pairing and Easy print by QR code] をタップします。
- 3 QRコードをカメラで読み取ります。
カメラプレビューの赤枠内に QRコードを入れてください。
- 4 [Print] をタップし、印刷します。

QRコードを印刷する

以下の手順で 사용합니다。

- 1 サンプルプログラムを起動します。詳細は、[環境構築 \(15 ページ\)](#) を参照してください。
- 2 メイン画面の [Print QR code] をタップします。
- 3 QRコード印刷用画面の [Find] をタップします。
[Printer List] に検索されたプリンターがリスト表示されます。
- 4 使用したいプリンターを選択します。
- 5 [Print] をタップし、印刷します。

プログラミングガイド

本章では、ePOS-Print SDK を使用したアプリケーション開発のプログラミング方法について説明します。



ePOS-Print SDK for iOS を使用した、iOS アプリケーション開発環境の構築方法については、「ePOS-Print SDK for iOS アプリケーション開発環境 - セットアップガイド」を参照してください。

ePOS-Print SDK for iOS の組み込み方法

ePOS-Print SDK for iOS の組み込み方法について説明します。
以下の手順で組み込んでください。

- 1 Xcode で新しいプロジェクトを作成します。
- 2 提供された以下の Objective-C ヘッダーを、Xcode の [Project Navigator] の対象プロジェクトの任意の階層にドラッグします。
 - ePOS-Print.h
 - ePOSEasySelect.h



ePOSEasySelect.h は、プリンター簡単選択を使用しない場合、必要ありません。

- 3 ExternalAccessory.framework を以下の手順で組み込みます。
 1. Project Navigator から Project ファイル（ルートのもの）を選択します。
 2. Targets → Build Phases を選択します。
 3. Link Binary With Libraries を展開し、+ ボタンを押します。
 4. ExternalAccessory.framework を選択し、Add ボタンを押します。
- 4 提供された以下のスタティックライブラリーを、Xcode の [Project Navigator] の対象プロジェクトの任意の階層にドラッグします。
 - libeposprint.a
 - libeposeasyselect.a



libeposeasyselect.a は、プリンター簡単選択を使用しない場合、必要ありません。

- 5 使用したいアプリケーションの *.m ソースファイルで、Objective-C ヘッダーのインポート定義を記載します。下記を参照してください。

```
#import "ePOS-Print.h"
```

```
#import "ePOSEasySelect.h"
```



プリンター簡単選択を使用しない場合、ePOSEasySelect.h を定義する必要はありません。

6 ExternalAccessory.framework を以下の手順で組み込みます。 protocol name を以下の手順で設定します。

Key	Type	Value
Localization native development region	String	en
Bundle display name	String	\$(PRODUCT_NAME)
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	test.\$(PRODUCT_NAME:rfc1034identifier)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Application requires iPhone environment	Boolean	YES
Required device capabilities	Array	(1 item)
Supported interface orientations	Array	(3 items)
Supported external accessory protocols	Array	(1 item)
Item 0	String	com.epson.espos

1. Project Navigator から xxxx.plist を選択（ファイル名は、Project 名 -info になる）します。
2. ポップアップメニューより Add Row を選択します。
3. Supported external accessory protocols を選択します。
4. 手順 3 で追加した項目を展開します。
5. Item 0 の Value として com.epson.espos と入力します。

ePOS-Print SDK

印刷モードについて

印字モードにはスタンダードモードとページモードがあります。

スタンダードモード

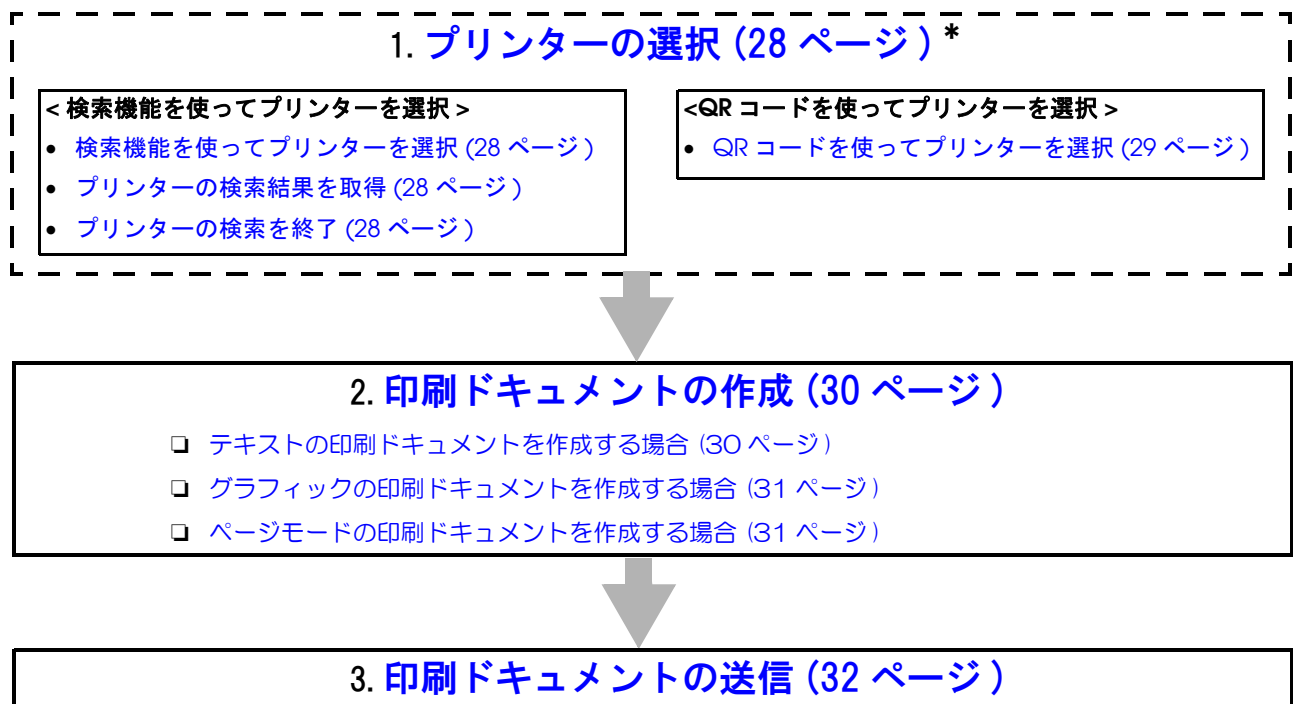
スタンダードモードとは、1行単位で印字する印字モードです。文字サイズ、画像、バーコードなどの高さに合わせて改行量が調整されます。レシートなど、印字量に合わせて用紙の長さが増える印刷に向いています。

ページモード

ページモードとは、印字領域を設定して印字データを展開し、一括印字する印字モードです。印字位置（座標）に、文字、画像、バーコードなどを展開します。

プログラミングフロー

以下のフローでプログラミングします。



*: 任意のプロセスです。



あらかじめプリンターの状態を確認して送信するプログラミングをすると、確実に印刷できます。方法については、[プリンターの状態を確認してから印刷する \(34 ページ\)](#) を参照してください。

プリンターの選択

検索機能を使ってプリンターを選択

EpsonIoFinder クラスの [start \(125 ページ\)](#) を使って、プリンターの検索を開始します。以下のプログラミングを参考にしてください。

```
int result = EPSONIO_OC_SUCCESS;
int devicetype = EPSONIO_OC_DEVTYPE_TCP;

// 検索開始
switch (devicetype) {
    //Wi-Fi®/Ethernet デバイスの場合
    case EPSONIO_OC_DEVTYPE_TCP:
        result = [EpsonIoFinder start:EPSONIO_OC_DEVTYPE_TCP FindOption:option];
        break;
    //Bluetooth デバイスの場合
    case EPSONIO_OC_DEVTYPE_BLUETOOTH:
        result = [EpsonIoFinder start:EPSONIO_OC_DEVTYPE_BLUETOOTH FindOption:option];
        break;
    default:
        result = [EpsonIoFinder start:EPSONIO_OC_DEVTYPE_TCP FindOption:option];
        break;
}
```

プリンターの検索結果を取得

EpsonIoFinder クラスの [getDeviceInfoList \(128 ページ\)](#) を使って、プリンターの検索結果を取得します。以下のプログラミングを参考にしてください。取得した結果を [openPrinter \(100 ページ\)](#) で使用してください。

```
int errStatus = EPSONIO_OC_SUCCESS;

// デバイス一覧の取得
NSArray *array = [[NSArray alloc]initWithArray:
    [EpsonIoFinder getDeviceInfoList:&errStatus
    FilterOption:EPSONIO_OC_PARAM_DEFAULT]];

```



プリンターの検索に時間がかかるため、EpsonIoFinder クラスの start を呼んだ直後に getDeviceInfoList を呼んだ場合、検索結果が無いこともあります。

プリンターの検索を終了

EpsonIoFinder クラスの [stop \(127 ページ\)](#) を使って、プリンターの検索を終了します。以下のプログラミングを参考にしてください。

```
// 検索終了
int errStatus = [EpsonIoFinder stop];

```

QR コードを使ってプリンターを選択

EposEasySelect クラスの [parseQR \(131 ページ\)](#) を使って、QR コードを解析します。

以下のプログラミングを参考にしてください。取得した結果を [openPrinter \(100 ページ\)](#) で使用してください。

```
id easySelect = [EposEasySelect alloc];
NSString *data;

// カメラ画像から取得した QR コードデータを格納

// QR コードの解析
EposEasySelectInfo *easySelectInfo = [easySelect parseQR:data];
if( easySelectInfo == nil){
    // 簡単選択用の QR コードでは無かった場合
    return ;
}

id printer = [[EposPrint alloc] init];

// 解析したデータを使って、プリンターをオープン
if ( printer != nil ) {
    errorStatus = [printer openPrinter:easySelectInfo.deviceType
                                DeviceName:easySelectInfo.macAddress];

    // 解析したデータを使って EposBuilder クラスを作成
    id builder = [[EposBuilder alloc]
                  initWithPrinterModel: easySelectInfo.printerName
                  Lang: EPOS_OC_MODEL_JAPANESE];

    errorStatus = [printer closePrinter];
    [printer release];
}
```

プリンター簡単選択用の QR コードを作成する方法

- プリンター簡単選択用の QR コードを自動印刷できる機種の場合
ダイナミックステータスシートの QR コードを使用してください。
ダイナミックステータスシートの印刷方法は機種ごとの Technical Reference Guide を参照してください。
- プリンター簡単選択用の QR コードを自動印刷できない機種の場合
[createQR \(132 ページ\)](#) を使用して QR コードを作成してください。
サンプルプログラムの QR コードを作成するを参照してください。

印刷ドキュメントの作成

印刷ドキュメントは、[EposBuilder クラス \(43 ページ\)](#) で作成します。

コンストラクターで EposBuilder クラスを作成し、EposBuilder クラスの各 API で印刷ドキュメントを作成します。

以下のプログラミングを参考にしてください。

```
//EposBuilder クラスのインスタンスを初期化
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                     Lang: EPOS_OC_MODEL_JAPANESE];

if ( builder != nil ) {
    // 印刷ドキュメントの作成
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextLang: EPOS_OC_LANG_JA];
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
    errorStatus = [builder addTextSize: 3 Height: 3];
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
    [builder release];
}
```

テキストの印刷ドキュメントを作成する場合

テキストの印刷ドキュメントを作成する場合、テキストの各 API でフォントの設定を命令バッファに格納し、印刷ドキュメントを作成します。以下のプログラミングを参考にしてください。

文字列「Hello, World!」を以下の設定で印刷ドキュメントを作成する場合

- フォント: FontA
- 倍率: 幅 4 倍、高さ 4 倍
- スタイル: 太字

```
//EposBuilder クラスのインスタンスを初期化
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                     Lang: EPOS_OC_MODEL_JAPANESE];

if ( builder != nil ) {
    // 印刷ドキュメントの作成
    int errorStatus = EPOS_OC_SUCCESS;
    //< 印字文字の設定 >
    errorStatus = [builder addTextLang: EPOS_OC_LANG_JA];
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
    errorStatus = [builder addTextSize: 4 Height: 4];
    errorStatus = [builder addTextStyle: EPOS_OC_FALSE U1: EPOS_OC_FALSE
                                     Em: EPOS_OC_TRUE Color: EPOS_OC_PARAM_UNSPECIFIED];
    //< 印刷データを指定 >
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}
```

グラフィックの印刷ドキュメントを作成する場合

グラフィックの印刷ドキュメントを作成する場合、グラフィックは、UIImage クラスを EposBuilder クラスの [addImage \(62 ページ\)](#) で命令バッファーに格納します。以下のプログラミングを参考にしてください。

```
UIImage * imageData = [UIImage imageNamed:@"Sample.png"];

//EposBuilder クラスのインスタンスを初期化
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                                    Lang: EPOS_OC_MODEL_JAPANESE];

if ( builder != nil ) {
    //印刷ドキュメントの作成
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256 Height: 256
                                Color: EPOS_OC_PARAM_DEFAULT];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}
```



グラフィック印刷する方法には、プリンターの NV メモリーに登録されているグラフィックを印字する方法もあります。詳細は、[addLogo \(70 ページ\)](#) を参照してください。

ページモードの印刷ドキュメントを作成する場合

EposBuilder クラスの [addPageBegin \(81 ページ\)](#) を命令バッファーに格納することで、ページモードが開始されます。印字領域 ([addPageArea \(83 ページ\)](#)) と印字開始位置 ([addPagePosition \(85 ページ\)](#)) を命令バッファーに格納します。印字開始位置は、印字データに合わせて指定します。その後、各 API を命令バッファーに格納し印字データを作成します。ページモードの終了は [addPageEnd \(82 ページ\)](#) を命令バッファーに格納します。

文字列「Hello, World!」を以下の設定で印刷ドキュメントを作成する場合

- ページモードの印字領域 (ドット単位):
横方向原点:100, 縦方向原点:50, 幅:200, 高さ:100
- ページモードの印字位置 (ドット単位):
横方向の印字位置:0, 縦方の印字位置:42
- フォント: FontA
- 倍率: 幅 2 倍、高さ 2 倍
- スタイル: 太字

```
//EposBuilder クラスのインスタンスを初期化
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                                    Lang: EPOS_OC_MODEL_JAPANESE];

if ( builder != nil ) {
    //印刷ドキュメントの作成
    //＜ページモード開始＞
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 200 Height: 100];
    errorStatus = [builder addPagePosition: 0 Y: 42];
    //＜印字文字の設定＞
    errorStatus = [builder addTextLang: EPOS_OC_LANG_JA];
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
    errorStatus = [builder addTextSize: 2 Height: 2];
    errorStatus = [builder addTextStyle: EPOS_OC_FALSE U1: EPOS_OC_FALSE
                                Em: EPOS_OC_TRUE Color: EPOS_OC_PARAM_UNSPECIFIED];
    //＜印刷データを指定＞
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    //＜ページモード終了＞
    errorStatus = [builder addPageEnd];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}
```

印刷ドキュメントの送信

印刷ドキュメントは、[EposPrint クラス \(45 ページ\)](#) で送信します。

コンストラクターで EposPrint クラスを作成し、sendData で、印刷ドキュメントの命令バッファを格納した EposBuilder クラスのインスタンスを指定して送信します。

EposBuilder に格納された命令バッファは、[clearCommandBuffer \(47 ページ\)](#) を実行するまで保管されます。[sendData \(107 ページ\)](#) 成功後に clearCommandBuffer を実行してください。



同じ印刷ドキュメントを繰り返し印刷したい場合、clearComandndBuffer を実行する必要はありません。

以下のプログラミングを参考にしてください。

```
//EposBuilder クラスのインスタンスを初期化
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                                    Lang: EPOS_OC_MODEL_JAPANESE];

if (builder != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    // 印刷ドキュメントの作成
    // 印字文字の設定
    errorStatus = [builder addTextLang: EPOS_OC_LANG_JA];
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
    errorStatus = [builder addTextSize: 4 Height: 4];
    errorStatus = [builder addTextStyle: EPOS_OC_FALSE Ul: EPOS_OC_FALSE
                                         Em: EPOS_OC_TRUE Color: EPOS_OC_PARAM_UNSPECIFIED];
    // 印刷データを指定
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
    //EposPrint クラスのインスタンスを初期化
    id printer = [[EposPrint alloc] init];
    long status;
    // 印刷ドキュメントを送信
    if (printer != nil) {
        // プリンターとの通信を開始
        errorStatus = [printer openPrinter:deviceList.deviceType
                                         DeviceName:deviceList.deviceName Enabled:EPOS_OC_TRUE
                                         Interval:EPOS_OC_PARAM_DEFAULT Timeout:EPOS_OC_PARAM_DEFAULT];

        // データを送信
        errorStatus = [printer sendData:builder Timeout:10000 Status:&status];
        // 命令バッファの削除
        if ((status & EPOS_OC_ST_PRINT_SUCCESS) == EPOS_OC_ST_PRINT_SUCCESS) {
            errorStatus = [builder clearCommandBuffer];
        }
        // プリンターとの通信を終了
        errorStatus = [printer closePrinter];
        [printer release];
    }
    [builder release];
}
```


設定用の命令バッファの有効範囲

設定用に使用される EposBuilder クラスの addXXX の有効範囲は、addXXX 設定後、sendData が実行されるまでです。設定した値は、sendData の実行ごとに初期化されます。以下を参考にしてください。

例

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                     Lang: EPOS_OC_MODEL_JAPANESE];

Builder builder = new Builder("TM-T88V", Builder.MODEL_JAPANESE);

errorStatus = [builder addText: @"Hello, World!\n"]; addTextFont の設定が無効な文字列
errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
errorStatus = [builder addText: @"Hello, World!\n"]; addTextFont の設定が有効な文字列 (FONT_A)
errorStatus = [printer sendData:builder Timeout:10000 Status:&status];
errorStatus = [builder addText: @"Hello, World!\n"]; addTextFont の設定が無効な文字列
errorStatus = [builder addTextFont: EPOS_OC_FONT_B];
errorStatus = [builder addText: @"Hello, World!\n"]; addTextFont の設定が有効な文字列 (FONT_B)
errorStatus = [printer sendData:builder Timeout:10000 Status:&status];
```

プリンターの状態を確認してから印刷する

あらかじめプリンターの状態を確認してから印刷すると、確実に印刷できます。

空の印刷データを送信し、プリンターがオンライン状態の場合に印刷します。以下を参考にしてください。

```
// 印刷ドキュメントの作成
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                     Lang: EPOS_OC_MODEL_JAPANESE];

if (builder != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    // 印刷ドキュメントの作成
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}

// 確認用 EposBuilder クラスのインスタンスを初期化
id conBuilder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                     Lang: EPOS_OC_MODEL_JAPANESE];

// EposPrint クラスのインスタンスを初期化
id printer = [[EposPrint alloc] init];
long status;
if (printer != nil) {
    // <プリンターとの通信を開始>
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP DeviceName:@"192.168.192.168"];

    // <確認用データを送信>
    errorStatus = [printer sendData:conBuilder Timeout:10000 Status:&status];

    if (errorStatus == EPOS_OC_SUCCESS && (status & EPOS_OC_ST_OFF_LINE) != EPOS_OC_ST_OFF_LINE) {
        // <印刷データを送信>
        errorStatus = [printer sendData:builder Timeout:10000 Status:&status];
    } else if ((status & EPOS_OC_ST_OFF_LINE) != EPOS_OC_ST_OFF_LINE) {
    }

    } else if (errorStatus == EPOS_OC_ERR_CONNECT) {
        // <プリンターとの通信を終了>
        errorStatus = [printer closePrinter];

        // <プリンターとの通信を開始>
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP DeviceName:@"192.168.192.168"];

        // <印刷データを送信>
        errorStatus = [printer sendData:builder Timeout:10000 Status:&status];
    } else {
    }

    // <プリンターとの通信を終了>
    errorStatus = [printer closePrinter];
    [printer release];
}
[conBuilder release];
[builder release];
}
```

①

②

③

④

⑤

⑥

- 1 印刷データを作成します。
- 2 プリンターの状態を確認するために空の印刷データを作成します。
- 3 ②で作成した印刷データを送信します。
- 4 ②で作成した印刷データが正常に送信され、プリンターがオンライン状態の場合に、続けて①で作成した印刷データを送信します。
- 5 ②で作成した印刷データが正常に送信され、プリンターがオフライン状態の場合は、プリンターのオフラインとなる要因を取り除いてください。
(プリンターのカバーオープン、用紙切れなど)
- 6 ②で作成した印刷データが正常に送信されなかった場合、プリンターとの通信を終了して、再度プリンターとの通信を開始した後、印刷データを送信します。

プリンタステータスを自動で取得

ePOS-Print SDK では、プリンタの状態をコールバックに自動でアプリケーションに通知できます。

以下を参考にしてください。

```
// プリンタステータスを通知するコールバックメソッドを実装
- (void)onStatusChange:(NSString *)deviceName Status:(NSNumber *)status
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        // プリンタステータスの通知先のコールバックメソッドを登録
        [printer setStatusChangeEventCallback @selector(onStatusChange:Status:)
        Target:self];

        // プリンタとの通信、およびプリンタステータスのモニタリングを開始
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
        Name:@"192.168.192.168"
        Enabled: EPOS_OC_TRUE
        Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

① / ④

②

③

1 イベント発生時の通知先のコールバックメソッドを実装します。



上記では、プリンタステータスを [openPrinter \(100 ページ\)](#) で指定した間隔で通知するコールバックメソッドを実装しています。
ePOS-Print には、カバーオープンやドロアオープンのイベントといった、プリンタステータスごとのコールバックメソッドも用意されています。用途に応じて使い分けてください。ePOS-Print で使用できるコールバックメソッドは、[イベント一覧 \(37 ページ\)](#) を参照してください。

2 プリンタステータスの通知先を登録します。

3 [openPrinter \(100 ページ\)](#) を使って、プリンタステータスのモニタリングを開始します。

4 ①で実装したイベントにプリンタステータスを通知します。



プリンタステータスの通知を終了した場合、EposPrint クラスの [closePrinter \(106 ページ\)](#) で終了します。

イベント一覧



コールバックメソッドの詳細は、下記、コールバックメソッド登録 API を説明している、[API リファレンス \(43 ページ\)](#) を参照してください。

イベント	コールバックメソッド登録 API
プリンタステータスの通知	setStatusChangeEventCallback (111 ページ)
オンラインの通知	setOnlineEventCallback (112 ページ)
オフラインの通知	setOfflineEventCallback (113 ページ)
無応答の通知	setPowerOffEventCallback (114 ページ)
カバークローズの通知	setCoverOkEventCallback (115 ページ)
カバーオープンの通知	setCoverOpenEventCallback (116 ページ)
用紙ありの通知	setPaperOkEventCallback (117 ページ)
用紙残量少の通知	setPaperNearEndEventCallback (118 ページ)
用紙なしの通知	setPaperEndEventCallback (119 ページ)
ドロアークローズの通知	setDrawerClosedEventCallback (120 ページ)
ドロアーオープンの通知	setDrawerOpenEventCallback (121 ページ)
バッテリー残量なしの通知	setBatteryLowEventCallback (122 ページ)
バッテリー残量ありの通知	setBatteryOkEventCallback (123 ページ)
バッテリーステータスの通知	setBatteryStatusChangeEventCallback (124 ページ)

ステータス

ePOS-Print SDK for iOS には、以下のステータスが定義されています。

ステータス	説明
エラーステータス	各クラスの API 実行時の戻り値です。 詳細は、 エラーステータスと対処方法 (38 ページ) を参照してください。
プリンターステータス	印刷データ送信時のプリンターのステータスです。 プリンターステータスは、 sendData (107 ページ) を実行時に取得します。 詳細は、 プリンターステータスと対処方法 (40 ページ) を参照してください。
バッテリーステータス	プリンターのバッテリー残量のステータスです。 詳細は、 バッテリーステータス (41 ページ) を参照してください。

エラーステータスと対処方法

エラーステータス	要因	対処方法
EPOS_OC_SUCCESS	処理に成功した。	-
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。 < 例 > • Nil など、不正な引数を渡された。 • サポートしていない範囲の値が指定された。	パラメーターの指定が間違えています。 パラメーターを見直してください。
EPOS_OC_ERR_OPEN	オープン処理に失敗した < 例 > 指定したプリンターに接続できなかった。	iOS デバイスとプリンターを確認してください。 (プリンターの電源、通信接続状態など)
EPOS_OC_ERR_CONNECT	プリンターとの通信に失敗した。 < 例 > プリンターへのデータ送信に失敗した。	closePrinter を実行後、openPrinter を実行して、デバイスとの通信を復帰させてください。 なお、インターフェイスが Bluetooth の場合、openPrinter を実施するためには、iOS デバイスとプリンターとのペアリング接続がされている必要があります。
EPOS_OC_ERR_TIMEOUT	指定されたタイムアウト時間を越えた。 < 例 > 指定された時間内にすべてのデータを送信できなかった。	タイムアウト時間を確認してください。 タイムアウト時間は、印刷所要時間以上に設定してください。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。	不要なアプリケーションを終了してください。
EPOS_OC_ERR_ILLEGAL	不適切な方法で使用された。 < 例 > プリンターがオープンされていない状態でプリンターにコマンドを送信する API が呼び出された。	API を適切な方法で使用してください。 27 ページ「プログラミングフロー」 を参照してください。

エラーステータス	要因	対処方法
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。 < 例 > 同様の処理を他のスレッドで実行中のため、処理を実行できなかった。	アプリケーションの処理のタイミングを見直し、処理が重ならないようにしてください。
EPOS_OC_ERR_UNSUPPORTED	サポートしていない機種名または言語仕様が指定された。	サポートしていない機種では使用できません。
EPOS_OC_ERR_OFF_LINE	プリンターがオフライン状態だった。	オフラインになる要因を取り除いてください。 (プリンターのカバーオープン、用紙切れなど)
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。	<ul style="list-style-type: none">• iOSデバイスの通信設定を確認してください。(Wi-Fi の接続設定、Bluetooth の接続設定など)• 実行環境に問題がないか確認してください。

プリンターステータスと対処方法

プリンターステータス	要因	対処方法
EPOS_OC_ST_NO_RESPONSE (0x00000001)	<ul style="list-style-type: none"> TM プリンターの電源が入っていない 通信が確立されていない 通信ケーブルが抜かれている 	電源、ケーブルなど プリンターの状態や通信状態を確認してください。
EPOS_OC_ST_PRINT_SUCCESS (0x00000002)	印刷終了	-
<TM-P シリーズ以外 > EPOS_OC_ST_DRAWER_KICK (0x00000004)	ドロアーキックコネクター 3 番ピンの状態 = "H"	-
<TM-P シリーズ > EPOS_OC_ST_BATTERY_OFFLINE (0x00000004)	バッテリー残量によるオフライン状態	バッテリーを充電してください。
EPOS_OC_ST_OFF_LINE (0x00000008)	オフライン状態	オフラインになる要因を取り除いてください。(カバーオープン、用紙切れなど)
EPOS_OC_ST_COVER_OPEN (0x00000020)	カバーが開いている	プリンターのカバーを閉めてください。
EPOS_OC_ST_PAPER_FEED (0x00000040)	紙送りスイッチによる紙送信中	-
EPOS_OC_ST_PANEL_SWITCH (0x00000200)	プリンターのスイッチ、またはボタンが押されている	-
EPOS_OC_ST_MECHANICAL_ERR (0x00000400)	メカニカルエラー発生	エラーの原因を取り除き、プリンターの電源を再投入してください。
EPOS_OC_ST_AUTOCUTTER_ERR (0x00000800)	オートカッターエラー発生	エラーの原因を取り除き、プリンターの電源を再投入してください。
EPOS_OC_ST_UNRECOVER_ERR (0x00002000)	プリンターに印刷できない復帰不可能エラー発生	ただちにプリンターの電源を切ってください。
EPOS_OC_ST_AUTORECOVER_ERR (0x00004000)	ヘッドの温度が上昇し、自動復帰エラーが発生	時間の経過により、ヘッドの温度が下降すれば自動的に解除されます。
EPOS_OC_ST_RECEIPT_NEAR_END (0x00020000)	用紙残量が少なくなった	プリンターに用紙を入れてください。
EPOS_OC_ST_RECEIPT_END (0x00080000)	用紙がなくなった	プリンターに用紙を入れてください。
EPOS_OC_ST_BUZZER (0x01000000)	ブザーが鳴っている (対応機器のみ)	-
	ラベル除去待ち状態 (対応機器のみ)	ラベルを取り除いてください。

バッテリーステータス

バッテリーステータスは、以下の 16 ビット (0x0000) で構成されています。

ビット	説明
上位 8 ビット	共通のバッテリーステータス (詳細は、 共通のバッテリーステータス (上位 8 ビット) (41 ページ) を参照してください。)
下位 8 ビット	機種専用のバッテリーステータス (詳細は、 プリンター別サポート情報 (149 ページ) を参照してください。)



バッテリーステータス取得不可能状態、もしくは機種がバッテリーステータスに対応していない場合、“0x0000” を返します。

共通のバッテリーステータス (上位 8 ビット)

バッテリーステータス	要因
0x30	AC アダプターが接続されている
0x31	AC アダプターが接続されていない



API リファレンス

本章では、ePOS-Print SDK for iOS で用意されている API について説明しています。

ePOS-Print API

ePOS-Print API は、印刷ドキュメントを作成し、印刷処理を行う API です。

以下のクラスが用意されています。

□ EposBuilder クラス ([43 ページ](#))

□ EposPrint クラス ([45 ページ](#))



プリンターによって、使用できる API や指定可能な設定値は異なります。

詳細は、[プリンターごとのサポート API 一覧 \(147 ページ\)](#)、および[プリンター別サポート情報 \(149 ページ\)](#)を参照してください。

EposBuilder クラス

印字する文字列やグラフィックの印刷、用紙カットなどプリンターの制御命令の印刷ドキュメントを作成します。

以下の API が用意されています。

API		説明	ページ
initWithPrinterModel		EposBuilder クラスのインスタンスを初期化	46
命令バッファのクリア	clearCommandBuffer	各 API で追加した命令バッファをクリア	47
テキスト	addTextAlign	位置揃え設定を命令バッファに追加	48
	addTextLineSpace	改行量設定を命令バッファに追加	49
	addTextRotate	倒立印字設定を命令バッファに追加	50
	addText	文字印字を命令バッファに追加	51
	addTextLang	言語設定を命令バッファに追加	52
	addTextFont	文字フォント設定を命令バッファに追加	53
	addTextSmooth	文字スムージング設定を命令バッファに追加	54
	addTextDouble	文字倍角設定を命令バッファに追加	55
	addTextSize	文字倍率設定を命令バッファに追加	56
	addTextStyle	文字装飾設定を命令バッファに追加	57
	addTextPosition	文字印字位置設定を命令バッファに追加	59
紙送り	addFeedUnit	ドット単位の紙送りを命令バッファに追加	60
	addFeedLine	行単位の紙送りを命令バッファに追加	61
	addFeedPosition	ラベル / ブラックマーク紙の紙送りを命令バッファに追加	95

API		説明	ページ
グラフィック	addImage	多階調のラスターイメージ印字を命令バッファに追加 イメージデータを圧縮して命令バッファに追加 (Bluetooth インターフェイス)	62
	addImage (旧フォーマット)	多階調のラスターイメージ印字を命令バッファに追加 (イメージデータの圧縮はできません (Bluetooth インターフェイス))	65
	addImage (旧フォーマット)	ラスターイメージ印字を命令バッファに追加 (イメージデータの圧縮はできません (Bluetooth インターフェイス)) 多階調は印刷できません)	68
	addLogo	NV ロゴ印字を命令バッファに追加	70
バーコード	addBarcode	バーコード印字を命令バッファに追加	71
	addSymbol	2次元シンボル印字を命令バッファに追加	76
ページモード	addPageBegin	ページモード開始を命令バッファに追加	81
	addPageEnd	ページモード終了を命令バッファに追加	82
	addPageArea	ページモード印字領域設定を命令バッファに追加	83
	addPageDirection	ページモード印字方向設定を命令バッファに追加	84
	addPagePosition	ページモード印字位置設定を命令バッファに追加	85
	addPageLine	ページモード直線描画を命令バッファに追加	86
	addPageRectangle	ページモード四角形描画を命令バッファに追加	88
カット	addCut	用紙カットを命令バッファに追加	89
ドロアーキック	addPulse	ドロアーキックを命令バッファに追加	90
ブザー	addSound	ブザー鳴動を命令バッファに追加	91
	addSound (旧フォーマット)	ブザー鳴動を命令バッファに追加 (鳴動周期は設定できません)	93
用紙レイアウト	addLayout	用紙レイアウト情報を命令バッファに追加	96
コマンド送信	addCommand	コマンドを命令バッファに追加	98

EposPrint クラス

EposBuilder クラスで作成した印刷ドキュメントを送信してプリンターを制御したり、送信結果や通信状態を監視したりします。

API	説明	ページ
init	ePOS-Print クラスのインスタンスを初期化	99
openPrinter	プリンターとの通信を開始	100
openPrinter(旧フォーマット)	プリンターとの通信を開始 (タイムアウトが設定できません)	102
openPrinter(旧フォーマット)	プリンターとの通信を開始 (プリンターステータスの取得、およびタイムアウトが設定できません)	104
closePrinter	プリンターとの通信を終了	106
sendData	プリンターにコマンドを送信	109
sendData(旧フォーマット)	プリンターにコマンドを送信 (バッテリーステータスは取得できません)	107
setStatusChangeEventCallback	プリンターステータスのコールバックメソッドを登録	111
setOnlineEventCallback	オンラインイベントのコールバックメソッドを登録	112
setOfflineEventCallback	オフラインイベントのコールバックメソッドを登録	113
setPowerOffEventCallback	無応答イベントのコールバックメソッドを登録	114
setCoverOkEventCallback	カバークローズイベントのコールバックメソッドを登録	115
setCoverOpenEventCallback	カバーオープンイベントのコールバックメソッドを登録	116
setPaperOkEventCallback	用紙ありイベントのコールバックメソッドを登録	117
setPaperNearEndEventCallback	用紙残量少イベントのコールバックメソッドを登録	118
setPaperEndEventCallback	用紙なしイベントのコールバックメソッドを登録	119
setDrawerClosedEventCallback	ドロアークローズイベントのコールバックメソッドを登録	120
setDrawerOpenEventCallback	ドロアオープンイベントのコールバックメソッドを登録	121
setBatteryLowEventCallback	バッテリー残量なしイベントの通知先を登録	122
setBatteryOkEventCallback	バッテリー残量ありイベントの通知先を登録	123
setBatteryStatusChangeEventCallback	バッテリーステータスのコールバックメソッドを登録	124

initWithPrinterModel

EposBuilder クラスのインスタンスを初期化します。

構文

```
(id) initWithPrinterModel:(NSString *)printerModel  
Lang:(int)lang;
```

パラメーター

- printerModel: 対象のプリンターの機種名を指定します。

設定値	説明
"TM-P20"	• TM-P20 • TM-P20 iOS Bluetooth モデル
"TM-P60II"	• TM-P60II • TM-P60II iOS Bluetooth モデル
"TM-T20II"	TM-T20II iOS Bluetooth モデル
"TM-T70"	TM-T70
"TM-T70II"	• TM-T70II • TM-T70II iOS Bluetooth モデル
"TM-T88V"	• TM-T88V • TM-T88V iOS Bluetooth モデル
"TM-T90II"	TM-T90II

- lang: プリンターの言語仕様を指定します。

設定値	説明
EPOS_OC_MODEL_ANK	ANK モデル
EPOS_OC_MODEL_JAPANESE	日本語モデル

戻り値

処理に成功した場合、初期化済みの EposBuilder クラスインスタンスが返ります。

処理に失敗した場合、nil が返ります。処理に失敗する原因には、以下の要因があります。

- 不正なパラメーターが指定された。
- メモリーを確保できなかった。
- サポートしていない機種名または言語仕様が指定された。

例

TM-T88V 日本語モデル用の命令バッファを初期化する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"  
Lang: EPOS_OC_MODEL_JAPANESE];  
if ( builder != nil ) {  
    ... 処理 ...  
    [builder release];  
}
```

clearCommandBuffer

EposBuilder クラスの API で使用した命令バッファークリアします。
EposBuilder クラスに格納された命令バッファークリアは、本 API を実行するまで保管されます。

構文

- (int) **clearCommandBuffer**;

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。

例

命令バッファークリアする場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel:@"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus;
    ... 処理 ...
    errorStatus = [builder clearCommandBuffer];
    ... 処理 ...
    [builder release];
}
```

addTextAlign

位置揃え設定を命令バッファに追加します。



- 本 API の設定は、バーコード / 2 次元シンボルにも適用されます。
- ページモードで位置揃えを設定する場合、本 API ではなく、[addPagePosition \(85 ページ\)](#) で設定してください。

構文

```
- (int) addTextAlign: (int)align;
```

パラメーター

- align: 位置揃えを指定します。

設定値	説明
EPOS_OC_ALIGN_LEFT (初期値)	左揃え
EPOS_OC_ALIGN_CENTER	中央揃え
EPOS_OC_ALIGN_RIGHT	右揃え

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

中央揃えに設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextAlign: EPOS_OC_ALIGN_CENTER];
    ... 処理 ...
}
```


addTextLineSpace

改行量設定を命令バッファに追加します。

構文

– (int) **addTextLineSpace**: (long) linespc;

パラメーター

- linespc: 改行量 (ドット単位) を指定します。0 ~ 255 の整数値で指定します。
(初期値: [プリンター別サポート情報 \(149 ページ\)](#) を参照してください。)

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

改行量を 50 ドットに設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextLineSpace: 50];
    ... 処理 ...
}
```

addTextRotate

倒立印字設定を命令バッファに追加します。



- 本 API の設定は、バーコード / 2 次元シンボルにも適用されます。
- ページモードで倒立印字を設定する場合、本 API ではなく、[addPageDirection \(84 ページ\)](#) で設定してください。

構文

– (int) **addTextRotate**: (int) rotate;

パラメーター

- rotate: 倒立印字の有無を指定します。

設定値	説明
EPOS_OC_TRUE	倒立印字を指定
EPOS_OC_FALSE (初期値)	倒立印字を解除

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

倒立印字を設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextRotate: EPOS_OC_TRUE];
    ... 処理 ...
}
```

addText

文字の印字を命令バッファに追加します。



テキストの印字後、テキスト以外を印字する場合、改行または紙送りを実行してください。
(例: テキスト印字後、何もせずグラフィック印字を実行したが、印刷されない。)

構文

- (int) **addText**: (NSString *)data;

パラメーター

- data: 印字する文字列を指定します。
水平タブ / 改行は、以下のエスケープシーケンスを使用します。

文字列	説明
\t	水平タブ (HT)
\n	改行 (LF)
\\	バックスラッシュ

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

文字列を追加する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    ... 処理 ...
}
```

addTextLang

言語設定を命令バッファに追加します。本 API で指定された言語情報に従って、[addText \(51 ページ\)](#) で指定された文字列をエンコードします。[initWithPrinterModel \(46 ページ\)](#) で設定した言語仕様に合わせて指定してください。

構文

```
- (int) addTextLang: (int) lang;
```

パラメーター

- lang: 対象言語を指定します。

設定値	説明
EPOS_OC_LANG_EN(初期値)	英語 (ANK 仕様)
EPOS_OC_LANG_JA	日本語

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

日本語に設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextLang: EPOS_OC_LANG_JA];
    ... 処理 ...
}
```

addTextFont

文字のフォント設定を命令バッファに追加します。

構文

– (int) **addTextFont**: (int) font;

パラメーター

- font: フォントを指定します。

設定値	説明	TM プリンター別設定値						
		TM-P20	TM-P60II	TM-T20II	TM-T70	TM-T70II	TM-T88V	TM-T90II
EPOS_OC_FONT_A (初期値)	フォント A	○	○	○	○	○	○	○
EPOS_OC_FONT_B	フォント B	○	○	○	○	○	○	○
EPOS_OC_FONT_C	フォント C	○	○	-	-	-	-	○
EPOS_OC_FONT_D	フォント D	○	-	-	-	-	-	-
EPOS_OC_FONT_E	フォント E	○	-	-	-	-	-	-

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

フォント B を設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextFont: EPOS_OC_FONT_B];
    ... 処理 ...
}
```

addTextSmooth

スムージング設定を命令バッファに追加します。

構文

– (int) **addTextSmooth**: (int) smooth;

パラメーター

- smooth: スムージングの有無を指定します。

設定値	説明
EPOS_OC_TRUE	スムージングを指定
EPOS_OC_FALSE (初期値)	スムージングを解除

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

スムージングを有効に設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
             Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    . . . 処理 . . .
}
```

addTextDouble

文字の倍角設定を命令バッファーに追加します。

構文

– (int) **addTextDouble**: (int)dw Dh: (int)dh;


パラメーター

- dw: 文字の横倍角を指定します。

設定値	説明
EPOS_OC_TRUE	横倍角を指定
EPOS_OC_FALSE (初期値)	横倍角を解除
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- dh: 文字の縦倍角を指定します。

設定値	説明
EPOS_OC_TRUE	縦倍角を指定
EPOS_OC_FALSE (初期値)	縦倍角を解除
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない



dw と dh のパラメーターの両方を EPOS_OC_TRUE にした場合、4 倍角の文字が印字されます。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

4 倍角を設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextDouble: EPOS_OC_TRUE
        Dh: EPOS_OC_TRUE];
    ... 処理 ...
}
```

addTextSize

文字の倍率設定を命令バッファに追加します。

構文

– (int) **addTextSize**: (long)width Height: (long)height;

パラメーター

- width: 文字の横倍率を指定します。

設定値	説明
1 ~ 8 の整数	横方向の倍率を指定 (初期値: 1)
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- height: 文字の縦倍率を指定します。

設定値	説明
1 ~ 8 の整数	縦方向の倍率を指定 (初期値: 1)
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

横倍率 4 倍、縦倍率 4 倍に設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextSize: 4 Height: 4];
    ... 処理 ...
}
```


addTextStyle

文字の装飾設定を命令バッファーに追加します。

構文

```
(int) addTextStyle: (int)reverse Ul: (int)ul Em: (int)em  
Color: (int)color;
```

パラメーター

- reverse: 白黒反転文字を指定します。

設定値	説明
EPOS_OC_TRUE	白黒反転文字を指定
EPOS_OC_FALSE (初期値)	白黒反転文字を解除
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- ul: アンダーラインを指定します。

設定値	説明
EPOS_OC_TRUE	アンダーラインを指定
EPOS_OC_FALSE (初期値)	アンダーラインを解除
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- em: 太字を指定します。

設定値	説明
EPOS_OC_TRUE	太字を指定
EPOS_OC_FALSE (初期値)	太字を解除
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- color: 色を指定します。

設定値	説明
EPOS_OC_COLOR_NONE	非印字 (印字しない)
EPOS_OC_COLOR_1 (初期値)	第 1 色
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

アンダーラインを設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextStyle: EPOS_OC_PARAM_UNSPECIFIED
        U1: EPOS_OC_TRUE Em: EPOS_OC_PARAM_UNSPECIFIED
        Color: EPOS_OC_PARAM_UNSPECIFIED];
    . . . 処理 . . .
}
```

addTextPosition

横方向の印字開始位置を命令バッファーに追加します。



本 API 実行後、[addTextAlign \(48 ページ\)](#)、[addTextRotate \(50 ページ\)](#) は使用できません。

構文

– (int) **addTextPosition**: (long)x;

パラメーター

- x: 横方向の印字開始位置 (ドット単位) を指定します。
0 ~ 65535 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

印字位置を左端から 120 ドットの位置に設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextPosition: 120];
    ... 処理 ...
}
```

addFeedUnit

ドット単位の紙送りを命令バッファに追加します。

構文

– (int) **addFeedUnit**: (long) unit;

パラメーター

- unit: 紙送り量 (ドット単位) を指定します。0 ~ 255 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

紙送りを 30 ドットする場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addFeedUnit: 30];
    ... 処理 ...
}
```

addFeedLine

行単位の紙送りを命令バッファーに追加します。

構文

– (int) **addFeedLine**: (long) line;

パラメーター

- unit: 紙送り量（行単位）を指定します。0 ～ 255 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

紙送りを 3 行する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addFeedLine: 3];
    ... 処理 ...
}
```

addImage

ラスターイメージの印字を命令バッファに追加します。

UIImage クラスのグラフィックを印字します。

UIImage クラスのグラフィックうち、指定範囲を本 API の設定に従って、ラスターイメージデータに変換します。

また、画像を圧縮して送信することもできます。

画像の 1 ピクセルがプリンターの 1 ドットに相当します。透明色が含まれている場合、画像の背景を白とみなします。



- 画像圧縮は、*Bluetooth* インターフェイスの場合のみ設定してください。
- ラスターイメージを高速に印字する場合、[addTextAlign \(48 ページ\)](#) を EPOS_OC_ALIGN_LEFT に指定し、本 API の width パラメーターの値をプリンターの用紙幅を超えない 8 の倍数に指定してください。
- ページモードでは多階調印字をサポートしていません。スタンダードモードでのみ多階調グラフィックスの印字が可能です。
- ページモードでは画像圧縮をサポートしていません。

構文

```
- (int) addImage: (UIImage *)data X:(long)x Y:(long)y  
                  Width:(long)width Height:(long)height  
                  Color:(int)color Mode:(int)mode  
                  Halftone:(int)halftone  
                  Brightness:(double)brightness;  
                  Compress:(int)compress;
```

パラメーター

- data : UIImage クラスのインスタンスを指定します。
- x : 印字範囲の横方向の開始位置 (ピクセル単位) を指定します。
0 ~ 65534 の整数値で指定します。
- y : 印字範囲の縦方向の開始位置 (ピクセル単位) を指定します。
0 ~ 65534 の整数値で指定します。
- width : 印字範囲の幅 (ピクセル単位) を指定します。1 ~ 65535 の整数値で指定します。
- height : 印字範囲の高さ (ピクセル単位) を指定します。1 ~ 65535 の整数値で指定します。



x/y パラメーターと width/height パラメーターで指定された領域が data パラメーターで指定した画像のサイズに収まっていない場合、戻り値に EPOS_OC_ERR_PARAM が返されます。

- color : 色を指定します。

設定値	説明
EPOS_OC_COLOR_NONE	非印字 (印字しない)
EPOS_OC_COLOR_1	第 1 色
EPOS_OC_PARAM_DEFAULT	既定値 (第 1 色) を選択

- mode : カラーモードを指定します。

設定値	説明	TM プリンター別設定値						
		TM-P20	TM-P60II	TM-T20II	TM-T70	TM-T70II	TM-T88V	TM-T90II
EPOS_OC_MODE_MONO	モノクロ (2 階調)	○	○	○	○	○	○	○
EPOS_OC_MODE_GRAY16	多階調 (16 階調)	-	-	-	-	○	○	○
EPOS_OC_PARAM_DEFAULT	既定値を選択 (モノクロ (2 階調))	○	○	○	○	○	○	○

- halftone : ハーフトーン処理方法を指定します。

設定値	説明
EPOS_OC_HALFTONE_DITHER	ディザー (グラフィックの印刷に適しています。)
EPOS_OC_HALFTONE_ERROR_DIFFUSION	誤差拡散 (文字とグラフィックが混在する印刷に適しています。)
EPOS_OC_HALFTONE_THRESHOLD	しきい値 (文字の印刷に適しています。)
EPOS_OC_PARAM_DEFAULT	既定値 (ディザー) を選択



多階調 (16 階調) の場合、無視されます。

- brightness : 明るさの補正値を指定します。

設定値	説明
0.1 ~ 10.0 の実数	明るさ補正値 (ガンマー値)
EPOS_OC_PARAM_DEFAULT	既定値 (1.0) を選択



1.0 以外を指定した場合、印字速度が遅くなります。

- compress : 画像圧縮を指定します。EPOS_OC_COMPRESS_DEFLATE は、Bluetooth インターフェースの場合のみ指定してください。

設定値	説明	TM プリンター別設定値						
		TM-P20	TM-P60II	TM-T20II	TM-T70	TM-T70II	TM-T88V	TM-T90II
EPOS_OC_COMPRESS_DEFLATE	画像圧縮します	○	-	○	-	○	○	-
EPOS_OC_COMPRESS_NONE	画像圧縮しません	○	○	○	○	○	○	○
EPOS_OC_PARAM_DEFAULT	既定値 (画像圧縮しません)	○	○	○	○	○	○	○



TCP 通信する場合、EPOS_OC_PARAM_DEFAULT を設定してください。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

```
UIImage * imageData = Nil;


id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    ... 処理 ...
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
        Height: 256 Color: EPOS_OC_PARAM_DEFAULT Mode: EPOS_OC_MODE_MONO
        Halftone: EPOS_OC_HALFTONE_DITHER Brightness: 1.0
        Compress: EPOS_OC_COMPRESS_NONE];
    ... 処理 ...
}
```


addImage (旧フォーマット)

ラスターイメージの印字を命令バッファーに追加します。Bluetooth インターフェイス使用時、画像圧縮印刷ができないため、白筋が入る場合があります。

UIImage クラスのグラフィックを印字します。

UIImage クラスのグラフィックうち、指定範囲を本 API の設定に従って、ラスターイメージデータに変換します。画像の 1 ピクセルがプリンターの 1 ドットに相当します。透明色が含まれている場合、画像の背景を白とみなします。




- ラスターイメージを高速に印字する場合、addTextAlign (48 ページ) を EPOS_OC_ALIGN_LEFT に指定し、本 API の width パラメーターの値をプリンターの用紙幅を超えない 8 の倍数に指定してください。
- ページモードでは多階調印字をサポートしていません。スタンダードモードでのみ多階調グラフィックスの印字が可能です。

構文

```
- (int) addImage:(UIImage *)data X:(long)x Y:(long)y
    Width:(long)width Height:(long)height
    Color:(int)color Mode:(int)mode
    Halftone:(int)halftone
    Brightness:(double)brightness;
```

パラメーター

- data : UIImage クラスのインスタンスを指定します。
- x : 印字範囲の横方向の開始位置 (ピクセル単位) を指定します。
0 ~ 65534 の整数値で指定します。
- y : 印字範囲の縦方向の開始位置 (ピクセル単位) を指定します。
0 ~ 65534 の整数値で指定します。
- width : 印字範囲の幅 (ピクセル単位) を指定します。1 ~ 65535 の整数値で指定します。
- height : 印字範囲の高さ (ピクセル単位) を指定します。1 ~ 65535 の整数値で指定します。



x/y パラメーターと width/height パラメーターで指定された領域が data パラメーターで指定した画像のサイズに収まっていない場合、戻り値に EPOS_OC_ERR_PARAM が返されます。

- color : 色を指定します。

設定値	説明
EPOS_OC_COLOR_NONE	非印字 (印刷しない)
EPOS_OC_COLOR_1	第 1 色
EPOS_OC_PARAM_DEFAULT	既定値 (第 1 色) を選択

- mode : カラーモードを指定します。

設定値	説明	TM プリンター別設定値						
		TM-P20	TM-P60II	TM-T20II	TM-T70	TM-T70II	TM-T88V	TM-T90II
EPOS_OC_MODE_MONO	モノクロ (2 階調)	○	○	○	○	○	○	○
EPOS_OC_MODE_GRAY16	多階調 (16 階調)	-	-	-	-	○	○	○
EPOS_OC_PARAM_DEFAULT	既定値を選択 (モノクロ (2 階調))	○	○	○	○	○	○	○

- halftone : ハーフトーン処理方法を指定します。

設定値	説明
EPOS_OC_HALFTONE_DITHER	ディザー (グラフィックの印刷に適しています。)
EPOS_OC_HALFTONE_ERROR_DIFFUSION	誤差拡散 (文字とグラフィックが混在する印刷に適しています。)
EPOS_OC_HALFTONE_THRESHOLD	しきい値 (文字の印刷に適しています。)
EPOS_OC_PARAM_DEFAULT	既定値 (ディザー) を選択



多階調 (16 階調) の場合、無視されます。

- brightness : 明るさの補正値を指定します。

設定値	説明
0.1 ~ 10.0 の実数	明るさ補正値 (ガンマー値)
EPOS_OC_PARAM_DEFAULT	既定値 (1.0) を選択



1.0 以外を指定した場合、印字速度が遅くなります。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

```

UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    . . . 処理 . . .
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
Height: 256 Color: EPOS_OC_PARAM_DEFAULT Mode: EPOS_OC_MODE_MONO
Halftone: EPOS_OC_HALFTONE_DITHER Brightness: 1.0];
    . . . 処理 . . .
}

```

ページモードで幅 256 ドット、高さ 256 ドットの画像を印字する

```

UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    . . . 処理 . . .
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPagePosition: 0 Y: 255];
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
Height: 256 Color: EPOS_OC_PARAM_DEFAULT Mode: EPOS_OC_MODE_MONO
Halftone: EPOS_OC_HALFTONE_DITHER Brightness: 1.0];
    errorStatus = [builder addPageEnd];
    . . . 処理 . . .
}

```

addImage (旧フォーマット)

ラスターイメージの印字を命令バッファに追加します。多階調印刷はできません。

Bluetooth 接続時、画像圧縮印刷ができないため、白筋が入る場合があります。

UIImage クラスのグラフィックを印字します。

UIImage クラスのグラフィックうち、指定範囲をディザ処理で二値化し、ラスターイメージデータに変換します。画像の1ピクセルがプリンターの1ドットに相当します。透明色が含まれている場合、画像の背景を白とみなします。



- 多階調で印字する場合、[addImage \(62 ページ\)](#) を使用してください。
- ラスターイメージを高速に印字する場合、[addTextAlign \(48 ページ\)](#) を EPOS_OC_ALIGN_LEFT に指定し、本 API の width パラメーターの値をプリンターの用紙幅を超えない 8 の倍数に指定してください。

構文

```
(int) addImage: (UIImage *)data X:(long)x Y:(long)y  
Width:(long)width Height:(long)height  
Color:(int)color;
```

パラメーター

- data : UIImage クラスのインスタンスを指定します。
- x : 印字範囲の横方向の開始位置 (ピクセル単位) を指定します。
0 ~ 65534 の整数値で指定します。
- y : 印字範囲の縦方向の開始位置 (ピクセル単位) を指定します。
0 ~ 65534 の整数値で指定します。
- width : 印字範囲の幅 (ピクセル単位) を指定します。1 ~ 65535 の整数値で指定します。
- height : 印字範囲の高さ (ピクセル単位) を指定します。1 ~ 65535 の整数値で指定します。



x/y パラメーターと width/height パラメーターで指定された領域が data パラメーターで指定した画像のサイズに収まっていない場合、戻り値に EPOS_OC_ERR_PARAM が返されます。

- color : 色を指定します。

設定値	説明
EPOS_OC_COLOR_NONE	非印字 (印刷しない)
EPOS_OC_COLOR_1	第 1 色
EPOS_OC_PARAM_DEFAULT	既定値 (第 1 色) を選択

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

```

UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    . . . 処理 . . .
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
    Height: 256 Color: EPOS_OC_PARAM_DEFAULT];
    . . . 処理 . . .
}

```

ページモードで幅 256 ドット、高さ 256 ドットの画像を印字する

```

UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    . . . 処理 . . .
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPagePosition: 0 Y: 255];
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
    Height: 256 Color: EPOS_OC_PARAM_DEFAULT];
    errorStatus = [builder addPageEnd];
    . . . 処理 . . .
}

```

addLogo

NV ロゴの印字を命令バッファに追加します。プリンターの NV メモリーに登録されているロゴを印字します。ページモードでは多階調印字をサポートしていません。スタンダードモードでのみ多階調グラフィックスの印字が可能です。



- ロゴは以下のユーティリティを使って、あらかじめプリンターにロゴの登録します。
 - * 機種専用ユーティリティ
 - * ロゴ登録ユーティリティ (TMFLogo)
- ページモードでは多階調印字をサポートしていません。スタンダードモードでのみ多階調グラフィックスの印字が可能です。

構文

- (int) **addLogo**: (long)key1 Key2: (long)key2;

パラメーター

- key1: NV ロゴのキーコード 1 を指定します。32 ~ 126 の整数値で指定します。
- key2: NV ロゴのキーコード 2 を指定します。32 ~ 126 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

キーコード 48,48 の NV ロゴを印字する

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addLogo: 48 Key2: 48];
    ... 処理 ...
}
```

addBarcode


バーコード印字を命令バッファに追加します。

構文

```
- (int) addBarcode: (NSString *)data Type:(int)type
    Hri:(int)hri Font:(int)font
    Width:(long)width
    Height:(long)height;
```

パラメーター

- data : バーコードデータを文字列で指定します。



type で指定するバーコードの規格に従った文字列を指定してください。規格に従っていない場合、バーコードは印刷されません。

種類	説明
UPC-A	11 桁の数字を指定した場合、チェックデジットを自動で付加します。 12 桁の数字を指定した場合、12 桁目をチェックデジットとして処理しますが、チェックデジットの検算は行いません。
UPC-E	最初の桁に 0 を指定してください。 2 ～ 6 桁目にメーカーコードを指定してください。 7 ～ 11 桁目にアイテムコードを右詰めで指定してください。アイテムコードの桁数はメーカーコードにより異なります。使用しない桁は 0 を指定してください。 11 桁の数字を指定した場合、チェックデジットを自動で付加します。 12 桁の数字を指定した場合、12 桁目をチェックデジットとして処理しますが、チェックデジットの検算は行いません。
EAN13	12 桁の数字を指定した場合、チェックデジットを自動で付加します。 13 桁の数字を指定した場合、13 桁目をチェックデジットとして処理しますが、チェックデジットの検算は行いません。
JAN13	
EAN8	7 桁の数字を指定した場合、チェックデジットを自動で付加します。 8 桁の数字を指定した場合、8 桁目をチェックデジットとして処理しますが、チェックデジットの検算は行いません。
JAN8	
CODE39	先頭の文字が * の場合、この文字をスタートキャラクターとして処理します。それ以外の場合、スタートキャラクターを自動で付加します。
ITF	スタートコードおよびストップコードを自動で付加します。 チェックデジットの付加および検算は行いません。

種類	説明																		
CODABAR	<p>スタートキャラクター (A ~ D, a ~ d) を指定してください。</p> <p>ストップキャラクター (A ~ D, a ~ d) を指定してください。</p> <p>チェックデジットの付加および検算は行いません。</p>																		
CODE93	<p>スタートキャラクターおよびストップキャラクターを自動で付加します。</p> <p>チェックデジットを計算して自動で付加します。</p>																		
CODE128	<p>スタートキャラクター (CODE A, CODE B, CODE C) を指定してください。</p> <p>ストップキャラクターを自動で付加します。</p> <p>チェックデジットを計算して自動で付加します。</p> <p>以下の文字をエンコードするには、文字 { で始まる 2 文字を指定してください。</p> <table> <tr><td>FNC1:</td><td>{1</td></tr> <tr><td>FNC2:</td><td>{2</td></tr> <tr><td>FNC3:</td><td>{3</td></tr> <tr><td>FNC4:</td><td>{4</td></tr> <tr><td>CODE A:</td><td>{A</td></tr> <tr><td>CODE B:</td><td>{B</td></tr> <tr><td>CODE C:</td><td>{C</td></tr> <tr><td>SHIFT:</td><td>{S</td></tr> <tr><td>{:</td><td>{{</td></tr> </table>	FNC1:	{1	FNC2:	{2	FNC3:	{3	FNC4:	{4	CODE A:	{A	CODE B:	{B	CODE C:	{C	SHIFT:	{S	{:	{{
FNC1:	{1																		
FNC2:	{2																		
FNC3:	{3																		
FNC4:	{4																		
CODE A:	{A																		
CODE B:	{B																		
CODE C:	{C																		
SHIFT:	{S																		
{:	{{																		
GS1-128	<p>スタートキャラクター、FNC1、チェックデジット、ストップキャラクターを自動で付加します。</p> <p>アプリケーション識別子 (AI) とそれに続くデータのチェックデジットを計算して自動で付加するには、チェックデジットの位置に文字 * を指定します。</p> <p>アプリケーション識別子 (AI) を括弧で囲むことができます。括弧は HRI の印字文字として使用し、データとしてエンコードしません。</p> <p>アプリケーション識別子 (AI) とデータの間に空白を挿入することができます。空白は HRI の印字文字として使用し、データとしてエンコードしません。</p> <p>以下の文字をエンコードするには、文字 { で始まる 2 文字を指定してください。</p> <table> <tr><td>FNC1:</td><td>{1</td></tr> <tr><td>FNC3:</td><td>{3</td></tr> <tr><td>(:</td><td>{{</td></tr> <tr><td>):</td><td>{}</td></tr> <tr><td>*:</td><td>{*</td></tr> <tr><td>{:</td><td>{{</td></tr> </table>	FNC1:	{1	FNC3:	{3	(:	{{):	{}	*:	{*	{:	{{						
FNC1:	{1																		
FNC3:	{3																		
(:	{{																		
):	{}																		
:	{																		
{:	{{																		
GS1 DataBar Omnidirectional	アプリケーション識別子 (AI) とチェックデジットを除く 13 桁の商品識別番号 (GTIN) を指定してください。																		
GS1 DataBar Truncated																			
GS1 DataBar Limited																			

種類	説明
GS1 DataBar Expanded	<p>アプリケーション識別子 (AI) を括弧で囲むことができます。括弧は HRI の印字文字として使用し、データとしてエンコードしません。</p> <p>以下の文字をエンコードするには、文字 { で始まる 2 文字を指定してください。</p> <p>FNC1: {1</p> <p>(: {(:</p> <p>): {)}</p>

文字列で表現できないバイナリーデータを指定する場合、以下のエスケープシーケンスで指定します。

文字列	説明
\xnn	コントロールコード
\\	バックスラッシュ

- type : バーコードの種類を指定します。

設定値	説明
EPOS_OC_BARCODE_UPC_A	UPC-A
EPOS_OC_BARCODE_UPC_E	UPC-E
EPOS_OC_BARCODE_EAN13	EAN13
EPOS_OC_BARCODE_JAN13	JAN13
EPOS_OC_BARCODE_EAN8	EAN8
EPOS_OC_BARCODE_JAN8	JAN8
EPOS_OC_BARCODE_CODE39	CODE39
EPOS_OC_BARCODE_ITF	ITF
EPOS_OC_BARCODE_CODABAR	CODABAR
EPOS_OC_BARCODE_CODE93	CODE93
EPOS_OC_BARCODE_CODE128	CODE128
EPOS_OC_BARCODE_GS1_128	GS1-128
EPOS_OC_BARCODE_GS1_DATABAR_OMNIDIRECTIONAL	GS1 DataBar Omnidirectional
EPOS_OC_BARCODE_GS1_DATABAR_TRUNCATED	GS1 DataBar Truncated
EPOS_OC_BARCODE_GS1_DATABAR_LIMITED	GS1 DataBar Limited
EPOS_OC_BARCODE_GS1_DATABAR_EXPANDED	GS1 DataBar Expanded

- hri : HRI の位置を指定します。

設定値	説明
EPOS_OC_HRI_NONE (初期値)	印字しない
EPOS_OC_HRI_ABOVE	バーコードの上
EPOS_OC_HRI_BELOW	バーコードの下
EPOS_OC_HRI_BOTH	バーコードの上と下の両方
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- font : HRI フォントを指定します。

設定値	説明
EPOS_OC_FONT_A (初期値)	フォント A
EPOS_OC_FONT_B	フォント B
EPOS_OC_FONT_C	フォント C
EPOS_OC_FONT_D	フォント D
EPOS_OC_FONT_E	フォント E
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- width : 1 モジュールの幅をドット単位で指定します。

設定値	説明
2 ～ 6 の整数値	1 モジュールの幅 (ドット単位)
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- height : バーコードの高さをドット単位で指定します。1 ～ 255 の整数値で指定します。

設定値	説明
1 ～ 255 の整数値	バーコードの高さ (ドット単位)
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

各種バーコードを印字する場合

```

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addBarcode: @"01234567890"
                                Type: EPOS_OC_BARCODE_UPC_A Hri: EPOS_OC_HRI_BELOW
                                Font: EPOS_OC_PARAM_UNSPECIFIED Width: 2 Height: 64];
    errorStatus = [builder addBarcode: @"01234500005"
                                Type: EPOS_OC_BARCODE_UPC_E Hri: EPOS_OC_PARAM_UNSPECIFIED
                                Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                                Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"201234567890"
                                Type: EPOS_OC_BARCODE_EAN13 Hri: EPOS_OC_PARAM_UNSPECIFIED
                                Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                                Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"201234567890"
                                Type: EPOS_OC_BARCODE_JAN13 Hri: EPOS_OC_PARAM_UNSPECIFIED
                                Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                                Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"2012345" Type: EPOS_OC_BARCODE_EAN8
                                Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"2012345" Type: EPOS_OC_BARCODE_JAN8
                                Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"ABCDE" Type: EPOS_OC_BARCODE_CODE39
                                Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"012345" Type: EPOS_OC_BARCODE_ITF
                                Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"A012345A"
                                Type: EPOS_OC_BARCODE_CODABAR Hri: EPOS_OC_PARAM_UNSPECIFIED
                                Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                                Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"ABCDE" Type: EPOS_OC_BARCODE_CODE93
                                Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"{Babcde"
                                Type: EPOS_OC_BARCODE_CODE128 Hri: EPOS_OC_PARAM_UNSPECIFIED
                                Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                                Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"(01)201234567890*"
                                Type: EPOS_OC_BARCODE_GS1_128 Hri: EPOS_OC_PARAM_UNSPECIFIED
                                Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                                Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"0201234567890"
                                Type: EPOS_OC_BARCODE_GS1_DATABAR_OMNIDIRECTIONAL
                                Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"0201234567890"
                                Type: EPOS_OC_BARCODE_GS1_DATABAR_TRUNCATED
                                Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"0201234567890"
                                Type: EPOS_OC_BARCODE_GS1_DATABAR_LIMITED
                                Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"(01)2012345678903"
                                Type: EPOS_OC_BARCODE_GS1_DATABAR_EXPANDED
                                Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    ... 処理 ...
}

```

addSymbol

2次元シンボル印字を命令バッファに追加します。

構文

```
(int) addSymbol:(NSString *)data Type:(int)type  
Level:(int)level Width:(long)width  
Height:(long)height Size:(long)size;
```

パラメーター

- data : 2次元シンボルデータを文字列で指定します。



type で指定する 2次元シンボルの規格に従った文字列を指定してください。規格に従っていない場合、2次元シンボルは印刷されません。

文字列	説明
Standard PDF417	文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。 データ領域の最大コードワード数は 928 個、1 段あたりのデータ領域の最大コードワード数は 30 個、最大段数は 90 段です。
Truncated PDF417	
QR Code Model 1	文字列をシフト JIS に変換後、エスケープシーケンスの処理を行い、データの種別を以下の中から選択してエンコードします。 数字： 0 ～ 9 英数字： 0 ～ 9, A ～ Z, スペース, \$, %, *, +, -, ., /, : 漢字： シフト JIS 値 8 ビットバイトデータ： 0x00 ～ 0xff
QR Code Model 2	

文字列	説明
MaxiCode Mode 2	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>モード 2 およびモード 3 の場合、最初のデータが <code>()>\x1e01\x1dyy</code> (<code>yy</code> は 2 桁の数字) の場合、これをメッセージヘッダーとして処理し、次のデータからプライマリメッセージとして処理します。それ以外の場合、最初のデータからプライマリメッセージとして処理します。</p> <p>モード 2 の場合、プライマリメッセージを以下の形式で指定してください。</p> <p>郵便コード (1 ~ 9 桁の数字) <code>GS:(\x1d)</code> ISO 国名コード (1 ~ 3 桁の数字) <code>GS:(\x1d)</code> サービスクラスコード (1 ~ 3 桁の数字)</p> <p>モード 3 の場合、プライマリメッセージを以下の形式で指定してください。</p> <p>郵便コード (1 ~ 6 個のコードセット A で変換可能なデータ) <code>GS(\x1d)</code> ISO 国名コード (1 ~ 3 桁の数字) <code>GS(\x1d)</code> サービスクラスコード (1 ~ 3 桁の数字)</p>
MaxiCode Mode 3	
MaxiCode Mode 4	
MaxiCode Mode 5	
MaxiCode Mode 6	
GS1 DataBar Stacked	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>アプリケーション識別子 (AI) とチェックデジットを除く 13 桁の商品識別番号 (GTIN) を指定してください。</p>
GS1 DataBar Stacked Omnidirectional	
GS1 DataBar Expanded Stacked	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>アプリケーション識別子 (AI) を括弧で囲むことができます。括弧は HRI の印字文字として使用し、データとしてエンコードしません。</p> <p>以下の文字をエンコードするには、文字 { で始まる 2 文字を指定してください。</p> <p>FNC1: {1 (: {(): }</p>
Aztec Code Full-Range モード	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>最大でテキスト 3067 文字、数字 3832 文字、バイナリーデータ 1914 バイトを指定できます。</p>
Aztec Code Compact モード	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>最大でテキスト 89 文字、数字 110 文字、バイナリーデータ 53 バイトを指定できます。</p>

文字列	説明
DataMatrix 正方形	文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。 シンボルは 10 行 x10 列～ 144 行 x144 列の正方形、または行数 8、行数 12、行数 16 の長方形です。 最大で英数字 2335 文字、数字 3116 文字、バイナリーデータ 1556 バイトを指定できます。
DataMatrix 長方形、行数 8	
DataMatrix 長方形、行数 12	
DataMatrix 長方形、行数 16	

文字列で表現できないバイナリーデータを指定する場合、以下のエスケープシーケンスで指定します。

文字列	説明
\xnn	コントロールコード
\\	バックスラッシュ

- type : 2次元シンボルの種類を指定します。

設定値	種類
EPOS_OC_SYMBOL_PDF417_STANDARD	Standard PDF417
EPOS_OC_SYMBOL_PDF417_TRUNCATED	Truncated PDF417
EPOS_OC_SYMBOL_QRCODE_MODEL_1	QR Code Model 1
EPOS_OC_SYMBOL_QRCODE_MODEL_2	QR Code Model 2
EPOS_OC_SYMBOL_MAXICODE_MODE_2	MaxiCode Mode 2
EPOS_OC_SYMBOL_MAXICODE_MODE_3	MaxiCode Mode 3
EPOS_OC_SYMBOL_MAXICODE_MODE_4	MaxiCode Mode 4
EPOS_OC_SYMBOL_MAXICODE_MODE_5	MaxiCode Mode 5
EPOS_OC_SYMBOL_MAXICODE_MODE_6	MaxiCode Mode 6
EPOS_OC_SYMBOL_GS1_DATABAR_STACKED	GS1 DataBar Stacked
EPOS_OC_SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL	GS1 DataBar Stacked Omnidirectional
EPOS_OC_SYMBOL_GS1_DATABAR_EXPANDED_STACKED	GS1 DataBar Expanded Stacked
EPOS_OC_SYMBOL_AZTECCODE_FULLRANGE	Aztec Code Full-Range モード
EPOS_OC_SYMBOL_AZTECCODE_COMPACT	Aztec Code Compact モード
EPOS_OC_SYMBOL_DATAMATRIX_SQUARE	DataMatrix 正方形
EPOS_OC_SYMBOL_DATAMATRIX_RECTANGLE_8	DataMatrix 長方形、行数 8
EPOS_OC_SYMBOL_DATAMATRIX_RECTANGLE_12	DataMatrix 長方形、行数 12
EPOS_OC_SYMBOL_DATAMATRIX_RECTANGLE_16	DataMatrix 長方形、行数 16

- level : エラー訂正レベルを指定します。

設定値	説明
EPOS_OC_LEVEL_0	PDF417 エラー訂正レベル 0
EPOS_OC_LEVEL_1	PDF417 エラー訂正レベル 1
EPOS_OC_LEVEL_2	PDF417 エラー訂正レベル 2
EPOS_OC_LEVEL_3	PDF417 エラー訂正レベル 3
EPOS_OC_LEVEL_4	PDF417 エラー訂正レベル 4
EPOS_OC_LEVEL_5	PDF417 エラー訂正レベル 5
EPOS_OC_LEVEL_6	PDF417 エラー訂正レベル 6
EPOS_OC_LEVEL_7	PDF417 エラー訂正レベル 7
EPOS_OC_LEVEL_8	PDF417 エラー訂正レベル 8
EPOS_OC_LEVEL_L	QR Code エラー訂正レベル L
EPOS_OC_LEVEL_M	QR Code エラー訂正レベル M
EPOS_OC_LEVEL_Q	QR Code エラー訂正レベル Q
EPOS_OC_LEVEL_H	QR Code エラー訂正レベル H
5 ~ 95 の整数	Aztec Code エラー訂正レベル (パーセント単位)
EPOS_OC_LEVEL_DEFAULT	既定レベル
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない



- 2次元シボルの種類に合せて選択してください。
- MaxiCode/2次元GS1 DataBar//DataMatrixの場合、EPOS_OC_LEVEL_DEFAULTを選択してください。

- width : モジュールの幅を指定します。

設定値	説明
1 ~ 255 の整数値	モジュールの幅
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない



MaxiCodeは無視されます。

- height : モジュールの高さを指定します。

設定値	説明
1 ~ 255 の整数値	モジュールの高さ
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない



QR Code/MaxiCode/2次元GS1 DataBar/Aztec Code/DataMatrixは無視されます。

- size : 2次元シボルの最大サイズを指定します。

設定値	説明
0 ~ 65535 の整数値	2次元シボルの最大サイズ
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない



QR Code/MaxiCode/Aztec Code/DataMatrixは無視されます。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。


例

各種 2 次元シンボルを印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addSymbol: @"ABCDE"
                                Type: EPOS_OC_SYMBOL_PDF417_STANDARD Level:
                                EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                                Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addSymbol: @"ABCDE"
                                Type: EPOS_OC_SYMBOL_QRCODE_MODEL_2 Level: EPOS_OC_LEVEL_Q
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED
                                Size: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addSymbol: @"908063840\\x1d850\\x1d001\\x1d\\x04"
                                Type: EPOS_OC_SYMBOL_MAXICODE_MODE_2 Level: EPOS_OC_PARAM_UNSPECIFIED
                                Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED
                                Size: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addSymbol: @"0201234567890"
                                Type: EPOS_OC_SYMBOL_GS1_DATABAR_STACKED
                                Level: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                                Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addSymbol: @"0201234567890"
                                Type: EPOS_OC_SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL
                                Level: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                                Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addSymbol: @"(01)02012345678903"
                                Type: EPOS_OC_SYMBOL_GS1_DATABAR_EXPANDED_STACKED
                                Level: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                                Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
    ... 処理 ...
}
```


addPageBegin

ページモード開始を命令バッファーに追加します。ページモードの処理が開始します。

 本 API は [addPageEnd \(82 ページ\)](#) と一緒にお使いください。

構文

- (int) **addPageBegin**;

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

ページモードで文字「ABCDE」を印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    ... 処理 ...
}
```

addPageEnd

ページモード終了を命令バッファに追加します。ページモードの処理が終了します。



本 API は [addPageBegin \(81 ページ\)](#) と一緒にお使いください。

構文

- (int) **addPageEnd**;

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。


例

ページモードで文字「ABCDE」を印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    . . . 処理 . . .
}
```

addPageArea

ページモード印字領域を命令バッファに追加します。
ページモード印字領域（座標）を指定します。本 API に続けて、addText など印刷データの API を指定します。




- 印字内容に合わせて印字領域を指定してください。印字データが印字領域をはみ出した場合、印字データが途中で切れた印字結果になります。
- 本 API は [addPageBegin \(81 ページ\)](#) と [addPageEnd \(82 ページ\)](#) に挟んでお使いください。

構文

- (int) **addPageArea**: (long)x Y: (long)y Width: (long)width
Height: (long)height;

パラメーター

- x: 横方向の原点（ドット単位）を指定します。0 ～ 65535 の整数値で指定します。
0 はプリンターの印字可能領域の左端になります。
- y: 縦方向の原点（ドット単位）を指定します。0 ～ 65535 の整数値で指定します。
0 は紙送りをしていない位置です。
- width: 印字領域の幅（ドット単位）を指定します。1 ～ 65535 の整数値で指定します。
- height: 印字領域の高さ（ドット単位）を指定します。1 ～ 65535 の整数値で指定します。



印字領域の幅と高さは、印字方向の設定に合わせて確定してください。
印字データが切れてしまう場合があります。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

原点 (100, 50)、幅 200 ドット、高さ 30 ドットの印字領域を指定して、文字「ABCDE」を印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 200 Height: 30];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    ... 処理 ...
}
```

addPageDirection

ページモード印字方向設定を命令バッファーに追加します。ページモードの印字方向を指定します。回転させない場合は、省略できます。



本 API は [addPageBegin \(81 ページ\)](#) と [addPageEnd \(82 ページ\)](#) に挟んでお使いください。

構文

– (int) **addPageDirection**: (int) dir;

パラメーター

- dir: ページモードの印字方向を指定します。

設定値	説明
EPOS_OC_DIRECTION_LEFT_TO_RIGHT (初期値)	回転しない (左上を始点に右方向へ印字)
EPOS_OC_DIRECTION_BOTTOM_TO_TOP	反時計回り 90 度回転 (左下を始点に上方向へ印字)
EPOS_OC_DIRECTION_RIGHT_TO_LEFT	180 度回転 (右下を始点に左方向へ印字)
EPOS_OC_DIRECTION_TOP_TO_BOTTOM	時計回り 90 度回転 (右上を始点に下方向へ印字)

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。


例

時計回りに 90 度回転させて、文字「ABCDE」を印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 30 Height: 200];
    errorStatus = [builder addPageDirection:
                  EPOS_OC_DIRECTION_TOP_TO_BOTTOM];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    ... 処理 ...
}
```

addPagePosition

ページモードの印字位置設定領域を命令バッファに追加します。
addPageArea で指定したエリア内での、印字開始位置（座標）を指定します。




本 API は [addPageBegin \(81 ページ\)](#) と [addPageEnd \(82 ページ\)](#) に挟んでお使いください。

構文

– (int) **addPagePosition**: (long)x Y: (long)y;

パラメーター

- x: 横方向の印字位置（ドット単位）を指定します。0 ～ 65535 の整数値で指定します。
- y: 縦方向の印字位置（ドット単位）を指定します。0 ～ 65535 の整数値で指定します。



印字開始位置（座標）は、印字内容に合わせて指定してください。以下を参考にしてください。

- * 文字列を印字する場合
最初の文字のベースライン左端を指定します。
標準の大きさかつ左詰めで印字する場合は省略可能です。高さが 2 倍の文字を印刷する場合は、y を 42 以上に指定します。
- * バーコードを印字する場合
シンボルの左下を指定します。y にバーコードの高さを指定してください。
- * グラフィック / ロゴを印字する場合
グラフィックデータの左下を指定します。y にグラフィックデータの高さを指定してください。
- * 2 次元シンボルを印字する場合
シンボルの左上を指定します。左上から印字する場合は、省略可能です。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

addPageArea で指定したエリア内の印字開始位置を (50, 30) に指定して、文字「ABCDE」を印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 200 Height: 100];
    errorStatus = [builder addPagePosition: 50 Y: 30];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    ... 処理 ...
}
```

addPageLine

ページモードの直線描画を命令バッファに追加します。ページモードで直線を描画します。



- 斜線は描画できません。
- 本 API は [addPageBegin \(81 ページ\)](#) と [addPageEnd \(82 ページ\)](#) に挟んでお使いください。

構文

```
(int) addPageLine:(long)x1 Y1:(long)y1 X2:(long)x2  
Y2:(long)y2 Style:(int)style;
```

パラメーター

- x1: 横方向の描画開始位置 (ドット単位) を指定します。0 ～ 65535 の整数値で指定します。
- y1: 縦方向の描画開始位置 (ドット単位) を指定します。0 ～ 65535 の整数値で指定します。
- x2: 横方向の描画終了位置 (ドット単位) を指定します。0 ～ 65535 の整数値で指定します。
- y2: 縦方向の描画終了位置 (ドット単位) を指定します。0 ～ 65535 の整数値で指定します。
- style: 罫線の種類を指定します。

設定値	説明
EPOS_OC_LINE_THIN	実線：細
EPOS_OC_LINE_MEDIUM	実線：中太
EPOS_OC_LINE_THICK	実線：太
EPOS_OC_LINE_THIN_DOUBLE	二重線：細
EPOS_OC_LINE_MEDIUM_DOUBLE	二重線：中太
EPOS_OC_LINE_THICK_DOUBLE	二重線：太
EPOS_OC_PARAM_DEFAULT	実線：細

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

開始位置 (100,0), 終了位置 (500,0) を頂点とする直線を、細い実線で描画する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60II"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageLine: 100 Y1: 0 X2: 500 Y2: 0
                                   Style: EPOS_OC_LINE_THIN];
    errorStatus = [builder addPageEnd];
    . . . 処理 . . .
}
```

addPageRectangle

ページモードの四角形描画を命令バッファに追加します。ページモードで四角形を描画します。



本 API は [addPageBegin \(81 ページ\)](#) と [addPageEnd \(82 ページ\)](#) に挟んでお使いください。

構文

```
- (int) addPageRectangle: (long)x1 Y1: (long)y1  
                        X2: (long)x2 Y2: (long)y2  
                        Style: (int)style;
```

パラメーター

- x1: 横方向の描画開始位置 (ドット単位) を指定します。0 ～ 65535 の整数値で指定します。
- y1: 縦方向の描画開始位置 (ドット単位) を指定します。0 ～ 65535 の整数値で指定します。
- x2: 横方向の描画終了位置 (ドット単位) を指定します。0 ～ 65535 の整数値で指定します。
- y2: 縦方向の描画終了位置 (ドット単位) を指定します。0 ～ 65535 の整数値で指定します。
- style: 罫線の種類を指定します。

設定値	説明
EPOS_OC_LINE_THIN	実線：細
EPOS_OC_LINE_MEDIUM	実線：中太
EPOS_OC_LINE_THICK	実線：太
EPOS_OC_LINE_THIN_DOUBLE	二重線：細
EPOS_OC_LINE_MEDIUM_DOUBLE	二重線：中太
EPOS_OC_LINE_THICK_DOUBLE	二重線：太
EPOS_OC_PARAM_DEFAULT	実線：細

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

開始位置 (100,0), 終了位置 (500,200) を頂点とする四角形を、細い二重線で描画する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60II"  
              Lang: EPOS_OC_MODEL_JAPANESE];  
if ( builder != nil ) {  
    int errorStatus = EPOS_OC_SUCCESS;  
    errorStatus = [builder addPageBegin];  
    errorStatus = [builder addPageRectangle: 100 Y1: 0 X2: 500 Y2: 200  
                                Style: EPOS_OC_LINE_THIN_DOUBLE];  
    errorStatus = [builder addPageEnd];  
    ... 処理 ...  
}
```


addCut

用紙カットを命令バッファに追加します。用紙カットを設定します。



ページモードでは使用できません。

構文

- (int) **addCut**: (int) type;

パラメーター

- type: 用紙カット方法を指定します。

設定値	説明
EPOS_OC_CUT_NO_FEED	フィードなしカット (紙送りせずにカット)
EPOS_OC_CUT_FEED	フィードカット (紙送り後カット)
EPOS_OC_CUT_RESERVE	カット予約 (後に続く印字を実行後、カット位置でカット)
EPOS_OC_PARAM_DEFAULT	フィードカット (紙送り後カット)

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

フィードカットする場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"  
             Lang: EPOS_OC_MODEL_JAPANESE];  
if ( builder != nil ) {  
    int errorStatus = EPOS_OC_SUCCESS;  
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];  
    ... 処理 ...  
}
```

addPulse

ドロアーキックを命令バッファに追加します。ドロアーキックを設定します。



- ページモードでは使用できません。
- ドロアーは、ブザーと一緒に使用できません。

構文

- (int) **addPulse**: (int) drawer Time: (int) time;

パラメーター

- drawer: ドロアーキックコネクタを指定します。

設定値	説明
EPOS_OC_DRAWER_1	ドロアーキックコネクタ 2 番ピン
EPOS_OC_DRAWER_2	ドロアーキックコネクタ 5 番ピン
EPOS_OC_PARAM_DEFAULT	ドロアーキックコネクタ 2 番ピン

- time: ドロアーキック信号のオン時間を指定します。

設定値	説明
EPOS_OC_PULSE_100	100 ミリ秒の信号
EPOS_OC_PULSE_200	200 ミリ秒の信号
EPOS_OC_PULSE_300	300 ミリ秒の信号
EPOS_OC_PULSE_400	400 ミリ秒の信号
EPOS_OC_PULSE_500	500 ミリ秒の信号
EPOS_OC_PARAM_DEFAULT	100 ミリ秒の信号

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

ドロアーキックコネクタ 2 番ピンに 100 ミリ秒のパルス信号を出力する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPulse: EPOS_OC_DRAWER_1 Time: EPOS_OC_PULSE_100];
    ... 処理 ...
}
```

addSound

ブザーの鳴動を命令バッファに追加します。ブザーを設定します。



- ページモードでは使用できません。
- ブザーの機能は、ドロアーと一緒に使用できません。
- 本 API はプリンターにブザーが付いてなければ使用できません。

構文

```
(int) addSound: (int)pattern Repeat: (long)repeat  
Cycle: (long)cycle;
```

パラメーター

- pattern: ブザーの音色を指定します。

設定値	説明
EPOS_OC_PATTERN_A	パターン A (外付けオプションブザー)
EPOS_OC_PATTERN_B	パターン B (外付けオプションブザー)
EPOS_OC_PATTERN_C	パターン C (外付けオプションブザー)
EPOS_OC_PATTERN_D	パターン D (外付けオプションブザー)
EPOS_OC_PATTERN_E	パターン E (外付けオプションブザー)
EPOS_OC_PATTERN_ERROR	エラー鳴動パターン (外付けオプションブザー)
EPOS_OC_PATTERN_PAPER_END	用紙なし鳴動パターン (外付けオプションブザー)
EPOS_OC_PATTERN_1	パターン 1 (内蔵ブザー)
EPOS_OC_PATTERN_2	パターン 2 (内蔵ブザー)
EPOS_OC_PATTERN_3	パターン 3 (内蔵ブザー)
EPOS_OC_PATTERN_4	パターン 4 (内蔵ブザー)
EPOS_OC_PATTERN_5	パターン 5 (内蔵ブザー)
EPOS_OC_PATTERN_6	パターン 6 (内蔵ブザー)
EPOS_OC_PATTERN_7	パターン 7 (内蔵ブザー)
EPOS_OC_PATTERN_8	パターン 8 (内蔵ブザー)
EPOS_OC_PATTERN_9	パターン 9 (内蔵ブザー)
EPOS_OC_PATTERN_10	パターン 10 (内蔵ブザー)
EPOS_OC_PARAM_DEFAULT	パターン A

- repeat: 繰り返し回数を指定します。

設定値	説明
1 ~ 255	1 ~ 255 回
EPOS_OC_PARAM_DEFAULT	1 回

- cycle: ブザーを鳴らす周期 (ミリ秒単位) を指定します。

設定値	説明
1000 ~ 25500	1000 ~ 25500 ミリ秒
EPOS_OC_PARAM_DEFAULT	1000 ミリ秒



パターン A ~ E / エラー鳴動パターン / 用紙なし鳴動パターンは無視されます。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。


例

パターン 1 を 1000 ミリ秒周期で 3 回鳴らす場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addSound: EPOS_OC_PATTERN_1 Repeat: 3 Cycle: 1000];
    . . . 処理 . . .
}
```

addSound (旧フォーマット)

ブザーの鳴動を命令バッファに追加します。ブザーを設定します。



- ブザーを鳴らす周期は設定できません。ブザーを鳴らす周期(ミリ秒単位)を任意で設定したい場合、[addSound \(91 ページ\)](#) を使用してください。
- ページモードでは使用できません。
- ブザーの機能は、ドロアーと一緒に使用できません。
- 本 API はプリンターにブザーが付いてなければ使用できません。

構文

```
- (int) addSound: (int)pattern Repeat: (long)repeat;
```

パラメーター

- pattern : ブザーの音色を指定します。

設定値	説明
EPOS_OC_PATTERN_A	パターン A (外付けオプションブザー)
EPOS_OC_PATTERN_B	パターン B (外付けオプションブザー)
EPOS_OC_PATTERN_C	パターン C (外付けオプションブザー)
EPOS_OC_PATTERN_D	パターン D (外付けオプションブザー)
EPOS_OC_PATTERN_E	パターン E (外付けオプションブザー)
EPOS_OC_PATTERN_ERROR	エラー鳴動パターン (外付けオプションブザー)
EPOS_OC_PATTERN_PAPER_END	用紙なし鳴動パターン (外付けオプションブザー)
EPOS_OC_PATTERN_1	パターン 1 (内蔵ブザー)
EPOS_OC_PATTERN_2	パターン 2 (内蔵ブザー)
EPOS_OC_PATTERN_3	パターン 3 (内蔵ブザー)
EPOS_OC_PATTERN_4	パターン 4 (内蔵ブザー)
EPOS_OC_PATTERN_5	パターン 5 (内蔵ブザー)
EPOS_OC_PATTERN_6	パターン 6 (内蔵ブザー)
EPOS_OC_PATTERN_7	パターン 7 (内蔵ブザー)
EPOS_OC_PATTERN_8	パターン 8 (内蔵ブザー)
EPOS_OC_PATTERN_9	パターン 9 (内蔵ブザー)
EPOS_OC_PATTERN_10	パターン 10 (内蔵ブザー)
EPOS_OC_PARAM_DEFAULT	パターン A

- repeat : 繰り返し回数を指定します。

設定値	説明
1 ~ 255	1 ~ 255 回
EPOS_OC_PARAM_DEFAULT	1 回

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

パターン A を 3 回鳴らす場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addSound: EPOS_OC_PATTERN_A Repeat: 3];
    . . . 処理 . . .
}
```

addFeedPosition

ラベル / ブラックマーク紙の紙送りを命令バッファに追加します。

構文

```
- (int) addFeedPosition: (int) position;
```

パラメーター

- position : 紙送りする位置を指定します。

設定値	説明
EPOS_OC_FEED_PEELING	剥離位置まで紙送り
EPOS_OC_FEED_CUTTING	カット位置まで紙送り
EPOS_OC_FEED_CURRENT_TOF	現在のラベル頭出し位置まで紙送り
EPOS_OC_FEED_NEXT_TOF	次のラベル頭出し位置まで紙送り

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

ラベル紙を剥離位置まで紙送りする場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60II"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addFeedPosition: EPOS_OC_FEED_PEELING];
    ... 処理 ...
}
```

addLayout

ラベル / ブラックマーク紙の用紙レイアウト情報を命令バッファに追加します。

構文

```
(int) addLayout:(int) type  
Width:(long) width Height:(long) height  
MarginTop:(long) marginTop  
MarginBottom:(long) marginBottom  
OffsetCut:(long) offsetCut  
OffsetLabel:(long) offsetLabel;
```

パラメーター

- type: 用紙種類を指定します。

設定値	説明
EPOS_OC_LAYOUT_RECEIPT	レシート紙 (ブラックマークなし)
EPOS_OC_LAYOUT_LABEL	ラベル紙 (ブラックマークなし)
EPOS_OC_LAYOUT_LABEL_BM	ラベル紙 (ブラックマークあり)
EPOS_OC_LAYOUT_RECEIPT_BM	レシート紙 (ブラックマークあり)

- width: 用紙幅 (0.1 mm 単位) を指定します。1 ~ 10000 の整数値で指定します。
- height: 印字基準から次の印字基準までの距離 (0.1 mm 単位) を指定します。
1 ~ 10000 の整数値で指定します。
0 を指定した場合、印字基準位置から次の印字基準位置までの距離を自動検出します。
- marginTop: 印字基準から頭出し位置までの距離 (0.1 mm 単位) を指定します。
-9999 ~ 10000 の整数値で指定します。
- marginBottom: 排出基準から印刷可能領域の下端までの距離 (0.1 mm 単位) を指定します。
-9999 ~ 10000 の整数値で指定します。
- offsetCut: 排出基準からカット位置までの距離 (0.1 mm 単位) を指定します。
-9999 ~ 10000 の整数値で指定します。
- offsetLabel: 排出基準からラベル下端までの距離 (0.1 mm 単位) を指定します。
0 ~ 10000 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

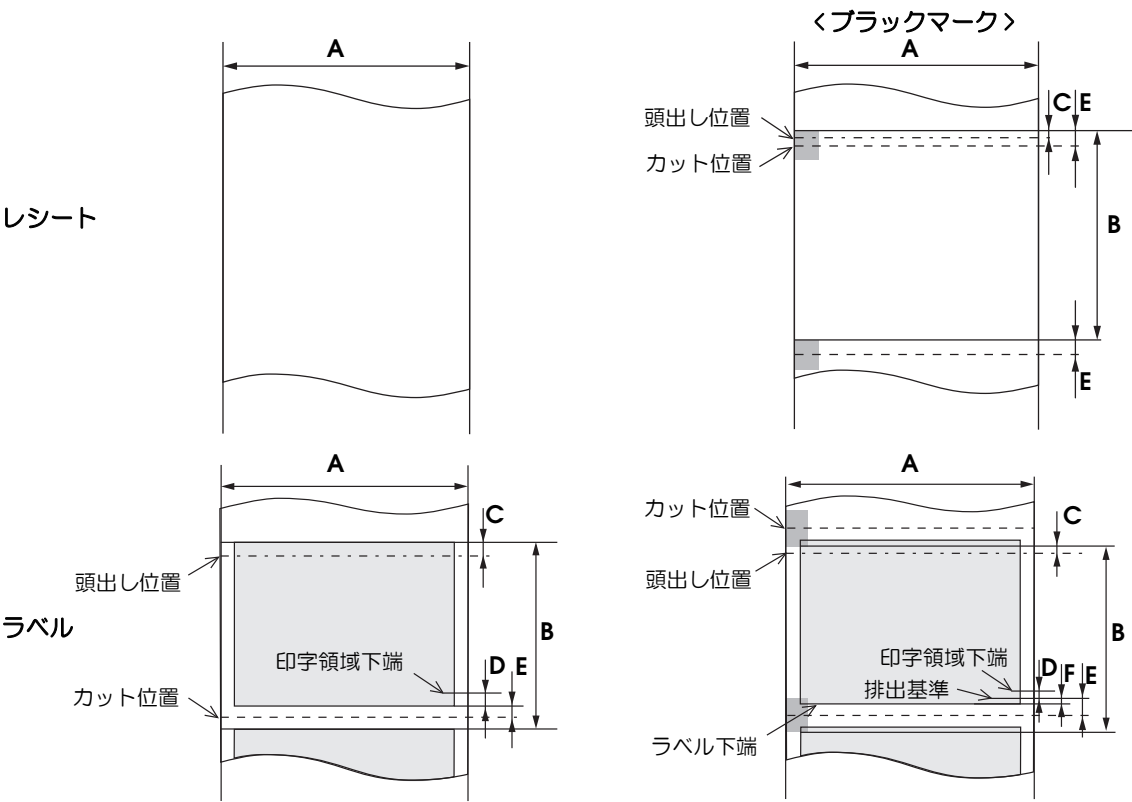
例

60 mm ラベル紙 (ブラックマークあり) に設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60II"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addLayout: EPOS_OC_PAPER_TYPE_LABEL_BM
        Width:600 Height:0 MarginTop:15 MarginBottom:-15
        OffsetCut:15 OffsetLabel:0];
    ... 処理 ...
}
```

詳細説明

□ 用紙ごと指定可能なパラメーターと、パラメーターの位置は以下を参照してください。



記号	パラメーター	設定値			
		レシート	レシート (ブラックマーク)	ラベル	ラベル (ブラックマーク)
A	width	1 ~ 10000	1 ~ 10000	1 ~ 10000	1 ~ 10000
B	height	0	0 ~ 10000	0 ~ 10000	0 ~ 10000
C	marginTop	0	-9999 ~ 10000	0 ~ 10000	-9999 ~ 10000
D	marginBottom	0	0	-9999 ~ 0	-9999 ~ 10000
E	offsetCut	0	-9999 ~ 10000	0 ~ 10000	0 ~ 10000
F	offsetLabel	0	0	0	0 ~ 10000

addCommand

コマンドを命令バッファに追加します。ESC/POS コマンドを送信します。



コマンドの詳細は、ESC/POS コマンドリファレンスを参照してください。
https://reference.epson-biz.com/modules/ref_escpos_ja/

構文

```
- (int) addCommand: (NSData *)data;
```

パラメーター

- data: ESC/POS コマンドをバイナリーデータで指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    NSData* data = Nil;
    ... 処理 ...
    errorStatus = [builder addCommand: data];
}
```

init

EposPrint クラスのインスタンスを初期化します。

構文

– (id) **init**;

戻り値

初期化済みの EposPrint クラスインスタンスが返ります。

例

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    ...処理...
    [printer release];
}
```

openPrinter

プリンターとの通信・プリンターステータスのモニタリングを開始します。



プリンターとの通信が不要になった場合、必ず [closePrinter \(106 ページ\)](#) を呼び出し、プリンターとの通信を終了してください。



- プリンターステータスは、EposPrint クラスで登録したコールバックメソッドに通知されます。詳細は、[プリンターステータスを自動で取得 \(36 ページ\)](#) を参照してください。
- プリンターステータスのモニタリングをやめたい場合、[closePrinter \(106 ページ\)](#) を呼び出してください。
- 一台のプリンターを複数のモバイル端末から使用する場合、[注意事項 \(164 ページ\)](#) を参照してください。

構文

```
(int) openPrinter: (int) deviceType  
                        DeviceName: (NSString *) deviceName  
                        Enabled: (int) enabled  
                        Interval: (long) interval  
                        Timeout: (long) timeout;
```

パラメーター

- deviceType: 通信を開始するデバイスの種別を指定します。

設定値	説明
EPOS_OC_DEVTYPE_TCP	Wi-Fi/Ethernet デバイス
EPOS_OC_DEVTYPE_BLUETOOTH	Bluetooth デバイス

- deviceName: 対象デバイスを特定するための識別子を指定します。
deviceType ごとに以下を指定します。

deviceType	設定値
EPOS_OC_DEVTYPE_TCP	以下のいずれかを指定できます。 <ul style="list-style-type: none">• IPv4 形式の IP アドレス (例: "192.168.192.168")• MAC アドレス (例: "01:23:45:67:89:AB")• プリンターホスト名 (任意の文字列)
EPOS_OC_DEVTYPE_BLUETOOTH	MAC アドレス (例: "01:23:45:67:89:AB")



- プリンターの IP アドレスを DHCP に設定している場合、deviceName に Mac アドレスまたはプリンターホスト名を指定してください。
- deviceType が EPOS_OC_DEVTYPE_TCP で、deviceName にプリンターホスト名を指定する場合、DNS サーバーからプリンターホスト名が検索可能な環境で使用してください。

- enabled: プリンターステータスのモニタリングの有効・無効を指定します。

設定値	設定値
EPOS_OC_TRUE	有効
EPOS_OC_FALSE	無効
EPOS_OC_PARAM_DEFAULT	無効

- interval: プリンターステータスを更新する間隔 (ミリ秒単位) を指定します。

設定値	設定値
1000 ~ 60000 の整数	プリンターステータスを更新する間隔 (ミリ秒単位)
EPOS_OC_PARAM_DEFAULT	1000 ミリ秒

- timeout: プリンターと通信確立するための最大待ち時間 (ミリ秒単位) を指定します。

設定値	設定値
1000 ~ 300000 の整数	エラーを返すまでの最大待ち時間 (ミリ秒単位)
EPOS_OC_PARAM_DEFAULT	既定値 (15000) を指定



- 指定したデバイスが存在しない場合、ただちにエラーを返します。
- deviceType が EPOS_OC_DEVTYPE_TCP で、指定したデバイスがすでに使用されている場合、タイムアウト時間まで本 API を再試行します。
- Bluetooth 通信する場合、EPOS_OC_PARAM_DEFAULT を設定してください。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_OPEN	ポートオープン処理に失敗した。
EPOS_OC_ERR_TIMEOUT	指定したデバイスがすでに使用されていて、タイムアウト時間内にプリンターと通信確立できなかった。
EPOS_OC_ERR_ILLEGAL	すでに通信が開始されているデバイスを再度通信開始しようとした。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

IP アドレスが "192.168.192.168" のプリンターと Wi-Fi/Ethernet でプリンターステータスのモニタリングを有効にして通信を開始する場合

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
        Name:@"192.168.192.168" Enabled: EPOS_OC_TRUE
        Interval:EPOS_OC_PARAM_DEFAULT Timeout:EPOS_OC_PARAM_DEFAULT];
    ... 処理 ...
}
```

openPrinter (旧フォーマット)

プリンターとの通信・プリンターステータスのモニタリングを開始します。



プリンターとの通信が不要になった場合、必ず [closePrinter \(106 ページ\)](#) を呼び出し、プリンターとの通信を終了してください。



- 本 API のタイムアウト時間は設定できません。本 API のタイムアウト時間を設定したい場合、[openPrinter \(100 ページ\)](#) を使用してください。
- プリンターステータスは、EposPrint クラスで登録したコールバックメソッドに通知されます。詳細は、[プリンターステータスを自動で取得 \(36 ページ\)](#) を参照してください。
- プリンターステータスのモニタリングをやめたい場合、[closePrinter \(106 ページ\)](#) を呼び出して下さい。
- 他のアプリケーションがプリンターをオープンしている場合、接続形式によって下記の注意が必要です。
 - * TCP 接続: 本 API を 15 秒間再試行します。15 秒後に EPOS_OC_ERR_OPEN が返されます。
 - * Bluetooth 接続: 本 API で通信を開始しようとすると、処理が戻ってこない場合があります。
- 一台のプリンターを複数のモバイル端末から使用する場合、[注意事項 \(164 ページ\)](#) を参照してください。

構文

```
- (int) openPrinter: (int)deviceType  
                    DeviceName: (NSString *)deviceName  
                    Enabled: (int)enabled  
                    Interval: (long)interval;
```

パラメーター

- deviceType: 通信を開始するデバイスの種別を指定します。

設定値	説明
EPOS_OC_DEVTYPE_TCP	Wi-Fi/Ethernet デバイス
EPOS_OC_DEVTYPE_BLUETOOTH	Bluetooth デバイス

- deviceName: 対象デバイスを特定するための識別子を指定します。
deviceType ごとに以下を指定します。

deviceType	設定値
EPOS_OC_DEVTYPE_TCP	以下のいずれかを指定できます。 <ul style="list-style-type: none">• IPv4 形式の IP アドレス (例: "192.168.192.168")• MAC アドレス (例: "01:23:45:67:89:AB")• プリンターホスト名 (任意の文字列)
EPOS_OC_DEVTYPE_BLUETOOTH	BD アドレス (例: "01:23:45:67:89:AB")



- プリンターの IP アドレスを DHCP に設定している場合、deviceName に Mac アドレスまたはプリンターホスト名を指定してください。
- deviceType が EPOS_OC_DEVTYPE_TCP で、deviceName にプリンターホスト名を指定する場合、DNS サーバーからプリンターホスト名が検索可能な環境で使用してください。

- enabled : プリンターステータスのモニタリングの有効・無効を指定します。

設定値	設定値
EPOS_OC_TRUE	有効
EPOS_OC_FALSE	無効
EPOS_OC_PARAM_DEFAULT	無効

- interval : プリンターステータスを更新する間隔（ミリ秒単位）を指定します。

設定値	設定値
1000 ～ 60000 の整数	プリンターステータスを更新する間隔（ミリ秒単位）
EPOS_OC_PARAM_DEFAULT	1000 ミリ秒

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_OPEN	• ポートオープン処理に失敗した。 • プリンターがすでに使われていた。
EPOS_OC_ERR_ILLEGAL	すでに通信が開始されているデバイスを再度通信開始しようとした。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

IP アドレスが “192.168.192.168” のプリンターと Wi-Fi/Ethernet でプリンターステータスのモニタリングを有効にして通信を開始する場合

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
        Name:@"192.168.192.168" Enabled: EPOS_OC_TRUE
        Interval:EPOS_OC_PARAM_DEFAULT];
    ... 処理 ...
}
```

openPrinter (旧フォーマット)

プリンターとの通信を開始します。



プリンターとの通信が不要になった場合、必ず [closePrinter \(106 ページ\)](#) を呼び出し、プリンターとの通信を終了してください。



- 本 API のタイムアウト時間は設定できません。本 API のタイムアウト時間を設定したい場合、[openPrinter \(100 ページ\)](#) を使用してください。
- プリンターステータスを自動で取得したい場合、[openPrinter \(100 ページ\)](#) を使用してください。
- 他のアプリケーションがプリンターをオープンしている場合、接続形式によって下記の注意が必要です。
 - * TCP 接続: 本 API を 15 秒間再試行します。15 秒後に EPOS_OC_ERR_OPEN が返されます。
 - * Bluetooth 接続: 本 API で通信を開始しようとすると、処理が戻ってこない場合があります。
- 一台のプリンターを複数のモバイル端末から使用する場合、[注意事項 \(164 ページ\)](#) を参照してください。

構文

```
(int) openPrinter: (int) deviceType  
                        DeviceName: (NSString *) deviceName;
```

パラメーター

- deviceType: 通信を開始するデバイスの種別を指定します。

設定値	説明
EPOS_OC_DEVTYPE_TCP	Wi-Fi/Ethernet デバイス
EPOS_OC_DEVTYPE_BLUETOOTH	Bluetooth デバイス

- deviceName: 対象デバイスを特定するための識別子を指定します。
deviceType ごとに以下を指定します。

deviceType	設定値
EPOS_OC_DEVTYPE_TCP	以下のいずれかを指定できます。 <ul style="list-style-type: none">• IPv4 形式の IP アドレス (例: "192.168.192.168")• MAC アドレス (例: "01:23:45:67:89:AB")• プリンターホスト名 (任意の文字列)
EPOS_OC_DEVTYPE_BLUETOOTH	BD アドレス (例: "01:23:45:67:89:AB")



- プリンターの IP アドレスを DHCP に設定している場合、deviceName に Mac アドレスまたはプリンターホスト名を指定してください。
- deviceType が EPOS_OC_DEVTYPE_TCP で、deviceName にプリンターホスト名を指定する場合、DNS サーバーからプリンターホスト名が検索可能な環境で使用してください。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_OPEN	<ul style="list-style-type: none">ポートオープン処理に失敗した。プリンターがすでに使われていた。
EPOS_OC_ERR_ILLEGAL	すでに通信が開始されているデバイスを再度通信開始しようとした。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

IP アドレスが“192.168.192.168”のプリンターと Wi-Fi/Ethernet で通信を開始する場合

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
        DeviceName:@"192.168.192.168"];
    . . . 処理 . . .
}
```

closePrinter

プリンターとの通信、およびプリンターステータスのモニタリングを終了します。

構文

– (int) **closePrinter**;

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_ILLEGAL	通信が開始されていない状態で、本 API が呼び出された。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
        DeviceName:@"192.168.192.168"];
    ... 処理 ...
    errorStatus = [printer closePrinter];
}
```

sendData

EposBuilder クラスで作成した印刷ドキュメントを送信します。



- Bluetooth 接続の場合、オフライン状態が検出できずタイムアウトエラーになることがあります。
- 一台のプリンターを複数のモバイル端末から使用する場合、[注意事項 \(164 ページ\)](#) を参照してください。

構文

```
- (int) sendData: (EposBuilder *)builder
               Timeout: (long)timeout
               Status: (unsigned long *)status
               Battery: (unsigned long *)battery;
```

パラメーター

- builder : EposBuilder クラスのインスタンスを指定します。
EposBuilder クラスの詳細は、[EposBuilder クラス \(43 ページ\)](#) を参照してください。
- timeout : 送受信待ちのタイムアウト時間を指定します。機種仕様の仕様、通信インターフェイス、送信データサイズによって、timeout 時間を調整してください。
0 ～ 6000000(ミリ秒単位) の整数値を指定します。
- status : コマンド送信終了時のプリンターステータスがセットされます。プリンターステータスの設定値の組み合わせがセットされます。詳細は、[プリンターステータスと対処方法 \(40 ページ\)](#) を参照してください。
- battery : バッテリーステータスがセットされます。
詳細は、[プリンター別サポート情報 \(149 ページ\)](#) を参照してください。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_ILLEGAL	通信が開始していない状態で本 API が呼び出された。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_TIMEOUT	指定された時間内に全データを送信できなかった。
EPOS_OC_ERR_CONNECT	通信エラーが発生した。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_OFF_LINE	プリンターがオフライン状態だった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

タイムアウトに 10 秒を指定し、プリンターにコマンドを送信する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
             Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    unsigned long status = 0;
    unsigned long battery = 0;

    errorStatus = [builder addText:@"ABCDE"];

    id printer = [[EposPrint alloc] init];

    if ( printer != nil ) {
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                         DeviceName:@"192.168.192.168"];
        errorStatus = [printer sendData:builder Timeout:10000
                        Status:&status Battery:&battery];
        errorStatus = [printer closePrinter];
        [printer release];
    }
    [builder release];
}
```

sendData (旧フォーマット)

EposBuilder クラスで作成した印刷ドキュメントを送信します。バッテリーステータスは、取得できません。



- Bluetooth 接続の場合、オフライン状態が検出できずタイムアウトエラーになることがあります。
- バッテリーステータスを取得できません。印刷ドキュメント送信時に、バッテリーステータスを取得したい場合、[sendData \(107 ページ\)](#) を使用してください。
- 一台のプリンターを複数のモバイル端末から使用する場合、[注意事項 \(164 ページ\)](#) を参照してください。

構文

```
(int) sendData: (EposBuilder *)builder
                Timeout: (long) timeout
                Status: (unsigned long *) status;
```

パラメーター

- builder : EposBuilder クラスのインスタンスを指定します。
EposBuilder クラスの詳細は、[EposBuilder クラス \(43 ページ\)](#) を参照してください。
- timeout : 送受信待ちのタイムアウト時間を指定します。機種仕様の仕様、通信インターフェイス、送信データサイズによって、timeout 時間を調整してください。
0 ~ 6000000 (ミリ秒単位) の整数値を指定します。
- status : コマンド送信終了時のプリンターステータスがセットされます。プリンターステータスの設定値の組み合わせがセットされます。詳細は、[プリンターステータスと対処方法 \(40 ページ\)](#) を参照してください。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_ILLEGAL	通信が開始していない状態で本 API が呼び出された。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_TIMEOUT	指定された時間内に全データを送信できなかった。
EPOS_OC_ERR_CONNECT	通信エラーが発生した。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_OFF_LINE	プリンターがオフライン状態だった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

タイムアウトに 10 秒を指定し、プリンターにコマンドを送信する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
             Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    unsigned long status = 0;

    errorStatus = [builder addText:@"ABCDE"];

    id printer = [[EposPrint alloc] init];

    if ( printer != nil ) {
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                         DeviceName:@"192.168.192.168"];
        errorStatus = [printer sendData:builder Timeout:10000
                     Status:&status];
        errorStatus = [printer closePrinter];
        [printer release];
    }
    [builder release];
}
```

setStatusChangeEventCallback

プリンタステータスのイベントのコールバックメソッドを登録します。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

- (void) **setStatusChangeEventCallback:** (SEL) method
Target: (NSObject*) target;

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

- (void) **メソッドの名称:** (NSString *)deviceName
Status: (NSNumber *)status;

パラメーター

- deviceName: プリンタステータスを通知した、デバイスの識別子 (IPv4形式のIPアドレス / BDアドレス / プリンターホスト名) がセットされます。
- status: プリンタステータスがセットされます。

例

```
- (void)onStatusChange:(NSString *)deviceName Status:(NSNumber *)status
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setStatusChangeEventCallback @selector(onStatusChange:Status:)
                                         Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                         Name:@"192.168.192.168"
                                         Enabled: EPOS_OC_TRUE
                                         Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setOnlineEventCallback

オンラインイベントのコールバックメソッドを登録します。プリンタステータスがオンライン時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setOnlineEventCallback: (SEL) method  
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: オンラインイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / BD アドレス / プリンターホスト名) がセットされます。

例

```
- (void)onOnline:(NSString *)deviceName  
{  
    ... 処理 ...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setOnlineEventCallback @selector(onOnline:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ... 処理 ...  
    }  
}
```


setOfflineEventCallback

オフラインイベントのコールバックメソッドを登録します。プリンタステータスがオフライン時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
- (void) setOfflineEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
- (void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: オフラインイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / BD アドレス / プリンターホスト名) がセットされます。

例

```
- (void)onOffline:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setOfflineEventCallback @selector(onOffline:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                     Name:@"192.168.192.168"
                                     Enabled: EPOS_OC_TRUE
                                     Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setPowerOffEventCallback

無応答イベントのコールバックメソッドを登録します。プリンタステータスが無応答時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

- (void) **setPowerOffEventCallback:** (SEL) method
Target: (NSObject*) target;

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

- (void) **メソッドの名称:** (NSString *)deviceName

パラメーター

- deviceName: 無応答イベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / BD アドレス / プリンターホスト名) がセットされます。

例

```
- (void)onPowerOff:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setPowerOffEventCallback @selector(onPowerOff:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                   Name:@"192.168.192.168"
                                   Enabled: EPOS_OC_TRUE
                                   Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setCoverOkEventCallback

カバークローズイベントのコールバックメソッドを登録します。プリンタステータスがカバークローズ時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setCoverOkEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: カバークローズイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / BD アドレス / プリンターホスト名) がセットされます。

例

```
- (void)onCoverOk:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setCoverOkEventCallback @selector(onCoverOk:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setCoverOpenEventCallback

カバーオープンイベントのコールバックメソッドを登録します。プリンタステータスがカバーオープン時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setCoverOpenEventCallback: (SEL) method  
Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: カバーオープンイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / BD アドレス / プリンターホスト名) がセットされます。

例

```
- (void)onCoverOpen:(NSString *)deviceName  
{  
    ... 処理 ...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setCoverOpenEventCallback @selector(onCoverOpen:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ... 処理 ...  
    }  
}
```

setPaperOkEventCallback

用紙ありイベントのコールバックメソッドを登録します。プリンタステータスが用紙あり時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setPaperOkEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: 用紙ありイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / BD アドレス / プリンターホスト名) がセットされます。

例

```
- (void)onPaperOk:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setPaperOkEventCallback @selector(onPaperOk:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setPaperNearEndEventCallback

用紙残量少イベントのコールバックメソッドを登録します。プリンタステータスが用紙残量少時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

- (void) **setPaperNearEndEventCallback:** (SEL) method
Target: (NSObject*) target;

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

- (void) **メソッドの名称:** (NSString *)deviceName

パラメーター

- deviceName: 用紙残量少イベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / BD アドレス / プリンターホスト名) がセットされます。

例

```
- (void)onPaperNearEnd:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setPaperNearEndEventCallback @selector(onPaperNearEnd:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setPaperEndEventCallback

用紙なしイベントのコールバックメソッドを登録します。プリンタステータスが用紙なし時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setPaperEndEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: 用紙なしイベントを通知した、デバイスの識別子 (IPv4形式のIPアドレス/ BDアドレス/ プリンターホスト名) がセットされます。

例

```
- (void)onPaperEnd:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setPaperEndEventCallback @selector(onPaperEnd:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setDrawerClosedEventCallback

ドロアークローズイベントのコールバックメソッドを登録します。プリンタステータスがドロアークローズ時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setDrawerClosedEventCallback: (SEL) method  
Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: ドロアークローズイベントを通知した、デバイスの識別子 (IPv4形式のIPアドレス / BDアドレス / プリンターホスト名) がセットされます。

例

```
- (void)onDrawerClosed:(NSString *)deviceName  
{  
    ... 処理 ...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setDrawerClosedEventCallback @selector(onDrawerClosed:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ... 処理 ...  
    }  
}
```


setDrawerOpenEventCallback

ドロアーオープンイベントのコールバックメソッドを登録します。プリンタステータスがドロアーオープン時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setDrawerOpenEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: ドロアーオープンイベントを通知した、デバイスの識別子 (IPv4形式のIPアドレス / BDアドレス / プリンターホスト名) がセットされます。

例

```
- (void)onDrawerOpen:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setDrawerOpenEventCallback @selector(onDrawerOpen:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setBatteryLowEventCallback

バッテリー残量なしイベントのコールバックメソッドを登録します。プリンタステータスがバッテリー残量によるオフライン時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setBatteryLowEventCallback: (SEL) method  
Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: バッテリー残量なしを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / BD アドレス / プリンターホスト名) がセットされます。

例

```
- (void)onBatteryLow:(NSString *)deviceName  
{  
    ...処理...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setBatteryLowEventCallback @selector(onBatteryLow:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ...処理...  
    }  
}
```

setBatteryOkEventCallback

バッテリー残量ありイベントのコールバックメソッドを登録します。プリンタステータスがバッテリー残量によるオフラインから復帰時に通知されるメソッドです。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setBatteryOkEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: バッテリー残量ありイベントを通知した、デバイスの識別子 (IPv4形式のIPアドレス/ BDアドレス/ プリンターホスト名) がセットされます。

例

```
- (void)onBatteryOk:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setBatteryOkEventCallback @selector(onBatteryOk:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setBatteryStatusChangeEventCallback

バッテリーステータスのイベントのコールバックメソッドを登録します。



- 本 API は、[openPrinter \(100 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setBatteryStatusChangeEventCallback:  
      (SEL) method Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName  
                        Battery: (NSNumber *)battery;
```

パラメーター

- deviceName: バッテリーステータスを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / BD アドレス / プリンターホスト名) がセットされます。
- battery: バッテリーステータスがセットされます。

例

```
- (void)onBatteryStatusChange:(NSString *)deviceName Battery:(NSNumber *)battery  
{  
    ... 処理 ...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setBatteryStatusChangeEventCallback  
                @selector(onBatteryStatusChange:Battery:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ... 処理 ...  
    }  
}
```

プリンター検索 API

プリンターを検索するための API です。以下のクラスが用意されています。

□ EpsonIoFinder クラス (125 ページ)


EpsonIoFinder クラス

プリンターを検索するクラスです。以下の API が用意されています。

API	説明	ページ
start	プリンター検索を開始	125
stop	プリンターとの通信を終了	127
getDeviceInfoList	プリンターの検索結果を取得	130
getResult (旧フォーマット)		130

start

指定されたデバイス種別のプリンター検索を開始します。



- 本 API を使用したら、必ず [stop \(127 ページ\)](#) で検索終了してください。
- すでにプリンター検索を開始している状態で、本 API を呼び出すことはできません。

構文

```
+ (int) start: (int)deviceType  
FindOption: (NSString *)findOption;
```

パラメーター

- deviceType: 検索するデバイス種別を指定します。以下の値を指定します。

deviceType	説明
EPSONIO_OC_DEVTYPE_TCP	ネットワークに接続された TM デバイスを検索します。
EPSONIO_OC_DEVTYPE_BLUETOOTH	ESC/POS をサポートした、iOS から <i>Bluetooth</i> による接続が可能なプリンターを検索します。

- findOption: 対象デバイスを検索する際の設定値を指定します。

deviceType	指定する値
EPSONIO_OC_DEVTYPE_TCP	検索範囲のブロードキャストアドレス
EPSONIO_OC_DEVTYPE_BULETOOTH	nil

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した。
EPSONIO_OC_ERR_ILLEGAL	すでに検索を開始した状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

stop

プリンター検索を終了します。

構文

```
+ (int) stop;
```

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した。
EPSONIO_OC_ERR_ILLEGAL	検索を開始していない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

getDeviceInfoList

本 API を呼び出した時点までの、デバイスの検索結果を取得します。



オープン済みの *Bluetooth* デバイスは、取得できません。

構文

```
+ (NSArray *) getDeviceInfoList: (int *)errorStatus  
    FilterOption: (int)filterOption;
```

パラメーター

- errorStatus: デバイスリストの取得結果が返されます。

エラーステータス	説明
EPSONIO_OC_SUCCESS	デバイスリストの取得に成功
EPSONIO_OC_ERR_ILLEGAL	検索を開始していない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理が実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

- filterOption: エプソン製プリンターのフィルタリング方法を指定します。以下の値を指定します。

設定値	説明
EPSONIO_OC_FILTER_NONE	フィルタリングしない
EPSONIO_OC_FILTER_NAME	プリンター名でフィルタリングする
EPSONIO_OC_PARAM_DEFAULT	既定値 (プリンター名でフィルタリングする)



iOS 端末の場合、filterOption の設定に関係なく、エプソン製プリンターのみ検索されます。

戻り値

検索されたデバイスのデバイス情報リスト (NSArray 型) が返されます。
リスト内には、デバイス情報が EpsonIoDeviceInfo 型で格納されています。
デバイス種別 (deviceType) によって、格納される情報が異なります。

deviceType	EpsonIoDeviceInfo	取得する情報
EPSONIO_OC_DEVTYPE_TCP	deviceType	EPSONIO_OC_DEVTYPE_TCP(固定)
	printerName	プリンターモデル名
	deviceName	• DHCP 無効の場合 : IP アドレス • DHCP 有効の場合 : MAC アドレス
	ipAddress	IP アドレス
	macAddress	MAC アドレス
EPSONIO_OC_DEVTYPE_BLUETOOTH	deviceType	EPSONIO_OC_DEVTYPE_BLUETOOTH(固定)
	printerName	Bluetooth デバイス名
	deviceName	BD アドレス (MAC アドレスと同じ形式)
	ipAddress	"" (空文字)
	macAddress	"" (空文字)

getResult (旧フォーマット)

本 API を呼び出した時点までの、プリンターの検索結果を取得します。



オープン済みの *Bluetooth* デバイスは、取得できません。

構文

```
+ (NSArray *) getResult: (int *)errorStatus;
```

パラメーター

- errorStatus: エラーステータスが返ります。

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した。
EPSONIO_OC_ERR_ILLEGAL	検索を開始していない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

戻り値

検索したデバイスのリストが返されます。

リスト内には、検索したデバイスの識別情報が、文字列 (String 型) で格納されています。

デバイス種別 (deviceType) によって、格納される結果が異なります。

deviceType	取得するリスト
EPSONIO_OC_DEVTYPE_TCP	プリンターの IP アドレスのリスト
EPSONIO_OC_DEVTYPE_BLUETOOTH	<i>Bluetooth</i> デバイスの BD アドレスのリスト

プリンター簡単選択 API

QR コードを使ってプリンターを選択する際に使う API です。QR コードから取得したデータを openPrinter に渡せる形に変換します。以下のクラスが用意されています。

- EposEasySelect クラス (131 ページ)
- EposEasySelectInfo クラス (131 ページ)

EposEasySelect クラス

QR コードデータを解析します。以下の API が用意されています。

API	説明	ページ
parseQR	QR コードデータの解析	131
createQR	簡単選択用 QR コード印刷データの作成	132

EposEasySelectInfo クラス

EasySelect クラスで解析したデータを格納し、openPrinter に渡す変数に変換するクラスです。以下のメンバー変数が用意されています。

メンバー変数	説明	ページ
deviceType	解析結果のデバイス種別	133
printerName	解析結果のプリンター名	133
macAddress	解析結果の MAC アドレス / BD アドレス	133

4

parseQR

文字列の QR コードのデータを解析します。

構文

- EposEasySelectInfo* **parseQR:** (NSString *)data

パラメーター

- data : QR コードの文字列データを指定します。

戻り値

QR コードの文字列データの解析結果が返されます。EposEasySelectInfo クラスに格納します。解析に失敗した場合、nil が返されます。

createQR

簡単選択用 QR コードの印刷データを作成します。

構文

```
(NSString *) createQR:(NSString *)printerName  
                        DeviceType:(int)deviceType  
                        MacAddress(NSString*) :macAddress
```

パラメーター

- printerName : プリンター名を指定します。
- deviceType : デバイスの種別を指定します。以下を指定します。

設定値	説明
EPOS_OC_DEVTYPE_TCP	Wi-Fi/ Ethernet デバイス
EPOS_OC_DEVTYPE_BLUETOOTH	Bluetooth デバイス

- macAddress : BD アドレスを指定します。
BD アドレスは、以下のフォーマットに対応しています。

フォーマット	説明
00:11:22:33:44:55	":" コロン区切り
00-11-22-33-44-55	"-" ハイフン区切り
001122334455	区切りなし

戻り値

簡単選択用 QR コードの印刷データが返されます。印刷データの作成に失敗した場合、nil が返されます。

deviceType

解析結果のデバイス種別を格納します。

格納されるデータ	説明
EPOS_OC_DEVTYPE_TCP	Wi-Fi/ Ethernet デバイス
EPOS_OC_DEVTYPE_BLUETOOTH	NFC 標準規格 (Bluetooth 用)

書式

```
int deviceType;
```

printerName

解析結果のプリンター名を格納します。

書式

```
NSString *printerName;
```

macAddress

解析結果の BD アドレスを格納します。

書式

```
String macAddress;
```

ログ設定 API

ログ出力の設定をします。以下のクラスが用意されています。

□ EposLog クラス ([134 ページ](#))

EposLog クラス

ログの出力機能を設定します。

API	説明	ページ
setLogSettings	ログ出力機能の設定	134

setLogSettings

ログ出力機能を設定します。

構文

```
+ (int) setLogSettings: (int) period
                        Enabled: (int) enabled
                        IPAddress: (NSString *) ipAddress
                        Port: (int) port LogSize: (int) logSize
                        LogLevel: (int) logLevel;
```

パラメーター

- period: ログ出力機能の設定方法を指定します。

設定値	説明
EPOS_OC_LOG_TEMPORARY	アプリケーションを終了すると、本 API の設定は無効になります。
EPOS_OC_LOG_PERMANENT	アプリケーションを終了させても、本 API の設定を有効にします。

- enabled: ログ出力機能の有効 / 無効、およびログの出力先を指定します。


設定値	説明
EPOS_OC_LOG_DISABLE	ログ出力機能を無効にする。
EPOS_OC_LOG_STORAGE	端末のストレージに出力する。
EPOS_OC_LOG_TCP	TCP で出力する。



enabled を EPOS_OC_LOG_STORAGE に指定する場合、iTunes のファイル共有を可能にしてください。以下の手順で設定します。

- アプリケーションの info.plist に “UIFileSharingEnabled” を追加します。
自動で “Application supports iTunes file sharing” に変更されます。
- “Application supports iTunes file sharing” の Value を、“YES” に設定します。


- `ipAddress` : TCP 通信の IP アドレス (IPv4 形式) を指定します。



`enabled` が以下の値の場合、“`nil`” も指定できます。

- * `EPOS_OC_LOG_DISABLE`
- * `EPOS_OC_LOG_STORAGE`


- `port` : TCP 通信のポート番号を指定します。0 ~ 65535 の整数値を指定します。



`enabled` に以下の値を指定した場合も、範囲内の任意の値を指定してください。

- * `EPOS_OC_LOG_DISABLE`
- * `EPOS_OC_LOG_STORAGE`

- `logSize` : 端末のストレージへ保存する、ログの最大容量を指定します。
1 ~ 50 (MB 単位) の整数値を指定します。



`enabled` に以下の値を指定した場合も、範囲内の任意の値を指定してください。

- * `EPOS_OC_LOG_DISABLE`
- * `EPOS_OC_LOG_TCP`

- `logLevel` : ログの出力レベルを指定します。

設定値	説明
<code>EPOS_OC_LOG_LOW</code>	低レベル

戻り値

エラーステータス	説明
<code>EPOS_OC_SUCCESS</code>	処理に成功した。
<code>EPOS_OC_ERR_PARAM</code>	不正なパラメーターが渡された。
<code>EPOS_OC_ERR_FAILURE</code>	その他のエラーが発生した。

例

TCP で、IP アドレス 192.168.192.168 の 8080 番ポートにログを出力する場合

```
errorStatus = [EposLog setLogSettings: EPOS_OC_LOG_PERMANENT
                Enabled:EPOS_OC_LOG_TCP ipAddress:@"192.168.192.168"
                Port:8080 LogSize:10 LogLevel: EPOS_OC_LOG_LOW ];
... 処理 ...
}
```

端末のストレージにログを出力する場合

```
errorStatus = [EposLog setLogSettings: EPOS_OC_LOG_PERMANENT
                Enabled: EPOS_OC_LOG_STORAGE ipAddress:nil
                Port:0 LogSize:10 LogLevel: EPOS_OC_LOG_LOW ];
... 処理 ...
}
```

ログ出力機能を無効にする場合

```
errorStatus = [EposLog setLogSettings: EPOS_OC_LOG_PERMANENT
                Enabled: EPOS_OC_LOG_DISABLE ipAddress:nil
                Port:0 LogSize:10 LogLevel: EPOS_OC_LOG_LOW ];
... 処理 ...
}
```

ログファイルの取り出し方法

保存先

□ itunes を使用してファイル共有から Log ファイルを取得してください。

http://support.apple.com/kb/HT4094?viewlocale=ja_JP

ファイル名

□ EposLog.xx

ログの見方

ログのフォーマット

ログのレコードは、以下の形式で構成されています。

≪ 日時, プロセス ID: スレッド ID, 入出力階層, 入出力方向, 入出力データ ≫

項目	説明
日時	yyyy/mm/dd,h:mm:ss.000 の形式です。
プロセス ID: スレッド ID	各処理の ID です。
入出力階層	データを入出力している階層です。 <ul style="list-style-type: none">• APIIO: アプリケーションから呼び出されるインターフェイス層• IOCM/DEVIO: デバイスとの通信層
入出力方向	データの入出力方向です。 <ul style="list-style-type: none">• ->: 階層からの入力です。• <=: 階層からの出力です。
入出力データ	呼び出された API、パラメーターおよび通信データです。



各項目は、コンマ (,) で区切られてます。

出力例

アプリケーションから addcut メソッドを呼び出す場合:

```
2014/07/28,20:12:35.836,00002ae9:00006008,APIIO,->,0x687bc5d8,,addCut,1
2014/07/28,20:12:35.836,00002ae9:00006008,APIIO,<-,0x687bc5d8,0,addCut}
```


コマンドの送受信

本章では、コマンド (ESC/POS コマンドなど) を送受信するための API について説明しています。



本章で説明している、コマンドを送受信するための API は、ESC/POS コマンドを熟知しているお客様向けの API です。

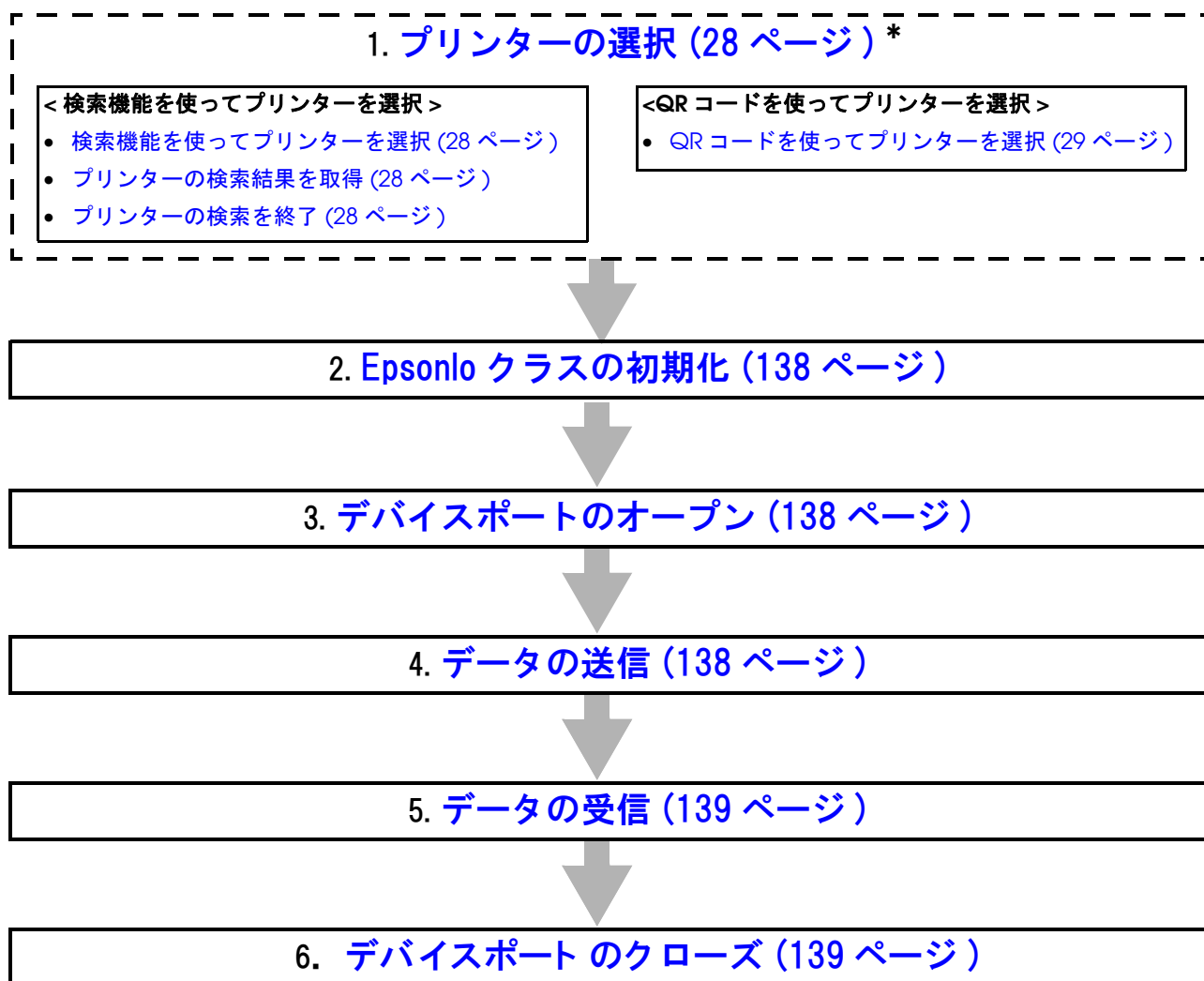


コマンド送受信 API は、ePOS-Print API の [EposPrint クラス \(45 ページ\)](#) と同時に使用できません。

プログラミング

プログラミングフロー

以下のフローでプログラミングします。



*: 任意のプロセスです。

EpsonIo クラスの初期化

[init \(141 ページ\)](#) を使って、EpsonIo クラスを初期化します。以下のプログラミングを参考にしてください。

```
//EpsonIo クラスの初期化
id port = [[EpsonIo alloc] init];
if ( port != nil ) {
    - 処理 -
    [port release];
}
```

デバイスポートのオープン

EpsonIo クラスの [open \(142 ページ\)](#) を使って、デバイスポートをオープンします。以下のプログラミングを参考にしてください。

```
//EpsonIo クラスの初期化
id port = [[EpsonIo alloc] init];
if ( port != nil ) {
    int errorStatus = EPSONIO_OC_SUCCESS;
    // デバイスポートのオープン
    errorStatus = [port open:EPSONIO_OC_DEVTYPE_TCP DeviceName:
        @"192.168.192.168" DeviceSettings:nil];
    if (EPSONIO_OC_SUCCESS == errorStatus ) {
        - 処理 -
    }
}
```

データの送信

EpsonIo クラスの [write \(144 ページ\)](#) を使って、プリンターにデータを送信します。以下のプログラミングを参考にしてください。

文字列「Hello, World!」を印刷する場合

```
// 送信設定
long sizeWritten;
int errStatus;
NSString *str = @"Hello, World!\r\n";
NSData *data = [str dataUsingEncoding:NSUTF8StringEncoding];

// データの送信
errStatus = [port write:data Offset:0 Size:[data length]
    Timeout:100 SizeWritten:& sizeWritten];
```

データの受信

EpsonIo クラスの [read \(146 ページ\)](#) を使って、プリンターからのデータを受信します。以下のプログラミングを参考にしてください。

```
// 受信設定
NSMutableData *data;
long sizeRead;
int errStatus;
data = [[NSMutableData alloc] initWithLength:256];

// データの受信
errStatus =
[port read:data Offset:0 Size:256 Timeout:100 SizeRead:& sizeRead];
```

デバイスポートのクローズ

EpsonIo クラスの [close \(143 ページ\)](#) を使って、デバイスポートをクローズします。以下のプログラミングを参考にしてください。

```
//EpsonIo クラスの初期化
id port = [[EpsonIo alloc] init];
if ( port != nil ) {
    int errorStatus = EPSONIO_OC_SUCCESS;
    // デバイスポートのオープン
    errorStatus = [port open:EPSONIO_OC_DEVTYPE_TCP DeviceName:
@"192.168.192.168" DeviceSettings:nil];
    if (EPSONIO_OC_SUCCESS == errorStatus ) {
        - 処理 -
    }
    // デバイスポートのクローズ
    errorStatus = [port close];
}
```

エラー値一覧

コマンド送受信 API には、以下のエラー値が定義されています。

エラー値	要因
EPSONIO_OC_SUCCESS	処理に成功した。
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された。 < 例 > <ul style="list-style-type: none">• Nil など、不正な引数を渡された。• サポートしていない範囲の値が指定された。
EPSONIO_OC_ERR_OPEN,	オープン処理に失敗した < 例 > TCP 通信用の Socket の作成に失敗した。
EPSONIO_OC_ERR_CONNECT	デバイスとの通信に失敗した。 < 例 > <ul style="list-style-type: none">• タイムアウト以外の要因で、対象デバイスへのデータ送信に失敗した。• タイムアウト以外の要因で、対象デバイスからのデータ受信に失敗した。
EPSONIO_OC_ERR_TIMEOUT	指定されたタイムアウト時間を越えた。 < 例 > <ul style="list-style-type: none">• 指定されたサイズのデータを指定時間内に送信できなかった。• 指定された時間内に 1 バイトも受信できなかった。
EPSONIO_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPSONIO_OC_ERR_ILLEGAL	不適切な方法で使用された。 < 例 > <ul style="list-style-type: none">• デバイSPORTがオープンされていない状態でデータ送受信 API が呼び出された。• すでにプリンター検索が開始されている状態で、再度検索開始 API が呼び出された。
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった。 < 例 > 同様の処理を他のスレッドで実行中のため、共有リソースのロック権限を取得できなかった。
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した。

コマンド送受信 API リファレンス

コマンド送受信 API には以下のクラスが用意されています。

Epsonlo クラス

データ送受信用のクラスです。以下の API が用意されています。

API	説明	ページ
init	Epsonlo クラスのインスタンスを初期化	141
open	デバイスポートのオープン	142
close	デバイスポートのクローズ	143
write	データ送信	144
read	データ受信	146

init

生成された Epsonlo クラスのインスタンスを初期化します。

構文

```
- (id) init;
```

戻り値

初期化済みの Epsonlo クラスのインスタンスが返されます。

open

指定されたデバイスポートをオープンします。

構文

```
(int) open:(int)deviceType  
DeviceName:(NSString *)deviceName  
DeviceSettings:(NSString *)deviceSettings;
```

パラメーター

- deviceType: オープンするデバイス種別を指定します。以下の値を指定します。

設定値	説明
EPSONIO_OC_DEVTYPE_TCP	オープンするプリンターが Wi-Fi/Ethernet の時に指定します。
EPSONIO_OC_DEVTYPE_BLUETOOTH	オープンするプリンターが <i>Bluetooth</i> の時に指定します。

- deviceName: 対象デバイスを特定するための識別子を指定します。以下の値を指定します。

deviceType	設定値
EPSONIO_OC_DEVTYPE_TCP	以下のいずれかを指定できます。 <ul style="list-style-type: none">IPv4形式のIPアドレス (例: "192.168.192.168")MAC アドレス (例: "01:23:45:67:89:AB")プリンターホスト名 (任意の文字列)
EPSONIO_OC_DEVTYPE_BLUETOOTH	BD アドレス (例: "01:23:45:67:89:AB")

- deviceSettings (Reserved):
"nil" を指定します。

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した
EPSONIO_OC_ERR_OPEN	オープン処理に失敗した
EPSONIO_OC_ERR_ILLEGAL	すでにオープンされているデバイスを再度オープンしようとした
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

close

指定されたデバイスポートをクローズします。

構文

```
- (int) close;
```

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した
EPSONIO_OC_ERR_ILLEGAL	オープンしていない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

write

データをデバイスポートへ送信します。



Bluetooth で本 API を使ってデータ送信する場合、プリンターの印刷が終了するまで、[close \(143 ページ\)](#) API は使用しないでください。データ送信が中断されることがあります。

構文

```
- (int) write: (NSData *)data  
               Offset: (size_t)offset  
               Size: (size_t)size  
               Timeout: (long)timeout  
               SizeWritten: (size_t *)sizeWritten;
```

パラメーター

- data : 送信データのバッファです。送信するデータを格納します。
- offset : 送信開始位置を指定します。
送信データバッファの先頭からのオフセット値を指定してください。
- size : 送信したいデータのバイト数を指定します。



size に 0 が指定された場合、送信されません。この場合、sizeWritten に 0 が返ります。

- timeout : 送信待ちのタイムアウト時間を、msec 単位で指定します。
指定可能な最長時間は 600000 msec (10 分) です。



- timeout は、伝送速度、送信データ量などを考慮して指定してください。
- timeout が短い場合、正常にデータが送信できている間、全データを送信し終わるまで timeout を超えても送信処理を継続します。

- sizeWritten : 送信を終了したデータのバイト数が格納されます。



- sizeWritten で返されるサイズのデータが、実際にプリンターが受信しているとは限りません。
- timeout で指定した時間を過ぎた場合、その時点までに送信を終了したバイト数を sizeWritten に格納します。

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した
EPSONIO_OC_ERR_ILLEGAL	オープンしていない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_TIMEOUT	指定された時間内に全データを送信できなかった
EPSONIO_OC_ERR_CONNECT	通信エラーが発生した
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

read

データをデバイスポートから受信します。



本 API は、受信エラーが発生するまでデータを受信し続けますが、timeout で指定された時間内に 1 バイトもデータが受信できなかった場合、処理が終了します。

構文

```
- (int) read: (NSMutableData *)data  
    Offset: (size_t)offset  
    Size: (size_t)size  
    Timeout: (long)timeout  
    SizeRead: (size_t *)sizeRead;
```

パラメーター

- data: 受信データの格納先バッファです。
- offset: 格納先バッファの格納開始位置を指定します。
受信データバッファの先頭からのオフセット値を指定します。
- size: 受信可能なバイト数を指定します。



size に 0 が指定された場合、受信されません。この場合、sizeRead に 0 が返ります。

- timeout: データ受信する時間を、msec 単位で指定します。
指定可能な最長時間は 600000 msec(10 分) です。
- sizeRead: 受信したデータのバイト数が格納されます。

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した
EPSONIO_OC_ERR_ILLEGAL	オープンしていない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_TIMEOUT	指定された時間内に 1 バイトもデータが受信できなかった
EPSONIO_OC_ERR_CONNECT	通信エラーが発生した
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

付録

プリンターごとのサポート API一覧

API	TM-P20 TM-P20 iOS Bluetooth モデル	TM-P60II TM-P60II iOS Bluetooth モデル	TM-T20II iOS Bluetooth モデル	TM-T70	TM-T70II TM-T70II iOS Bluetooth モデル	TM-T88V TM-T88V iOS Bluetooth モデル	TM-T90II
addTextAlign (48 ページ)	○	○	○	○	○	○	○
addTextLineSpace (49 ページ)	○	○	○	○	○	○	○
addTextRotate (50 ページ)	○	○	○	○	○	○	○
addText (51 ページ)	○	○	○	○	○	○	○
addTextLang (52 ページ)	○	○	○	○	○	○	○
addTextFont (53 ページ)	○	○	○	○	○	○	○
addTextSmooth (54 ページ)	○	○	○	○	○	○	○
addTextDouble (55 ページ)	○	○	○	○	○	○	○
addTextSize (56 ページ)	○	○	○	○	○	○	○
addTextStyle (57 ページ)	○	○	○	○	○	○	○
addTextPosition (59 ページ)	○	○	○	○	○	○	○
addFeedUnit (60 ページ)	○	○	○	○	○	○	○
addFeedLine (61 ページ)	○	○	○	○	○	○	○
addImage (62 ページ)	○	○	○	○	○	○	○
addImage(旧フォーマット) (65 ページ)	○	○	○	○	○	○	○
addImage(旧フォーマット) (68 ページ)	○	○	○	○	○	○	○
addLogo (70 ページ)	○	○	○	○	○	○	○
addBarcode (71 ページ)	○	○	○	○	○	○	○
addSymbol (76 ページ)	○	○	○	○	○	○	○
addPageBegin (81 ページ)	○	○	○	○	○	○	○
addPageEnd (82 ページ)	○	○	○	○	○	○	○
addPageArea (83 ページ)	○	○	○	○	○	○	○
addPageDirection (84 ページ)	○	○	○	○	○	○	○
addPagePosition (85 ページ)	○	○	○	○	○	○	○
addPageLine (86 ページ)	○	○					
addPageRectangle (88 ページ)	○	○					
addCut (89 ページ)	○	○	○	○	○	○	○
addPulse (90 ページ)			○	○	○	○	○
addSound (91 ページ)	○	○	○		○	○	
addSound(旧フォーマット) (93 ページ)	○	○	○		○	○	

API	TM-P20 TM-P20 iOS <i>Bluetooth</i> モデル	TM-P60II TM-P60II iOS <i>Bluetooth</i> モデル	TM-T20II iOS <i>Bluetooth</i> モデル	TM-T70	TM-T70II TM-T70II iOS <i>Bluetooth</i> モデル	TM-T88V TM-T88V iOS <i>Bluetooth</i> モデル	TM-T90II
addFeedPosition (95 ページ)	○	○					
addLayout (96 ページ)	○	○					
addCommand (98 ページ)	○	○	○	○	○	○	○

プリンター別サポート情報

TM-P20

		58 mm 仕様
解像度		203 x 203 dpi
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル
印字幅		384 ドット
印字桁数	フォント A	ANK 32 桁 / 漢字 16 桁
	フォント B	ANK 42 / 漢字 19 桁
	フォント C	ANK 42 桁 / 漢字 24 桁
	フォント D	ANK 38 桁
	フォント E	ANK 48 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット
	フォント B	ANK 9 x 24 ドット / 漢字 20 x 24 ドット
	フォント C	ANK 9 x 17 ドット / 漢字 16 x 16 ドット
	フォント D	ANK 10 x 24 ドット
	フォント E	ANK 8 x 16 ドット
文字のベースライン	フォント A	文字の上端から 21 ドット目
	フォント B	文字の上端から 21 ドット目
	フォント C	文字の上端から 16 ドット目
	フォント D	文字の上端から 21 ドット目
	フォント E	文字の上端から 15 ドット目
初期改行量		30 ドット
色指定		第 1 色
ページモード初期領域		384 x 2400 ドット
ページモード最大領域		384 x 2400 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded

	58 mm 仕様
2 次元シンボル	PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composite Symbology
用紙のカット	カット位置まで紙送り
ドロアーキック	非サポート
ブザー	オプション
バッテリー	サポート

バッテリーステータス

上位 8 ビット

バッテリーステータス	要因
0x30	AC アダプターが接続されている
0x31	AC アダプターが接続されていない

下位 8 ビット

バッテリーステータス	要因
0x30	バッテリー残量 0(リアルエンド)
0x31	バッテリー残量 1(ニアエンド)
0x32	バッテリー残量 2
0x33	バッテリー残量 3
0x34	バッテリー残量 4
0x35	バッテリー残量 5
0x36	バッテリー残量 6



バッテリーステータス取得不可能状態の場合、“0x0000”を返します。

TM-P20 iOS *Bluetooth* モデル

		58 mm 仕様
解像度		203 x 203 dpi
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル
印字幅		384 ドット
印字桁数	フォント A	ANK 32 桁 / 漢字 16 桁
	フォント B	ANK 42 / 漢字 19 桁
	フォント C	ANK 42 桁 / 漢字 24 桁
	フォント D	ANK 38 桁
	フォント E	ANK 48 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット
	フォント B	ANK 9 x 24 ドット / 漢字 20 x 24 ドット
	フォント C	ANK 9 x 17 ドット / 漢字 16 x 16 ドット
	フォント D	ANK 10 x 24 ドット
	フォント E	ANK 8 x 16 ドット
文字のベースライン	フォント A	文字の上端から 21 ドット目
	フォント B	文字の上端から 21 ドット目
	フォント C	文字の上端から 16 ドット目
	フォント D	文字の上端から 21 ドット目
	フォント E	文字の上端から 15 ドット目
初期改行量		30 ドット
色指定		第 1 色
ページモード初期領域		384 x 2400 ドット
ページモード最大領域		384 x 2400 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composite Symbology
用紙のカット		カット位置まで紙送り

	58 mm 仕様
ドロアーキック	非サポート
ブザー	オプション
バッテリー	サポート

バッテリーステータス

上位 8 ビット

バッテリーステータス	要因
0x30	AC アダプターが接続されている
0x31	AC アダプターが接続されていない

下位 8 ビット

バッテリーステータス	要因
0x30	バッテリー残量 0(リアルエンド)
0x31	バッテリー残量 1(ニアエンド)
0x32	バッテリー残量 2
0x33	バッテリー残量 3
0x34	バッテリー残量 4
0x35	バッテリー残量 5
0x36	バッテリー残量 6



バッテリーステータス取得不可能状態の場合、“0x0000”を返します。

TM-P60II

		58 mm 仕様	60 mm 仕様
解像度		203 x 203 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		420 ドット	432 ドット
印字桁数	フォント A	ANK 35 桁 / 漢字 17 桁	ANK 36 桁 / 漢字 18 桁
	フォント B	ANK 42 桁	ANK 43 桁
	フォント C	ANK 52 桁	ANK 54 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 10 x 24 ドット	
	フォント C	ANK 8 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 21 ドット目	
	フォント C	文字の上端から 15 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		420 x 1624 ドット	432 x 1624 ドット
ページモード最大領域		420 x 1624 ドット	432 x 1624 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composit Symbology	
用紙のカット		カット / フィードカット	
ドロアーキック		非サポート	
ブザー		オプション	
バッテリー		サポート	

バッテリーステータス

上位 8 ビット

バッテリーステータス	要因
0x30	AC アダプターが接続されている
0x31	AC アダプターが接続されていない

下位 8 ビット

バッテリーステータス	要因
0x30	バッテリー残量 0(リアルエンド)
0x31	バッテリー残量 1(ニアエンド)
0x32	バッテリー残量 2
0x33	バッテリー残量 3
0x34	バッテリー残量 4
0x35	バッテリー残量 5
0x36	バッテリー残量 6



バッテリーステータス取得不可能状態の場合、“0x0000”を返します。

TM-P60II iOS *Bluetooth* モデル

		58 mm 仕様	60 mm 仕様
解像度		203 x 203 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		420 ドット	432 ドット
印字桁数	フォント A	ANK 35 桁 / 漢字 17 桁	ANK 36 桁 / 漢字 18 桁
	フォント B	ANK 42 桁	ANK 43 桁
	フォント C	ANK 52 桁	ANK 54 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 10 x 24 ドット	
	フォント C	ANK 8 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 21 ドット目	
	フォント C	文字の上端から 15 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		420 x 1624 ドット	432 x 1624 ドット
ページモード最大領域		420 x 1624 ドット	432 x 1624 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composit Symbology	
用紙のカット		カット / フィードカット	
ドロアーキック		非サポート	
ブザー		オプション	
バッテリー		サポート	

バッテリーステータス

上位 8 ビット

バッテリーステータス	要因
0x30	AC アダプターが接続されている
0x31	AC アダプターが接続されていない

下位 8 ビット

バッテリーステータス	要因
0x30	バッテリー残量 0(リアルエンド)
0x31	バッテリー残量 1(ニアエンド)
0x32	バッテリー残量 2
0x33	バッテリー残量 3
0x34	バッテリー残量 4
0x35	バッテリー残量 5
0x36	バッテリー残量 6



バッテリーステータス取得不可能状態の場合、“0x0000” を返します。

TM-T20II iOS *Bluetooth* モデル

		58 mm 仕様	80 mm 仕様
解像度		203 x 203 dpi	
言語		日本語モデル	
印字幅		420 ドット	576 ドット
印字桁数	フォント A	ANK 35 桁 / 漢字 17 桁	ANK 48 桁 / 漢字 24 桁
	フォント B	ANK 46 桁 / 漢字 23 桁	ANK 64 桁 / 漢字 32 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 9 x 17 ドット / 漢字 16 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 16 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		420 x 831 ドット	576 x 831 ドット
ページモード最大領域		420 x 1662 ドット	576 x 1662 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composite Symbolology	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		オプション	
バッテリー		非サポート	

TM-T70

		58 mm 仕様	80 mm 仕様
解像度		203 x 203 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		416 ドット	576 ドット
印字桁数	フォント A	ANK 34 桁 / 漢字 17 桁	ANK 48 桁 / 漢字 24 桁
	フォント B	ANK 52 桁 / 漢字 26 桁	ANK 72 桁 / 漢字 36 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 8 x 16 ドット / 漢字 16 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 15 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		416 x 1662 ドット	576 x 1662 ドット
ページモード最大領域		416 x 1662 ドット	576 x 1662 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128	
2 次元シンボル		QR Code	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		非サポート	
バッテリー		非サポート	

TM-T70II

		58 mm 仕様	80 mm 仕様
解像度		203 x 203 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		416 ドット	576 ドット
印字桁数	フォント A	ANK 34 桁 / 漢字 17 桁	ANK 48 桁 / 漢字 24 桁
	フォント B	ANK 52 桁 / 漢字 26 桁	ANK 72 桁 / 漢字 36 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 9 x 17 ドット / 漢字 16 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 15 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		416 x 1662 ドット	576 x 1662 ドット
ページモード最大領域		416 x 1662 ドット	576 x 1662 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		サポート	
バッテリー		非サポート	

TM-T70II iOS *Bluetooth* モデル

		58 mm 仕様	80 mm 仕様
解像度		203 x 203 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		416 ドット	576 ドット
印字桁数	フォント A	ANK 34 桁 / 漢字 17 桁	ANK 48 桁 / 漢字 24 桁
	フォント B	ANK 52 桁 / 漢字 26 桁	ANK 72 桁 / 漢字 36 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 9 x 17 ドット / 漢字 16 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 15 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		416 x 1662 ドット	576 x 1662 ドット
ページモード最大領域		416 x 1662 ドット	576 x 1662 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		サポート	
バッテリー		非サポート	

TM-T88V

		58 mm 仕様	80 mm 仕様
解像度		180 x 180 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		360 ドット	512 ドット
印字桁数	フォント A	ANK 30 桁 / 漢字 15 桁	ANK 42 桁 / 漢字 21 桁
	フォント B	ANK 40 桁	ANK 56 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 9 x 17 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 16 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		360 x 831 ドット	512 x 831 ドット
ページモード最大領域		360 x 1662 ドット	512 x 1662 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composit Symbolology 非サポート)	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		オプション	
バッテリー		非サポート	

TM-T88V iOS *Bluetooth* モデル

		58 mm 仕様	80 mm 仕様
解像度		180 x 180 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		360 ドット	512 ドット
印字桁数	フォント A	ANK 30 桁 / 漢字 15 桁	ANK 42 桁 / 漢字 21 桁
	フォント B	ANK 40 桁	ANK 56 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 9 x 17 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 16 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		360 x 831 ドット	512 x 831 ドット
ページモード最大領域		360 x 1662 ドット	512 x 1662 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composit Symbolology 非サポート)	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		オプション	
バッテリー		非サポート	

TM-T90II

		58 mm 仕様	80 mm 仕様
解像度		203 x 203 dpi	
言語		日本語モデル	
印字幅		420 ドット	576 ドット
印字桁数	フォント A	ANK 35 桁 / 漢字 17 桁	ANK 48 桁 / 漢字 24 桁
	フォント B	ANK 42 桁 / 漢字 21 桁	ANK 57 桁 / 漢字 28 桁
	フォント C	ANK 52 桁 / 漢字 26 桁	ANK 72 桁 / 漢字 36 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 10 x 24 ドット / 漢字 20 x 24 ドット	
	フォント C	ANK 8 x 16 ドット / 漢字 16 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 21 ドット目	
	フォント C	文字の上端から 15 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		420 x 1662 ドット	576 x 1662 ドット
ページモード最大領域		420 x 1662 ドット	576 x 1662 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		サポート	
バッテリー		非サポート	

注意事項

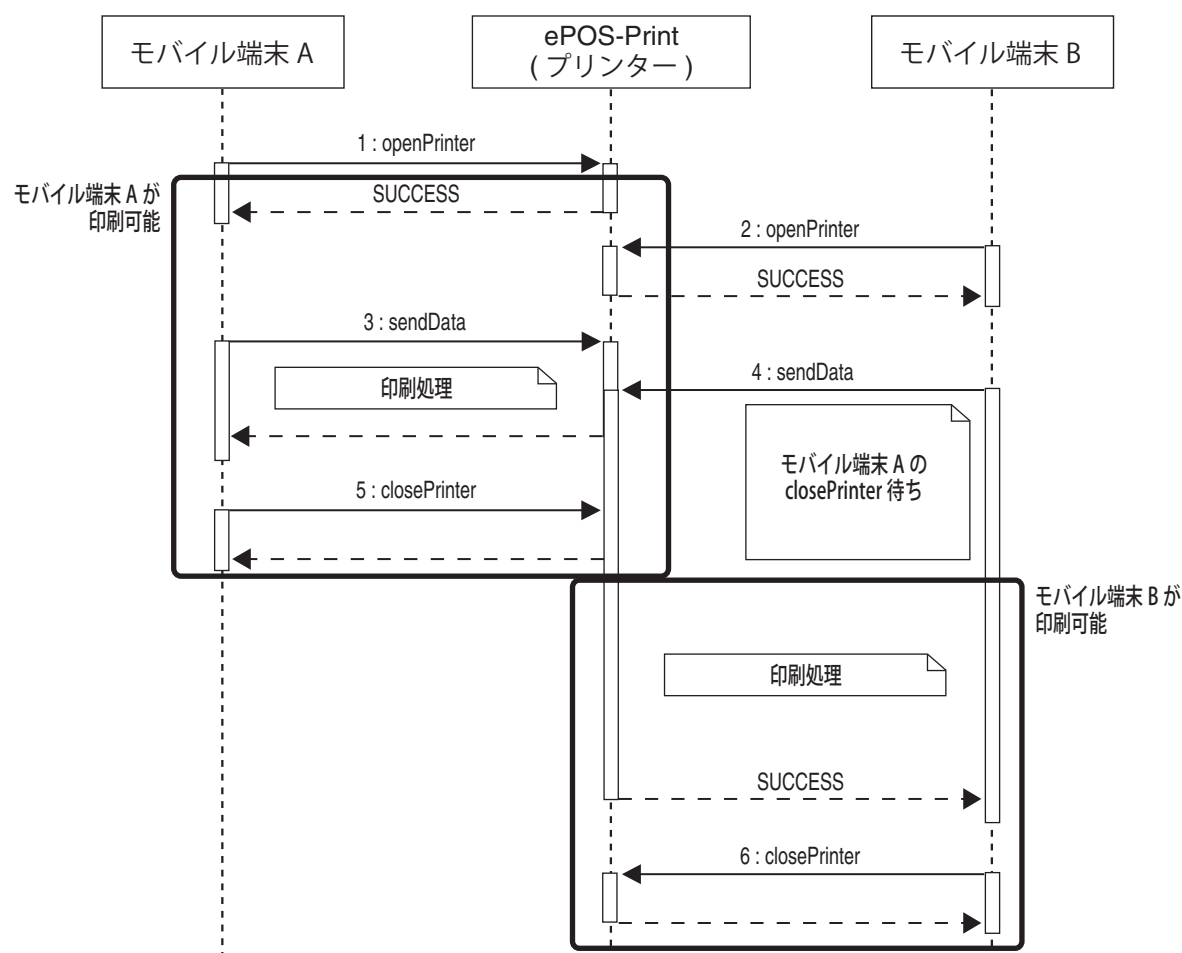
一台のプリンターを複数のモバイル端末から使用する場合

一台のプリンターを複数のモバイル端末から使用する場合、一台の端末から使用している間は他の端末からは印刷ができません。Version 1.6.0 以降では、別の端末によってプリンターが使用されている時に、その処理の終了を openPrinter の処理の中で待つようになります。

以下の図は、モバイル端末 A とモバイル端末 B から 1 台のプリンターを使用する場合の処理の流れを示しています。

Version 1.5.0 以前

Version 1.5.0 以前では、モバイル端末 B は sendData の処理の中でモバイル端末 A の closePrinter 処理の終了を待ちます。



Version 1.6.0 以降

Version 1.6.0 以降では、モバイル端末 B は openPrinter の処理の中でモバイル端末 A の closePrinter 処理の終了を待ちます。

