

CITIZEN

iOS Label Print SDK (Swift) プログラムマニュアル

Ver. 1.05 用

シチズン・システムズ株式会社

更新履歴

年月日	バージョン	履歴
2018/08/28	1.00	初版
2018/10/10	1.01	<ul style="list-style-type: none"> ・SDK 種類に Swift4.2 用のフレームワークを追加 ・drawTextPtrFont に中国モデル、韓国モデル用のロケールを追加 ・ロケールの定義に中国モデル、韓国モデルを追加
2019/02/06	1.02	バージョン番号のみ更新
2019/04/04		・SDK 種類に Swift5.0 用のフレームワークを追加
2019/08/08		<ul style="list-style-type: none"> ・drawTextLocalFont メソッドの resolution、measurementUnit パラメータの設定内容を修正 ・drawBitmap メソッドの measurementUnit パラメータの設定内容を修正
2019/10/07		・SDK 種類に Swift5.1 用のフレームワークを追加
2019/10/31	1.03	・新モデル CL-S700II/703II、CL-S620II/630II、CL-S520II/530II を追加
2019/11/08		・SDK 種類に Swift5.1.2 用のフレームワークを追加
2020/03/16		・SDK 種類に Swift5.1 以降用のフレームワークを追加
2020/06/02		・新モデル CL-E300EX/303EX、CL-E321EX/331EX を追加
2021/02/03		・「1.3 定義方法」の説明を、XCFramework 形式に変更
2021/11/29	1.04	<ul style="list-style-type: none"> ・定義方法に LAN モデルの検索を行うための設定 (Bonjour) を追加 ・drawTextLocalFont メソッドのイタリック指定についての説明を追加
2022/12/16		・SDK の種類に Swift5.3(iOS14/iPadOS14)以降用 XCFramework を追加
2023/12/22		・新モデル CL-S700III/703III を追加
2024/01/19		・プライバシーマニフェスト対応
2024/09/20	1.05	<ul style="list-style-type: none"> ・対象モデルのインターフェースに USB(Lightning/Type-C)を追加 (6 頁) ・connect メソッド、searchCitizenPrinter メソッド、searchLabelPrinter メソッドの接続タイプに USB(Lightning/Type-C)を追加 (17,27,29 頁) ・drawTextPtrFont および drawTextDLFont メソッドの UTF-8 対応 (59,61 頁)

ご注意

1. 本書の内容の一部、または全部を無断で転載することは、固くお断りいたします。
2. 本書の内容については、事前の予告なしに変更することがあります。
3. 本書の内容については万全を期して作成いたしましたが、万一誤り・お気付きの点がございましたら、ご連絡くださいますようお願いいたします。
4. 運用した結果の影響につきましては、3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。
5. 上記に同意いただけない場合は、本SDKをご使用いただけません。

商標

iOS は米国およびその他の国における Cisco の商標または登録商標です。

iPod touch、Swift、Xcode は米国およびその他の国における Apple Inc.の商標または登録商標です。

その他、記載されている会社名、製品名は、各社の商標または登録商標です。

CITIZEN は、シチズン時計株式会社の登録商標です。

目 次

1. 概要	6
1.1. ドキュメント対象範囲	6
1.2. システム概要	6
1.3. 定義方法	8
1.4. Bluetooth の利用方法	10
1.5. 機能一覧	12
2. SDK インターフェイス	15
2.1. 戻り値	15
2.2. LabelPrinter クラス	16
2.2.1 コンストラクタ	16
2.2.2 connect メソッド	17
2.2.3 disconnect メソッド	19
2.2.4 printerCheck メソッド	20
2.2.5 print メソッド	22
2.2.6 storeNVBitmap メソッド	23
2.2.7 clearOutput メソッド	25
2.2.8 sendData メソッド	26
2.2.9 searchCitizenPrinter メソッド	27
2.2.10 searchLabelPrinter メソッド	29
2.2.11 setLog メソッド	30
2.2.12 HorizontalMagnification プロパティ	31
2.2.13 VerticalMagnification プロパティ	32
2.2.14 FormatAttribute プロパティ	33
2.2.15 ContinuousMediaLength プロパティ	34
2.2.16 MeasurementUnit プロパティ	35
2.2.17 PrintSpeed プロパティ	36
2.2.18 FeedSpeed プロパティ	37
2.2.19 SlewSpeed プロパティ	38
2.2.20 BackupSpeed プロパティ	39
2.2.21 PrintDarkness プロパティ	40
2.2.22 DoubleHeat プロパティ	41
2.2.23 VerticalOffset プロパティ	42
2.2.24 HorizontalOffset プロパティ	43
2.2.25 MediaHandling プロパティ	44
2.2.26 StartOffset プロパティ	45
2.2.27 StopOffset プロパティ	46
2.2.28 LabelSensor プロパティ	47
2.2.29 PrintMethod プロパティ	48
2.2.30 SensorLocation プロパティ	49
2.2.31 CommandInterpreterInAction プロパティ	50
2.2.32 PaperError プロパティ	51
2.2.33 RibbonEnd プロパティ	52
2.2.34 BatchProcessing プロパティ	53
2.2.35 Printing プロパティ	54
2.2.36 Pause プロパティ	55
2.2.37 WaitingForPeeling プロパティ	56
2.3. LabelDesign クラス	57

2.3.1	コンストラクタ	57
2.3.2	drawTextPtrFont メソッド	58
2.3.3	drawTextDLFont メソッド	60
2.3.4	drawTextLocalFont メソッド	62
2.3.5	drawNVBitmap メソッド	64
2.3.6	drawBitmap メソッド	65
2.3.7	drawBarCode メソッド	67
2.3.8	drawMaxiCode メソッド	72
2.3.9	drawPDF417 メソッド	73
2.3.10	drawDataMatrix メソッド	74
2.3.11	drawQRCode メソッド	75
2.3.12	drawAztec メソッド	76
2.3.13	drawGS1DataBar メソッド	77
2.3.14	drawLine メソッド	79
2.3.15	drawRect メソッド	80
2.3.16	fillRect メソッド	81
2.3.17	drawCircle メソッド	82
2.3.18	fillCircle メソッド	83
2.3.19	drawPolygon メソッド	84
2.3.20	fillPolygon メソッド	85
2.3.21	embedRawDesignCommand メソッド	86
3.	補足	87
3.1.	印字位置指定	87
3.2.	ログ機能について	88
3.3.	パラメータ	90

1. 概要

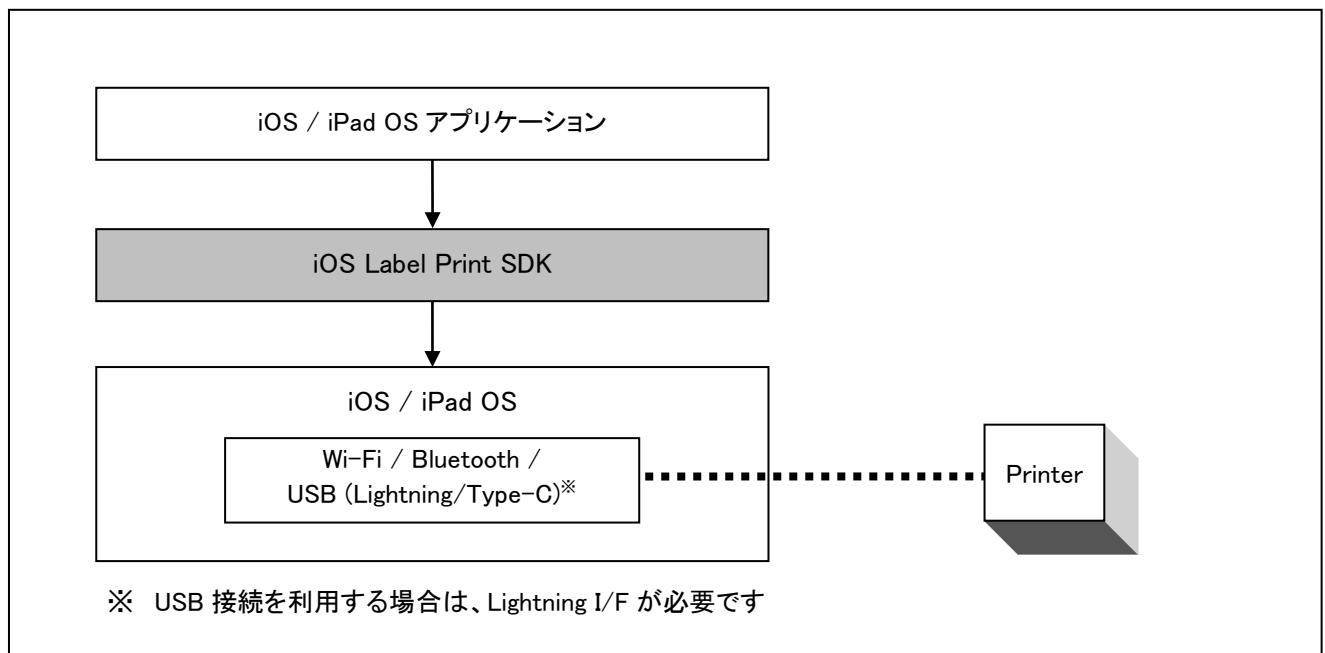
本ドキュメントは、iOS Label Print SDK のプログラムマニュアルです。

1.1. ドキュメント対象範囲

本ドキュメントは、CITIZEN Label プリンターを利用する iOS または iPad OS アプリケーション (Swift) の開発者が参照することを目的としています。

1.2. システム概要

本 SDK は CITIZEN Label プリンターを利用する iOS または iPad OS アプリケーションから参照されることを想定しています。



SDK システム構成図

対応端末

本 SDK が対応する端末とその開発環境の仕様は以下の通りです。

iOS バージョン	対応インターフェース	プライバシーマニフェスト対応
iOS 8.0 以上	Wi-Fi, Bluetooth	×
iOS 10.0.2 以上	USB(Lightning/Type-C)	×
iOS 12.0 以上	Wi-Fi, Bluetooth, USB(Lightning/Type-C)	○

対象モデル

本 SDK の対象モデルおよびそのモデルに対応するインターフェースは以下の通りです。
各モデルの機能詳細についてはプリンターの取扱説明書をご参照ください。

対象モデル	インターフェース	プリンター機能
CL-S700/703	有線/無線 LAN	感熱/熱転写
CL-S700II/703II	有線/無線 LAN	感熱/熱転写
CL-S700III/703III	有線/無線 LAN	感熱/熱転写
CL-E720/730	有線/無線 LAN, Bluetooth	感熱/熱転写
CL-S620/630	有線/無線 LAN	感熱/熱転写

CL-S620II/630II	有線/無線 LAN	感熱/熱転写
CL-S520/530	有線/無線 LAN	感熱
CL-S520II/630II	有線/無線 LAN	感熱
CL-S400DT	有線/無線 LAN, Bluetooth	感熱
CL-E300/CL-E303	有線 LAN	感熱
CL-E300EX/CL-E303EX	有線/無線 LAN, Bluetooth, USB(Lightning/Type-C)	感熱
CL-E321/CL-E331	有線 LAN	感熱/熱転写
CL-E321EX/CL-E331EX	有線/無線 LAN, Bluetooth, USB(Lightning/Type-C)	感熱/熱転写

1.3. 定義方法

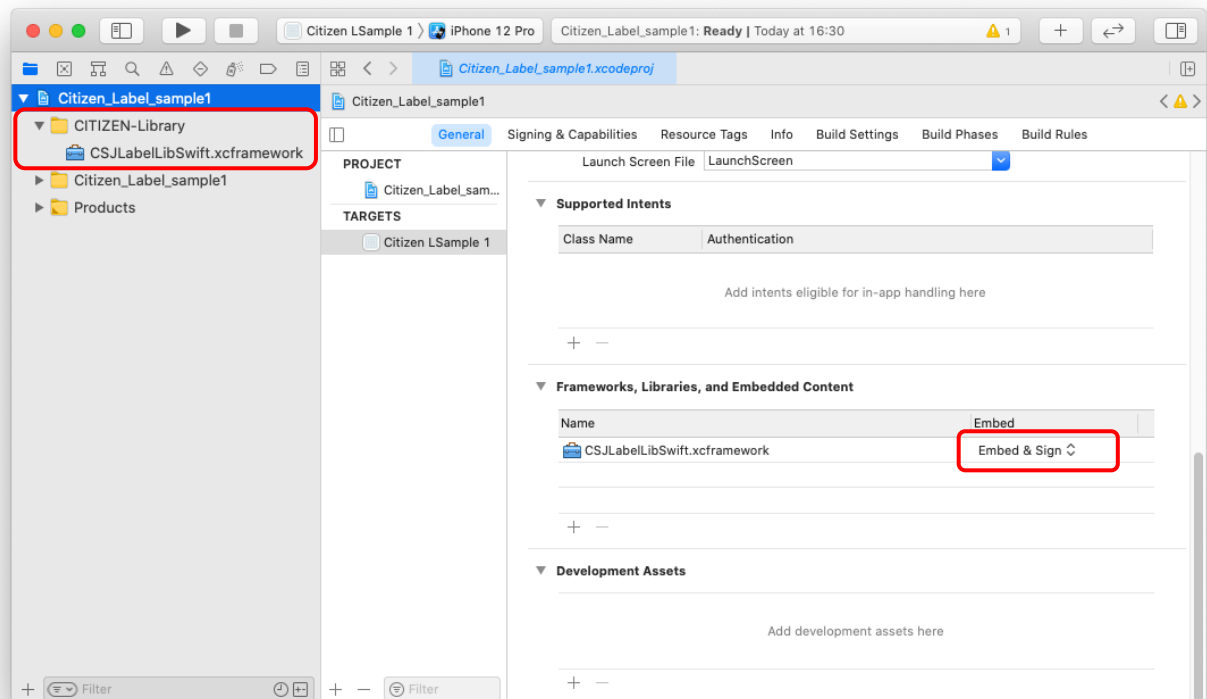
SDK 種類

SDK(フレームワーク)を開発環境に合わせて選択してください。

SDK (Framework)	開発環境	対象 OS
Swift5.1 以降 (XCFramework)	Xcode11.0 以降	使用する XCode に依存
Swift5.3 以降 (XCFramework)	Xcode12.0 以降	使用する XCode に依存
Swift5.9 以降 プライバシーマニフェスト対応版 (XCFramework)	Xcode15.0 以降	iOS12.0 以降

Xcode に SDK を追加

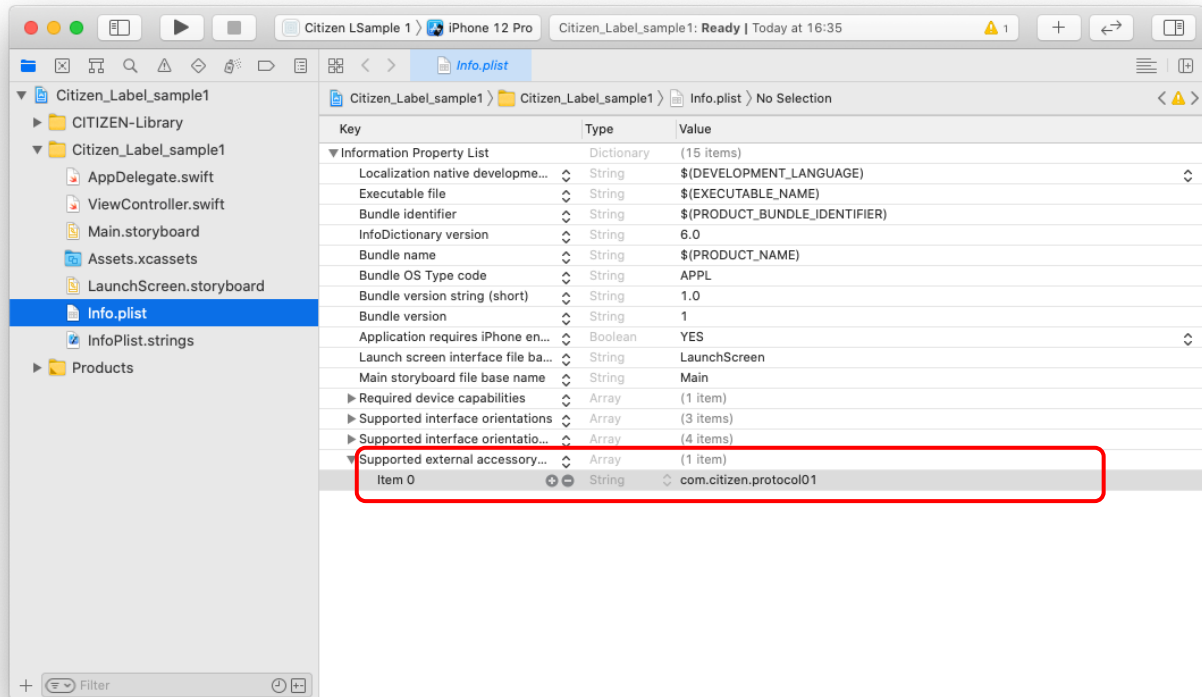
- ・開発するプロジェクトにフレームワーク (CSJLabelLibSwift.xcframework)を追加します。
- ・[Finder]から Xcode の[Project Navigator]の任意の場所へフレームワークをドラッグします。



- ・ [Project Navigator]からプロジェクトファイルを選択し、[TARGET] → [General]の[Frameworks, Libraries, and Embedded Content]欄の追加したフレームワークの[Embed]を[Embed & Sign]へ変更します。

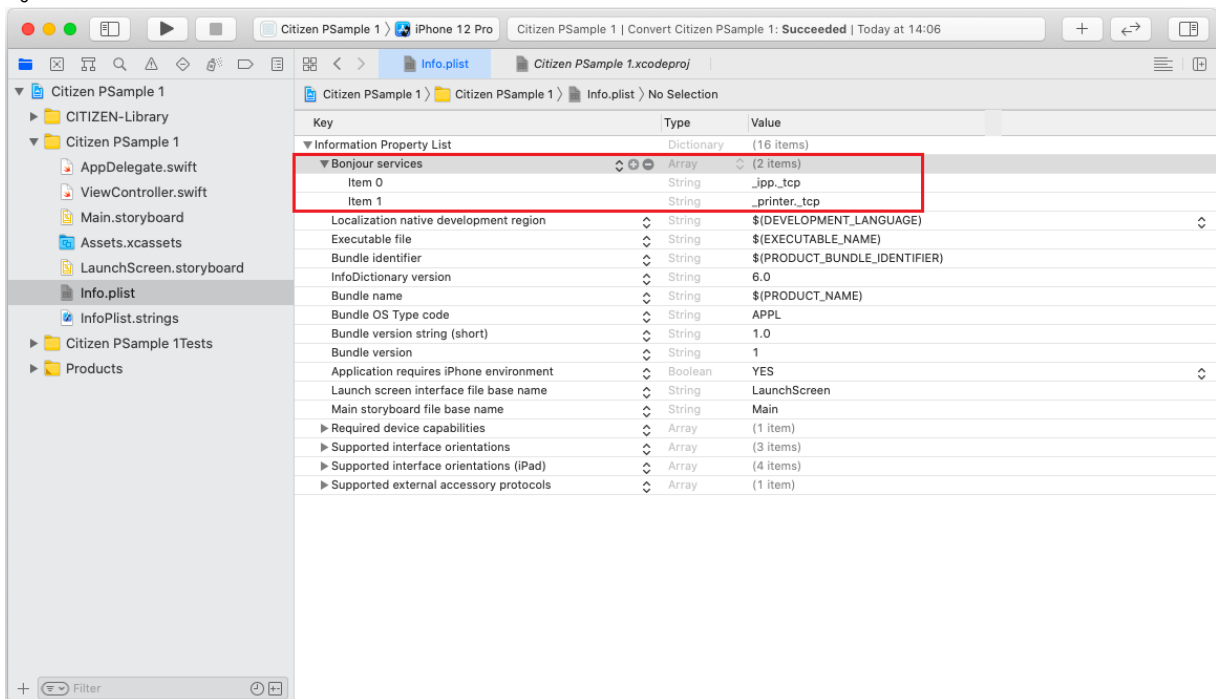
Bluetooth および USB(Lightning/Type-C)の開発を行うための設定

- ・Protocol name を追加します。[Project Navigator]から “Info.plist”ファイルを開きます。ポップアップメニュー (Control+クリックすると開きます)から Add Row を選び、“Supported external accessory protocols”を選択し追加します。その後“Supported external accessory protocols”を開き、Item0 の値に“com.citizen.protocol01”を入力します。



LAN モデルの検索を行うための設定 (Bonjour)

- ・V104 以上の SDK を利用して LAN の検索を行うメソッド (searchESCPOSPrinter, searchCitizenPrinter) を使用する場合は、App に Bonjour サービスを利用するための設定を追加する必要があります。
- ・[Project Navigator] から "Info.plist" ファイルを開きます。"Information Property List" をクリックして、"Bonjour service" を追加します。"Bonjour service" の "Item 0" に "_ipp_tcp"、"Item 1" に "_printer_tcp" を追加してください。



以上で準備完了です。

1.4. Bluetooth の利用方法

Bluetooth のペアリングと接続

プリンターと iOS デバイスで Bluetooth 通信をするには、iOS デバイスからプリンターを発見してペアリングをする必要があります。(以下の画面は iPod Touch/iOS11.2 での例です)

・プリンターの電源をオンにします。iOS デバイスの[設定]から[Bluetooth]を選び、周辺の Bluetooth 機器を検索します。Bluetooth がオフの場合、オンすると検索が始まります。



・しばらくすると周辺に有る Bluetooth 機器が表示されます。”機種名_(BD アドレスの下 2 桁)”と表示されている機器がプリンターです。初めて利用する場合ペアリングが必要です。既にペアリングされているプリンターは、そのまま接続が確立します。※



・ペアリングするには“機種名_(BD アドレスの下 2 桁)”の箇所をタップします。しばらくすると接続が確立します。



※再接続要求について

・プリンターのメモリスイッチが”再接続要求ー有効”の場合、プリンター側から iOS デバイスへの接続を復元します。iOS の仕様上、Bluetooth 接続が切断される度に、Bluetooth 設定画面より接続を行う必要があります。この機能を有効にすると、プリンター側が最後に接続していた iOS デバイスに対し、自動で接続を要求するため、この手間を省けます。

・複数の iOS デバイスとプリンターを交互に使用される場合は、この再接続要求によって却って利便性が低下しますので、無効の状態をご利用ください。

・iOS 以外のデバイスと接続される場合は、再接続要求を無効の状態をご利用ください。

1.5. 機能一覧

本 SDK は以下の機能を提供します。

LabelPrinter クラス メソッド一覧

No	機能	詳細
1	クラス生成 (インスタンス)	インスタンスです。
2	プリンター接続処理 (connect メソッド)	プリンターと接続します。
3	プリンター切断処理 (disconnect メソッド)	プリンターとの接続を切断します。
4	プリンター状態確認処理 (printerCheck メソッド)	プリンターのステータスを取得します。
5	印刷処理 (print メソッド)	デザインしたラベルを印刷します。
6	NV ビットマップ登録処理 (storeNVBitmap メソッド)	ビットマップ画像をフラッシュメモリへ登録します。
7	プリントバッファクリア (clearOutput メソッド)	プリンターのプリントバッファをクリアします。
8	バイトデータ送信処理 (sendData メソッド)	バイトデータをプリンターへ送信します。
9	プリンター検索 (searchCitizenPrinter メソッド)	プリンターを検索しプリンター情報のリストを取得します。
10	プリンター検索2 (searchLabelPrinter メソッド)	プリンターを検索しアドレスのリストを取得します。
11	ログ設定 (setLog メソッド)	ログ機能を設定します。

LabelPrinter クラス プロパティ一覧

No	機能	属性	詳細
1	水平ピクセルサイズの設定/取得 (HorizontalMagnification プロパティ)	R/W	水平方向のピクセルサイズを示します。
2	垂直ピクセルサイズの設定/取得 (VerticalMagnification プロパティ)	R/W	垂直方向のピクセルサイズを示します。
3	展開方法の指定設定/取得 (FormatAttribute プロパティ)	R/W	描画が重なった部分の重ね書き/白抜きを示します。
4	連続用紙長の設定/取得 (ContinuousMediaLength プロパティ)	R/W	連続紙を使用した場合のラベル長を示します。
5	単位の設定/取得 (MeasurementUnit プロパティ)	R/W	距離指定パラメータの単位を示します。
6	速度(印刷部分)の設定/取得 (PrintSpeed プロパティ)	R/W	印刷速度を示します。
7	速度(非印刷部分)の設定/取得 (FeedSpeed プロパティ)	R/W	非印刷部分の紙送り速度を示します。
8	速度(紙送り部分)の設定/取得 (SlewSpeed プロパティ)	R/W	紙送り部分の紙送り速度を示します。
9	速度(バックフィード)の設定/取得 (BackupSpeed プロパティ)	R/W	バックフィード時の速度を示します。
10	印字濃度の設定/取得 (PrintDarkness プロパティ)	R/W	印字濃度を示します。
11	ダブルヒートの設定/取得 (DoubleHeat プロパティ)	R/W	ダブルヒート(同位置に 2 回熱をかけ、より濃く印字)を示します。

12	Y 座標オフセットの設定/取得 (VerticalOffset プロパティ)	R/W	印字開始位置における、Y 座標のオフセットを示します。
13	X 座標オフセットの設定/取得 (HorizontalOffset プロパティ)	R/W	印字開始位置における、X 座標のオフセットを示します。
14	印刷後動作の設定/取得 (MediaHandling プロパティ)	R/W	印刷後のポーズ、カッター、剥離の動作を示します。
15	開始オフセットの設定/取得 (StartOffset プロパティ)	R/W	印字開始位置の設定を示します。
16	終了オフセットの設定/取得 (StopOffset プロパティ)	R/W	印刷後のフィード量を示します。
17	センサー選択(透過/反射/なし)の設定/取得 (LabelSensor プロパティ)	R/W	用紙センサーを示します。
18	印字方法の設定/取得 (PrintMethod プロパティ)	R/W	印字方法(感熱、熱転写)の選択を示します。
19	センサー選択(前方/後方)の設定/取得 (SensorLocation プロパティ)	R/W	前方センサー、後方センサーの選択を示します。
20	ステータス(コマンドインタプリタ動作中)の取得 (CommandInterpreterInAction プロパティ)	R	コマンド処理中のステータスを示します。
21	ステータス(ペーパーエラー)の取得 (PaperError プロパティ)	R	ペーパーエラーのステータスを示します。
22	ステータス(リボンエンド)の取得 (RibbonEnd プロパティ)	R	リボンエンドのステータスを示します。
23	ステータス(バッチ処理印字中)の取得 (BatchProcessing プロパティ)	R	バッチ処理印字中のステータスを示します。
24	ステータス(印字中)の取得 (Printing プロパティ)	R	印字中のステータスを示します。
25	ステータス(ポーズ中)の取得 (Pause プロパティ)	R	ポーズ中のステータスを示します。
26	ステータス(剥離待ち中)の取得 (WaitingForPeeling プロパティ)	R	剥離待ち中のステータスを示します。

LabelDesign クラス メソッド一覧

No	機能	詳細
1	クラス生成 (インスタンス)	インスタンスです。
2	テキスト描画(プリンタフォント) (drawTextPtrFont メソッド)	プリンターのフォントを使用して、テキストを描画します。
3	テキスト描画(プリンタダウンロードフォント) (drawTextDLFont メソッド)	プリンターにダウンロードしたフォントを使用して、テキストを描画します。
4	テキスト描画(Local フォント) (drawTextLocalFont メソッド)	端末のフォントを使用して、テキストを描画します。
5	画像描画(プリンタ格納画像) (drawNVBitmap メソッド)	プリンターに格納している画像を描画します。
6	画像描画(ローカル画像ファイル) (drawBitmap メソッド)	PC 上の画像ファイルを描画します。
7	バーコード描画(1D) (drawBarCode メソッド)	バーコードを描画します。
8	バーコード描画(UPS MaxiCode) (drawMaxiCode メソッド)	
9	バーコード描画(PDF417) (drawPDF417 メソッド)	
10	バーコード描画(DataMatrix) (drawDataMatrix メソッド)	
11	バーコード描画(QRCode) (drawQRCode メソッド)	
12	バーコード描画(Aztec) (drawAztec メソッド)	
13	バーコード描画(GS1DataBar(RSS)) (drawGS1DataBar メソッド)	
14	線の描画 (drawLine メソッド)	線を描画します。
15	矩形の描画 (drawRect メソッド)	矩形を描画します。
16	矩形の描画(塗りつぶし) (fillRect メソッド)	矩形(塗りつぶし)を描画します。
17	円の描画 (drawCircle メソッド)	円を描画します。
18	円の描画(塗りつぶし) (fillCircle メソッド)	円(塗りつぶし)を描画します。
19	多角形の描画 (drawPolygon メソッド)	多角形を描画します。
20	多角形の描画(塗りつぶし) (fillPolygon メソッド)	多角形(塗りつぶし)を描画します。
21	生デザインコマンド挿入 (embedRawDesignCommand メソッド)	デザインコマンドを挿入します。

2. SDK インターフェイス

本 SDK のインターフェイスを以下に示します。

2.1. 戻り値

以降に示すメソッドは、下記の戻り値を返します。

戻り値	説明
CLS_SUCCESS (0)	正常終了。
CLS_E_CONNECTED (1001)	プリンターへ既に接続済みです。
CLS_E_DISCONNECT (1002)	プリンターへ接続していません。
CLS_E_NOTCONNECT (1003)	プリンターへ接続できませんでした。
CLS_E_CONNECT_NOTFOUND (1004)	プリンター接続後の対応機種確認に失敗しました。
CLS_E_ILLEGAL (1101)	サポートされていない処理または無効なパラメータ値です。
CLS_E_OFFLINE (1102)	プリンターがオフラインです。
CLS_E_NOEXIST (1103)	指定のファイルが存在しません。
CLS_E_FAILURE (1104)	要求された処理が実行できません。
CLS_E_TIMEOUT (1105)	所定の時間が経過してもプリンターからの応答がありません。
CLS_E_NO_LIST (1106)	プリンター検索にてプリンターが見つかりません。
CLS_EPTR_BADFORMAT (1203)	指定されたファイルの書式がサポートされていません。

2.2. LabelPrinter クラス

2.2.1 コンストラクタ

形式

LabelPrinter

説明

クラスを生成し、初期化します。

使用例

```
var : printer? = CSJLabelLibSwift.LabelPrinter()
```


2.2.2 connect メソッド

形式

- 1) func connect(connectType: Int32, withAddress addr: String?) -> Int32
- 2) func connect(connectType: Int32, withAddress addr: String?, withPort port: Int32) -> Int32
- 3) func connect(connectType: Int32, withAddress addr: String?, withPort port: Int32, withTimeout timeout: Int32) -> Int32

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
connectType	[IN]	接続タイプ	CLS_PORT_WiFi: Wi-Fi 接続 CLS_PORT_BLUETOOTH: Bluetooth 接続 CLS_PORT_USB: USB 接続 (Lightning/Type-C)
addr	[IN]	接続先の IP アドレス または Bluetooth デバイスアドレス または Bluetooth デバイス名 またはシリアル番号	WiFi : 0.0.0.0 ~ 255.255.255.255 Bluetooth : 00:00:00:00:00:00 ~ FF:FF:FF:FF:FF:FF デバイス名 USB : 空白またはシリアル番号
port	[IN]	接続先ポート番号	
timeout	[IN]	タイムアウト (msec)	

説明

このメソッドは、プリンターと接続するために使用します。プリンターの接続タイプとアドレスを指定してください。Bluetooth の場合、iOS デバイスの Bluetooth 設定画面にてペアリング済でかつ接続された状態のプリンターにだけ接続する事が出来ます(「1.6 Bluetooth の利用方法」を参照してください)。また Bluetooth デバイスアドレスを利用して接続するには iOS6 以上である必要があります。それ以外は Bluetooth デバイス名を利用して接続してください。

USB 接続の場合、アドレスに空白を指定した場合は自動的に接続されます。プリンターのシリアル番号を指定して特定のプリンターへ接続する事も可能です。

接続先ポート番号は、接続タイプに Wi-Fi を指定した場合のみ有効です。省略された場合は、9100 番で接続します。

タイムアウトは、プリンターへの接続の最大時間(ミリ秒単位)を指定します。省略された場合は、Wi-Fi の場合は 4000 ミリ秒、Bluetooth の場合は 8000 ミリ秒で接続します。

Bluetooth 接続の場合、タイムアウトの推奨値は 8000 ミリ秒以上です。

プリンターと接続した際に、プリンターのステータスと対応機種を同時に確認します。

プリンターとの通信が不要になった場合は、必ず [disconnect メソッド](#) を実行し、プリンターとの接続を切断してください。切断しなかった場合は、次の接続がエラーとなります。

戻り値

成功時は CLS_SUCCESS(0) を返します。失敗時は下記のエラーコードの説明を確認してください。

それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

エラーコード	説明
CLS_E_CONNECTED (1001)	既にプリンターと接続済みです。
CLS_E_NOTCONNECT (1003)	プリンターへ接続できませんでした。 ①プリンターが未接続 ②プリンターの電源が入っていない

	③インターフェースポートのハンドルを取得できない
CLS_E_CONNECT_NOTFOUND (1004)	プリンター接続後の対応機種確認に失敗しました。 ①対応機種でない

使用例

```
printer!.connect(CLS_PORT_WiFi, withAddress: "192.168.182.100", withPort: 9100)
```

```
printer!.connect(CLS_PORT_BLUETOOTH, withAddress: "00:01:90:F0:81:AB",  
withPort: 9100, withTimeout:8000)
```

```
printer!.connect(CLS_PORT_BLUETOOTH, withAddress: "CITIZEN SYSTEMS",  
withPort: 9100, withTimeout:8000)
```

```
printer!.connect(CLS_PORT_USB, withAddress: "")
```

2.2.3 disconnect メソッド

形式

```
func disconnect() -> Int32
```

パラメータ

ありません。

説明

このメソッドは、プリンターとの接続を切断するために使用します。

印刷の終了、あるいは、何らかのエラーが発生した場合は、本メソッドを実行して接続を切断してください。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
printer!.disconnect()
```

2.2.4 printerCheck メソッド

形式

```
func printerCheck() -> Int32
```

パラメータ

ありません。

説明

このメソッドは、プリンターのステータスを取得し、ステータス情報を [CommandInterpreterInAction](#)、[PaperError](#)、[RibbonEnd](#)、[BatchProcessing](#)、[Printing](#)、[Pause](#)、[WaitingForPeeling](#) プロパティに格納します。

本メソッドの実行結果が失敗の場合は、通信異常やデバイスの異常が発生した可能性があります。この場合、[disconnect メソッド](#)および [connect メソッド](#)を使用して再接続してください。

接続後に時間を空けて印刷する場合は、必ず事前に本メソッドを実行し、プロパティを取得してプリンターのステータスを確認してください。

ネットワーク接続の場合、長時間放置すると自動的に切断されます。接続を保持する場合は、定期的に本メソッドを実行してください。

ネットワーク接続の場合、CL-S5xx/6xx/70x 系のプリンターは、予め「Parallel Error Output」の設定を OFF に変更しておいてください。パラレル接続の場合、全機種において「Parallel Error Output」の設定を OFF に変更しておいてください。設定の変更は、ラベルプリンターユーティリティの「拡張」タブから行うことができます。

戻り値

成功時は CLS_SUCCESS(0)を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
// printerCheck
if CLS_SUCCESS == printer!.printerCheck() {

    // CommandInterpreterInActionプロパティ
    if 1 == printer!.getCommandInterpreterInAction() {
        // Command interpreter in action
    }

    // PaperErrorプロパティ
    if 1 == printer!.getPaperError() {
        // Paper error
    }

    // RibbonEndプロパティ
    if 1 == printer!.getRibbonEnd() {
        // Ribbon end
    }

    // BatchProcessingプロパティ
    if 1 == printer!.getBatchProcessing() {
        // Batch processing
    }

    // Printingプロパティ
    if 1 == printer!.getPrinting() {
        // Printing
    }
}
```

```
    }

    // Pauseプロパティ
    if 1 == printer!.getPause() {
        // Pause
    }

    // WaitingForPeelingプロパティ
    if 1 == printer!.getWaitingForPeeling() {
        // Waitiong for peeling
    }
}
else
{
    // Fail
}
```

2.2.5 print メソッド

形式

```
func print (design: LabelDesign!, quantity: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
design	[IN]	LabelDesign クラスインスタンス	
quantity	[IN]	印刷枚数	1～9999

説明

このメソッドは、デザインしたラベルを印刷する際に使用します。プリンターへ印刷要求を送信します。印刷要求を行う際、LabelDesign クラスで指定されたラベルのフォーマットや、プロパティの設定をコマンドに反映します。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
design.fillCircle(50, y: 50, radius: 50, pattern: CLS_SHADED_PTN_11)
printer!.print(design, quantity: 1)
```

2.2.6 storeNVBitmap メソッド

形式

```
func storeNVBitmap (filePath: String!, name: String!, rotation: Int32, width: Int32,
    height: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
filePath	[IN]	対象ファイル名	
name	[IN]	格納名称	以下の条件を除いた ASCII コードが使用可能 ・先頭がアンダースコアは不可 ・以下の文字は使用不可(禁則文字) ¥ / : * ? " < >
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
width	[IN]	横サイズ(ピクセル単位)	
height	[IN]	縦サイズ(ピクセル単位)	

説明

このメソッドは、ビットマップのファイル名、格納名称、回転方向、横サイズ、縦サイズを指定して、プリンターのフラッシュメモリにビットマップ画像(ロゴ)を登録するために使用します。登録したロゴは、[drawNVBitmap メソッド](#)を使用して描画できます。登録可能なビットマップ形式は、BMP/GIF/EXIF/JPG/PNG/TIFF です。

登録されるビットマップ画像サイズは、指定された縦横サイズに合わせて、アスペクト比を維持した状態でリサイズします。

例: 以下の場合、リサイズ後の画像サイズは「200x50」ピクセルとなる。

画像サイズ: 400x100 ピクセル
横サイズ: 200
縦サイズ: 200

指定のサイズが 0 の場合、対象の画像サイズを基準とする。

例: 以下の場合、リサイズ後の画像サイズは「800x200」ピクセルとなる。

画像サイズ: 400x100 ピクセル
横サイズ: 0
縦サイズ: 200

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
// Resource内ファイル取得
let FileName = NSBundle.mainBundle().pathForResource("sample_1.jpg",
    ofType: nil)

// 画像登録(プリンターへの格納)
printer!.storeNVBitmap(FileName, name: "Sample", rotation: CLS_RT_NORMAL, ,
    width: 0, height: 0);

// 画像描画(プリンター側格納画像)
```

```
design.drawNVBitmap("Sample", hexp: 1, vexp: 1, x: 0, y: 0)

// 印刷
printer!.print(design, quantity: 1)
```


2.2.7 clearOutput メソッド

形式

```
func clearOutput() -> Int32
```

パラメータ

ありません。

説明

プリンターをリセットします。プリンター側では、電源 ON 時と同等の初期化処理が行われるため、未印字データもクリアされます。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
printer!.clearOutput()
```

2.2.8 sendData メソッド

形式

```
func sendData (data: UnsafeMutablePointer<Int8>, withLength length: UInt) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	送信データ	
length	[IN]	データ長	

説明

このメソッドは、バイトデータをそのままプリンターに送信するために使用します。
通常は必要ありませんが、プリンターのコマンドを直接送信したい場合に使用します。
ご使用の際は、他のメソッドに影響を与えない様に注意する必要があります。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
var data: [Int8] = [0x01, 0x23] // [01]# (リセット)

printer!.sendData(&data, withLength: 2)
```

2.2.9 searchCitizenPrinter メソッド

形式

```
class func searchCitizenPrinter(ifType: Int32, withSearchTime searchTime: Int32, result:
    UnsafeMutablePointer<Int32>) -> [AnyObject]?
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
ifType	[IN]	接続タイプ	CLS_PORT_WiFi: Wi-Fi 接続 CLS_PORT_BLUETOOTH: Bluetooth 接続 CLS_PORT_USB: USB 接続(Lightning/Type-C)
searchTime	[IN]	検索時間 (秒)	1～30
result	[OUT]	エラーコード	---

result には成功時は CLS_SUCCESS(0)を返します。失敗時は下記のエラーコードの説明を確認してください。それ以外のエラーコードは「2.1 戻り値」を参照してください。

エラーコード	説明
CLS_E_ILLEGAL (1101)	無効なパラメータ値です。 ①未対応の接続タイプを指定した ②範囲外の検索時間を指定した
CLS_E_NO_LIST (1106)	検索の結果、プリンターが発見出来ませんでした

説明

このメソッドは、プリンターを検索し、プリンターの情報リストを取得するために使用します。接続タイプと検索時間を指定して実行してください。

シミュレーター利用時は Wi-Fi 接続のみ使用可能です。

検索時間経過後に、結果を result パラメータにセットし、検索できたプリンターの情報を配列で返します。接続タイプが CLS_PORT_WiFi の場合、CL-S400/520/530/620/630/700/703 系のプリンターと、CL-E720/730/300/303/321/331 系のプリンターのみ検索可能です。検索時間の推奨値は 3 秒以上です。これより短い時間の場合、ネットワークの状態によっては検索漏れが発生する場合があります。接続タイプが CLS_PORT_BLUETOOTH および CLS_PORT_USB の場合、検索時間である searchTime に関係無く、検索は直ぐに完了します。

戻り値

成功した時は検索されたプリンターの情報リストが返されます。失敗した時は空のリストが返されます。プリンターの情報リストは CitizenPrinterInfo 型で格納されており、接続タイプによって取得出来る情報が異なります。

接続タイプ	CitizenPrinterInfo	取得出来る情報
CMP_PORT_WiFi	ipAddress	IP アドレス
	macAddress	MAC アドレス
	deviceName	(空)
	bdAddress	(空)
	usbSerialNo	(空)
CMP_PORT_BLUETOOTH	ipAddress	(空)
	macAddress	(空)
	deviceName	(空) / Bluetooth デバイス名 ※プリンタ機種に依存
	bdAddress	(空) / BD アドレス ※iOS6 以上
	usbSerialNo	(空)

CMP_PORT_USB	ipAddress	(空)
	macAddress	(空)
	deviceName	モデル名
	bdAddress	(空)
	usbSerialNo	シリアル番号

使用例

```
var result = Int32(0)
let array = printer.searchCitizenPrinter(CLS_PORT_WiFi, withSearchTime: 4,
    result: &result)!
for var i = 0; i < array.count; i++ {
    let prninfo = array[i] as? CitizenPrinterInfo
    println("IP Address : \(prninfo.ipAddress)")
    println("MAC Address : \(prninfo.macAddress)")
}
```

2.2.10 searchLabelPrinter メソッド

形式

```
class func searchLabelPrinter(ifType: Int32, withSearchTime searchTime: Int32, result:
    UnsafeMutablePointer<Int32>) -> [AnyObject]?
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
ifType	[IN]	接続タイプ	CLS_PORT_WiFi: Wi-Fi 接続 CLS_PORT_BLUETOOTH: Bluetooth 接続 CLS_PORT_USB: USB 接続(Lightning/Type-C)
searchTime	[IN]	検索時間 (秒)	1～30
result	[OUT]	エラーコード	---

result には成功時は CLS_SUCCESS(0)を返します。失敗時は下記のエラーコードの説明を確認してください。それ以外のエラーコードは「2.1 戻り値」を参照してください。

エラーコード	説明
CLS_E_ILLEGAL (1101)	無効なパラメータ値です。 ①未対応の接続タイプを指定した ②範囲外の検索時間を指定した
CLS_E_NO_LIST (1106)	検索の結果、プリンターが発見出来ませんでした

説明

このメソッドは、プリンターを検索し、アドレスのリストを取得するために使用します。接続タイプと検索時間を指定して実行してください。

シミュレーター利用時は Wi-Fi 接続のみ使用可能です。

検索時間経過後に、結果を result パラメータにセットし、プリンターの情報を NSArray 型で返します。

接続タイプが CLS_PORT_WiFi の場合、CL-S400/520/530/620/630/700/703 系のプリンターと、CL-E720/730/300/303/321/331 系のプリンターのみ検索可能です。検索時間の推奨値は 3 秒以上です。これより短い時間の場合、ネットワークの状態によっては検索漏れが発生する場合があります。

接続タイプが CLS_PORT_BLUETOOTH および CLS_PORT_USB の場合、検索時間である searchTime に関係無く、検索は直ぐに完了します。

戻り値

接続タイプが CLS_PORT_WiFi の場合、成功した時はプリンターの IP アドレスのリストが返されます。失敗した時は空のリストが返されます。

接続タイプが CLS_PORT_BLUETOOTH で iOS6 以上の場合、成功した時はプリンターの Bluetooth デバイスアドレスのリストが返されます。失敗した時は空のリストが返されます。iOS6 未満の場合、Bluetooth デバイスアドレスの情報に対応していませんので、空のリストが返されます。

接続タイプが CLS_PORT_USB の場合、成功した時はプリンターのシリアル番号のリストが返されます。失敗した時は空のリストが返されます。

使用例

```
var result = Int32(0)
let array = printer.searchLabelPrinter(CLS_PORT_WiFi, withSearchTime: 4,
    result: &result)!
```

2.2.11 setLog メソッド

形式

```
func setLog(mode: Int32, withPath path: String?, withMaxSize maxSize: Int32)
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
mode	[IN]	ログモード	0 : 記録なし 1 : アクセス履歴の記録 2 : エラーのみ記録
path	[IN]	格納フォルダ	アプリケーション共有フォルダ内のフォルダ
maxSize	[IN]	ログサイズ	0: サイズ制限なし 1~: 最大サイズ(MB)

説明

このメソッドは、ログ機能を設定するために使用します。ログ機能の詳細は「[3.2 ログ機能について](#)」を参照してください。

戻り値

ありません。

使用例

```
printer.setLog(1, withPath: "/", withMaxSize: 10)
```

2.2.12 HorizontalMagnification プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、水平方向のピクセルサイズ(ドット構成単位)を保持します。値は、1(dot)もしくは、2(dot)で指定します。

設定方法

```
func setHorizontalMagnification(horizontalMagnification: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

デフォルトを「1」とし、未設定時はデフォルトでコマンドを送信します。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getHorizontalMagnification() -> Int32
```

戻り値として、設定されている水平ピクセルサイズを返します。

デフォルトを「1」とし、未設定時はデフォルトが返ります。

使用例

```
var pixelSize: Int32! = 0

printer!.setHorizontalMagnification(2)
pixelSize = printer!.getHorizontalMagnification()
```

2.2.13 VerticalMagnification プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、垂直方向のピクセルサイズ(dot 構成単位)を保持します。値は、1(dot)から 3(dot)の間で指定します。

設定方法

```
func setVerticalMagnification(verticalMagnification: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

デフォルトを「1」とし、未設定時はデフォルトでコマンドを送信します。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getVerticalMagnification() -> Int32
```

戻り値として、設定されている垂直ピクセルサイズを返します。

デフォルトを「1」とし、未設定時はデフォルトが返ります。

使用例

```
var pixelSize: Int32! = 0
```

```
printer!.setVerticalMagnification(3)  
pixelSize = printer!.getVerticalMagnification()
```


2.2.14 FormatAttribute プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、印刷領域が重なっている箇所の展開方法を保持します。値は、以下の通りです。

0: XOR 展開指定となり、文字やバーコードの重なった部分が白抜きとなります。(初期値)

1: OR 展開指定となり、文字やバーコードの重ね書きを行います。

設定方法

```
func setFormatAttribute(formatAttribute: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時は、コマンドを送信しません。

プリンターのデフォルト値は「0」です。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getFormatAttribute() -> Int32
```

戻り値として、設定されている展開方法を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var attr: Int32! = 0

printer!.setFormatAttribute(1)
attr = printer!.getFormatAttribute()
```

印刷結果



2.2.15 ContinuousMediaLength プロパティ

形式

Int32

属性

Read/Write

説明

連続紙を使用した場合のラベル長の設定です。
ラベルフォーマットの長さは、このコマンドで設定した長さになります。
オートカッター使用時は、この設定の長さで、ラベルカットを行います。

インチ設定	0001 ~ 9999 (0.01 インチ~99.99 インチ)
ミリ設定	0001 ~ 9999 (0.1mm~999.9mm)

設定方法

```
func setContinuousMediaLength(continuousMediaLength: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。
未設定時は、コマンドを送信しません。
プリンターのデフォルト値は「0000」です。
成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getContinuousMediaLength() -> Int32
```

戻り値として、設定されている連続用紙長を返します。
未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var mediaLength: Int32! = 0

printer!.setContinuousMediaLength(2000)
mediaLength = printer!.getContinuousMediaLength()
```

2.2.16 MeasurementUnit プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、全ての全距離指定コマンドのパラメータの単位を保持します。値は以下の通りです。

値	意味
CLS_UNIT_MILLI (0)	ミリ(0.1mm 単位)
CLS_UNIT_INCH (1)	インチ(0.01 インチ単位)

設定方法

```
func setMeasurementUnit(measurementUnit: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時は、コマンドを送信しません。

プリンターのデフォルト値は「1」です。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getMeasurementUnit() -> Int32
```

戻り値として、設定されている距離指定パラメータの単位を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var unit: Int32! = 0
```

```
printer!.setMeasurementUnit(CLS_UNIT_MILLI)  
unit = printer!.getMeasurementUnit()
```

2.2.17 PrintSpeed プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、印字部分の速度を保持します。値は、1 文字のアルファベットか数字で設定します。(「[3.3. パラメータ](#)」の No21 を参照)

※指定可能範囲、プリンターの初期値は、機種により異なります。

使用機種の取扱い説明書をご参照ください。

設定方法

```
func setPrintSpeed(printSpeed: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getPrintSpeed() -> Int32
```

戻り値として、設定されている印刷速度を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var printSpeed: Int32! = 0

printer!.setPrintSpeed(CLS_SPEEDSETTING_X)
printSpeed = printer!.getPrintSpeed()
```

2.2.18 FeedSpeed プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、非印字部分の速度を保持します。値は、1 文字のアルファベットか数字で設定します。
(「[3.3.パラメータ](#)」の No21 を参照)

※指定可能範囲、プリンターの初期値は、機種により異なります。

使用機種の取扱い説明書をご参照ください。

設定方法

```
func setFeedSpeed(_ feedSpeed: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getFeedSpeed() -> Int32
```

戻り値として、設定されている非印刷部分の紙送り速度を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var feedSpeed: Int32! = 0

printer!.setFeedSpeed(LabelConst.CLS_SPEEDSETTING_W)
feedSpeed = printer!.getFeedSpeed()
```

2.2.19 SlewSpeed プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、紙送りのフィード速度を保持します。値は、1 文字のアルファベットか数字で設定します。
(「[3.3.パラメータ](#)」の No21 を参照)

※指定可能範囲、プリンターの初期値は、機種により異なります。

使用機種の取扱い説明書をご参照ください。

設定方法

```
func setSlewSpeed(slewSpeed: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getSlewSpeed() -> Int32
```

戻り値として、設定されている紙送り部分の紙送り速度を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var slewSpeed: Int32! = 0

printer!.setSlewSpeed(CLS_SPEEDSETTING_V)
slewSpeed = printer!.getSlewSpeed()
```

2.2.20 BackupSpeed プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、バックフィード時の速度を保持します。値は、1 文字のアルファベットか数字で設定します。

値	意味
CLS_SPEEDSETTING_A or CLS_SPEEDSETTING_B	1.0 インチ (25.4mm) / 秒
CLS_SPEEDSETTING_C or CLS_SPEEDSETTING_D	2.0 インチ (50.8mm) / 秒
CLS_SPEEDSETTING_E or CLS_SPEEDSETTING_F	3.0 インチ (76.2mm) / 秒
CLS_SPEEDSETTING_G or CLS_SPEEDSETTING_H	4.0 インチ (101.6mm) / 秒
CLS_SPEEDSETTING_I or CLS_SPEEDSETTING_J	5.0 インチ (127.0mm) / 秒
CLS_SPEEDSETTING_K or CLS_SPEEDSETTING_L	6.0 インチ (152.4mm) / 秒
CLS_SPEEDSETTING_M or CLS_SPEEDSETTING_N	7.0 インチ (177.8mm) / 秒
CLS_SPEEDSETTING_O	8.0 インチ (203.2mm) / 秒
CLS_SPEEDSETTING_1 ～ CLS_SPEEDSETTING_8	1.0 インチ (25.4mm) / 秒 ～ 8.0 インチ (203.2mm) / 秒

※指定可能範囲、プリンターの初期値は、機種により異なります。

使用機種の取扱い説明書をご参照ください。

設定方法

```
func setBackupSpeed(backupSpeed: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getBackupSpeed() -> Int32
```

戻り値として、設定されているバックフィード設定を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var backupSpeed: Int32! = 0

printer!.setBackupSpeed(CLS_SPEEDSETTING_O)
backupSpeed = printer!.getBackupSpeed()
```

2.2.21 PrintDarkness プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、印刷濃度の設定を保持します。設定範囲と初期値は以下の通りです。

設定範囲	0 ~ 30
初期値	10

設定方法

```
func setPrintDarkness(printDarkness: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getPrintDarkness() -> Int32
```

戻り値として、設定されている印字濃度を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var printDarkness: Int32! = 0

printer!.setPrintDarkness(30)
printDarkness = printer!.getPrintDarkness()
```


2.2.22 DoubleHeat プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、ダブルヒート設定を保持します。ダブルヒートを設定することで、同じ位置に 2 回熱をかけて印字します。そのため、印字速度は半分になりますが、より濃く印字することができます。

- 0: ダブルヒート OFF (初期値)
- 1: ダブルヒート ON

設定方法

```
func setDoubleHeat(doubleHeat: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getDoubleHeat() -> Int32
```

戻り値として、設定されているダブルヒート設定を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var doubleHeat: Int32! = 0

printer!.setDoubleHeat(1)
doubleHeat = printer!.getDoubleHeat()
```

2.2.23 VerticalOffset プロパティ

形式

Int32

属性

Read/Write

説明

印字内容全体の位置を調整する為に、紙の上下(行)方向の印字開始位置のオフセット値を設定します。

インチ設定	0000 ~ 9999 (0.00 インチ～99.99 インチ)
ミリ設定	0000 ~ 9999 (0.0mm～999.9mm)
初期値	0000

設定方法

```
func setVerticalOffset(verticalOffset: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getVerticalOffset() -> Int32
```

戻り値として、設定されている上下オフセットを返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var verticalOffset: Int32! = 0

printer!.setVerticalOffset(10)
verticalOffset = printer!.getVerticalOffset()
```

2.2.24 HorizontalOffset プロパティ

形式

Int32

属性

Read/Write

説明

印字内容全体の位置を調整する為、用紙の左右オフセット値(列方向の印字開始位置)を設定します。

インチ設定	0000 ~ 9999 (0.00 インチ～99.99 インチ)
ミリ設定	0000 ~ 9999 (0.0mm～999.9mm)
初期値	0000

設定方法

```
func setHorizontalOffset(horizontalOffset: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getHorizontalOffset() -> Int32
```

戻り値として、設定されている左右オフセットを返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var horizontalOffset: Int32! = 0

printer!.setHorizontalOffset(20)
horizontalOffset = printer!.getHorizontalOffset()
```

2.2.25 MediaHandling プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、印刷後の動作設定を保持します。設定可能な動作は、以下の通りです。

値	意味
CLS_MEDIAHANDLING_NONE (0)	印刷後の動作を OFF にします。
CLS_MEDIAHANDLING_TEAROFF (1)	排出動作のみ ON にします。
CLS_MEDIAHANDLING_DISPENSES (2)	剥離センサーと排出動作を ON にします。
CLS_MEDIAHANDLING_PAUSE (3)	排出動作を ON にし、排出後にポーズ状態にします。
CLS_MEDIAHANDLING_CUT (4)	オートカッターのみ ON にします。
CLS_MEDIAHANDLING_CUTANDPAUSE (5)	オートカッターを ON にし、カット後にポーズ状態にします。
CLS_MEDIAHANDLING_PEELOFF (6)	剥離センサーのみ ON にします。
CLS_MEDIAHANDLING_REWIND (7)	リワインダー(巻き取り)モードを ON にします。

設定方法

```
func setMediaHandling(mediaHandling: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getMediaHandling() -> Int32
```

戻り値として、設定されている印刷後動作を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var mediaHandling: Int32! = 0

printer!.setMediaHandling(CLS_MEDIAHANDLING_PAUSE)
mediaHandling = printer!.getMediaHandling()
```

2.2.26 StartOffset プロパティ

形式

Int32

属性

Read/Write

説明

基準点から印字ヘッドまでの距離を指定することができます。この値を変える事により、物理的な印字開始位置を変更することが出来ます。

インチ指定			ミリ指定		
初期値	最小値	最大値	初期値	最小値	最大値
0220	0120	0320	0559	0305	0813

※単位 0.01 インチ又は 0.1mm

設定方法

```
func setStartOffset(startOffset: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

インチ/ミリ指定の対応のチェックは行いません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getStartOffset() -> Int32
```

戻り値として、設定されている開始オフセットを返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var startOffset: Int32! = 0

printer!.setStartOffset(220)
startOffset = printer!.getStartOffset()
```

2.2.27 StopOffset プロパティ

形式

Int32

属性

Read/Write

説明

下記範囲の値にて、基準点(基準線)からカット位置又は、剥離位置までの距離を指定できます。

指定値が小さい場合、フィード量が小さいので印刷したラベルをカットしてしまいます。

指定値が適切な場合、必要量フィード後、紙間でカットします。

指定値が大きい場合、フィード量が大きいので、次のラベルをカットしてしまいます。

動作	インチ指定			ミリ指定		
	初期値	最小値	最大値	初期値	最小値	最大値
通常印刷	0000	0000	9999	0000	0000	9999
カッター	0100			0254		
剥離	0050			0127		
ティアオフ	0070			0178		

※単位 0.01 インチ又は 0.1mm

設定方法

```
func setStopOffset(stopOffset: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getStopOffset() -> Int32
```

戻り値として、設定されている終了オフセットを返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var stopOffset: Int32! = 0

printer!.setStopOffset(240)
stopOffset = printer!.getStopOffset()
```

2.2.28 LabelSensor プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、用紙センサーの設定を保持します。設定値は、以下の通りです。

値	意味
CLS_SELSENSOR_NONE (0)	なし。 なしの場合、連続用紙長設定の値(ContinuousMediaLength プロパティ)をチェックし、未設定、または 0000 の場合は引数エラーとします。
CLS_SELSENSOR_SEETHROUGH (1)	エッジセンサー。 ラベル紙の紙間、ダイカット紙、タグ紙のノッチ穴検出などに使用します。
CLS_SELSENSOR_REFLECT (2)	反射型用紙センサー。 ラベル裏面に印刷された黒線を検出して、ラベル位置を認識します。

設定方法

```
func setLabelSensor(labelSensor: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getLabelSensor() -> Int32
```

戻り値として、設定されている用紙センサーを返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
// センサー「なし」以外のとき
```

```
var labelSensor: Int32! = 0
```

```
printer!.setLabelSensor(CLS_SELSENSOR_SEETHROUGH);
```

```
labelSensor = printer!.getLabelSensor();
```

```
// センサー「なし」のとき
```

```
var labelSensor: Int32! = 0
```

```
printer!.setContinuousMediaLength(100)
```

```
printer!.setLabelSensor(CLS_SELSENSOR_NONE)
```

```
labelSensor = printer!.getLabelSensor()
```

2.2.29 PrintMethod プロパティ

形式

Int32

属性

Read/Write

説明

リボンを使用する熱転写モードと感熱紙を使用する感熱モードの印刷方法の指定を行います。

値	意味
CLS_PRTMETHOD_TT (0)	熱転写モード
CLS_PRTMETHOD_DT (1)	感熱モード

設定方法

```
func setPrintMethod(printMethod: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getPrintMethod() -> Int32
```

戻り値として、設定されている印字方法を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var printMethod: Int32! = 0

printer!.setPrintMethod(CLS_PRTMETHOD_DT)
printMethod = printer!.getPrintMethod()
```


2.2.30 SensorLocation プロパティ

形式

Int32

属性

Read/Write

説明

前方センサーと後方センサーの 2 種類の紙検出センサーを搭載している機種において、使用する紙検出センサーを切り替えます。設定した内容はバックアップメモリに記憶され電源を切っても設定は有効です。

値	意味
CLS_SENS_LOCATION_FRONT (0)	前方センサー
CLS_SENS_LOCATION_ADJUSTABLE (1)	後方センサー

設定方法

```
func setSensorLocation(sensorLocation: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

未設定時はコマンドを送信しません。

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

取得方法

```
func getSensorLocation() -> Int32
```

戻り値として、設定されているセンサー設定(前方/後方)を返します。

未設定時は、デフォルトとして CLS_PROPERTY_DEFAULT(999999) を返します。

使用例

```
var location: Int32! = 0
```

```
printer!.setSensorLocation(CLS_SENS_LOCATION_ADJUSTABLE)  
location = printer!.getSensorLocation()
```

2.2.31 CommandInterpreterInAction プロパティ

形式

Int32

属性

Read

説明

このプロパティは、プリンターがコマンド処理中であることを示します。

このプロパティを参照する際は、あらかじめ [printerCheck メソッド](#)により、プリンターからステータスを取得しておく必要があります。

設定方法

なし。

取得方法

```
func getCommandInterpreterInAction() -> Int32
```

戻り値として、以下の値を返します。

値	意味
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	ステータス未取得

使用例

[printerCheck メソッド](#)を参照

2.2.32 PaperError プロパティ

形式

Int32

属性

Read

説明

このプロパティは、プリンターのペーパーエラー状態を示します。

このプロパティを参照する際は、あらかじめ [printerCheck メソッド](#)により、プリンターからステータスを取得しておく必要があります。

設定方法

なし。

取得方法

```
func getPaperError() -> Int32
```

戻り値として、以下の値を返します。

値	意味
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	ステータス未取得

使用例

[printerCheck メソッド](#)を参照

2.2.33 RibbonEnd プロパティ

形式

Int32

属性

Read

説明

このプロパティは、プリンターのリボンエンド状態を示します。

このプロパティを参照する際は、あらかじめ [printerCheck メソッド](#)により、プリンターからステータスを取得しておく必要があります。

設定方法

なし。

取得方法

```
func getRibbonEnd() -> Int32
```

戻り値として、以下の値を返します。

値	意味
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	ステータス未取得

使用例

[printerCheck メソッド](#)を参照

2.2.34 BatchProcessing プロパティ

形式

Int32

属性

Read

説明

このプロパティは、プリンターがバッチ処理印字中であることを示します。

このプロパティを参照する際は、あらかじめ [printerCheck メソッド](#)により、プリンターからステータスを取得しておく必要があります。

設定方法

なし。

取得方法

```
func getBatchProcessing() -> Int32
```

戻り値として、以下の値を返します。

値	意味
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	ステータス未取得

使用例

[printerCheck メソッド](#)を参照

2.2.35 Printing プロパティ

形式

Int32

属性

Read

説明

このプロパティは、プリンターが印字中であることを示します。

このプロパティを参照する際は、あらかじめ [printerCheck メソッド](#)により、プリンターからステータスを取得しておく必要があります。

設定方法

なし。

取得方法

```
func getPrinting() -> Int32
```

戻り値として、以下の値を返します。

値	意味
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	ステータス未取得

使用例

[printerCheck メソッド](#)を参照

2.2.36 Pause プロパティ

形式

Int32

属性

Read

説明

このプロパティは、プリンターがポーズ中であることを示します。

このプロパティを参照する際は、あらかじめ [printerCheck メソッド](#)により、プリンターからステータスを取得しておく必要があります。

設定方法

なし。

取得方法

```
func getPause() -> Int32
```

戻り値として、以下の値を返します。

値	意味
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	ステータス未取得

使用例

[printerCheck メソッド](#)を参照

2.2.37 WaitForPeeling プロパティ

形式

Int32

属性

Read

説明

このプロパティは、プリンターが剥離待ち中であることを示します。

このプロパティを参照する際は、あらかじめ [printerCheck メソッド](#)により、プリンターからステータスを取得しておく必要があります。

設定方法

なし。

取得方法

```
func getWaitForPeeling() -> Int32
```

戻り値として、以下の値を返します。

値	意味
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	ステータス未取得

使用例

[printerCheck メソッド](#)を参照

2.3. LabelDesign クラス

2.3.1 コンストラクタ

形式

LabelDesign

パラメータ

ありません。

説明

クラスを生成し、初期化します。

使用例

```
var : design? = CSJLabelLibSwift.LabelDesign()
```

2.3.2 drawTextPtrFont メソッド

形式

- 1) func drawTextPtrFont(data: String!, locale: Int32, font: Int32, rotation: Int32, hexp: Int32, vexp: Int32, size: Int32, x: Int32, y: Int32) -> Int32
- 2) func drawTextPtrFont(data: String!, locale: Int32, font: Int32, rotation: Int32, hexp: Int32, vexp: Int32, size: Int32, x: Int32, y: Int32, encoding: Int32) -> Int32

パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	印字文字列	※文字コード変換できない文字は、空白に置き換える。
locale	[IN]	ロケール(定数)	CLS_LOCALE_JP: 日本モデル用ロケール CLS_LOCALE_OTHER: 海外モデル用ロケール CLS_LOCALE_CN: 中国モデル用ロケール CLS_LOCALE_KR: 韓国モデル用ロケール
font	[IN]	文字種(定数)	CLS_PRT_FNT_0: システムフォント 0 CLS_PRT_FNT_1: システムフォント 1 CLS_PRT_FNT_2: システムフォント 2 CLS_PRT_FNT_3: システムフォント 3 CLS_PRT_FNT_4: システムフォント 4 CLS_PRT_FNT_5: システムフォント 5 CLS_PRT_FNT_6: システムフォント 6 CLS_PRT_FNT_7: システムフォント 7 CLS_PRT_FNT_8: システムフォント 8 CLS_PRT_FNT_TRIUMVIRATE: スムースフォント CLS_PRT_FNT_TRIUMVIRATE_B: スムースフォント(Bold) CLS_PRT_FNT_KANJI: 漢字(横書き) CLS_PRT_FNT_KANJIT: 漢字(縦書き)
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
hexp	[IN]	水平拡大率	1～24
vexp	[IN]	垂直拡大率	1～24
size	[IN]	文字サイズ(定数)	CLS_PRT_FNT_SIZE_4: 文字種ポイント数(4pt) CLS_PRT_FNT_SIZE_5: 文字種ポイント数(5pt) CLS_PRT_FNT_SIZE_6: 文字種ポイント数(6pt) CLS_PRT_FNT_SIZE_8: 文字種ポイント数(8pt) CLS_PRT_FNT_SIZE_10: 文字種ポイント数(10pt) CLS_PRT_FNT_SIZE_12: 文字種ポイント数(12pt) CLS_PRT_FNT_SIZE_14: 文字種ポイント数(14pt) CLS_PRT_FNT_SIZE_18: 文字種ポイント数(18pt) CLS_PRT_FNT_SIZE_24: 文字種ポイント数(24pt) CLS_PRT_FNT_SIZE_30: 文字種ポイント数(30pt) CLS_PRT_FNT_SIZE_36: 文字種ポイント数(36pt) CLS_PRT_FNT_SIZE_48: 文字種ポイント数(48pt) CLS_PRT_FNT_KANJI_SIZE_16: 漢字文字種(16ドット) CLS_PRT_FNT_KANJI_SIZE_24: 漢字文字種(24ドット) CLS_PRT_FNT_KANJI_SIZE_32: 漢字文字種(32ドット) ※ CLS_PRT_FNT_KANJI_SIZE_48: 漢字文字種(48ドット) ※ ※中国モデル用ロケールでは、非サポート。

x	[IN]	印字位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	
encoding	[IN]	印字文字列の文字コード (定数)	「 3.3.パラメータ 」の No9 を参照

説明

プリンターのシステムフォントを用いて、回転・縦横拡大率・フォント種・印字位置等・指定条件で、入力された内容の文字を描画します。

文字サイズは、文字種ごとに以下のパターンを指定可能です。

漢字: 16,24,32,48 ※ドット単位

スムースフォント: 4,5,6,8,10,12,14,18,24,30,36,48

上記以外: 文字種ごとにサイズ固定

文字コードが省略された場合は、自動的にロケールに合わせた文字コードが設定されます。文字コードに UTF-8 (CLS_ENC_CDPG_UTF_8) が指定できるモデルは CL-S700III のみです。(2024 年 4 月現在)

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
// Locale:OTHER
design.drawTextPtrFont("drawTextPtrFont",
    locale: CLS_LOCALE_OTHER, font: CLS_PRT_FNT_TRIUMVIRATE,
    rotation: CLS_RT_NORMAL, hexp: 1, vexp: 1,
    size: CLS_PRT_FNT_SIZE_10, x: 100, y: 100)

// Locale:JP
design.drawTextPtrFont("テキスト印刷(プリンターフォント)",
    locale: CLS_LOCALE_JP, font: CLS_PRT_FNT_KANJI,
    rotation: CLS_RT_NORMAL, hexp: 1, vexp: 1,
    size: CLS_PRT_FNT_KANJI_SIZE_16, x: 100, y: 300)

// Locale:CN
design.drawTextPtrFont("测试打印", locale: CLS_LOCALE_CN,
    font: CLS_PRT_FNT_KANJI, rotation: CLS_RT_NORMAL,
    hexp: 1, vexp: 1, size: CLS_PRT_FNT_KANJI_SIZE_16,
    x: 100, y: 300);

// Locale:KR
design.drawTextPtrFont("테스트 인쇄", locale: CLS_LOCALE_KR,
    font: CLS_PRT_FNT_KANJI, rotation: CLS_RT_NORMAL,
    hexp: 1, vexp: 1, size: CLS_PRT_FNT_KANJI_SIZE_16,
    x: 100, y: 300);
```

2.3.3 drawTextDLFont メソッド

形式

```
func drawTextDLFont(data: String!, encoding: Int32, fontID: String!, rotation: Int32, hexp: Int32,
    vexp: Int32, point: Int32, x: Int32, y: Int32) -> Int32
```

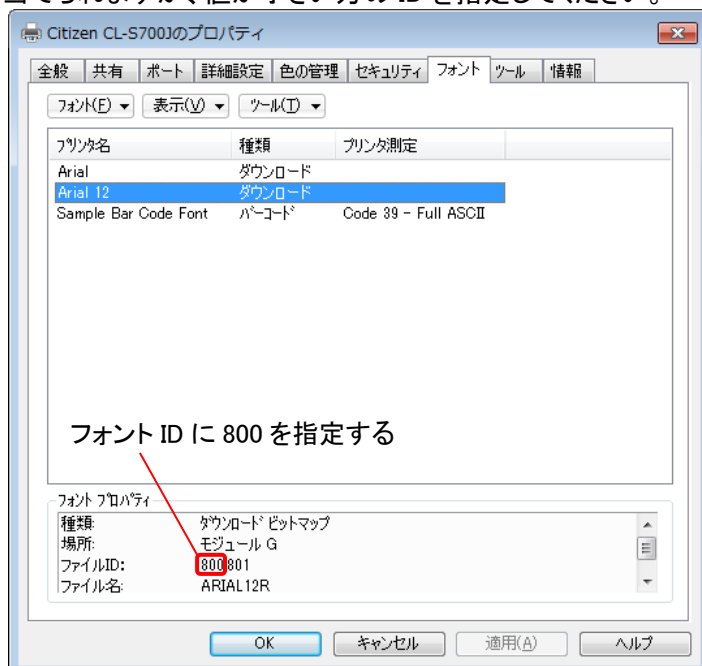
パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	印字文字列	※文字コード変換できない文字は、空白に置き換える。
encoding	[IN]	印字文字列の文字コード(定数)	「 3.3.パラメータ 」の No9 を参照
fontID	[IN]	プリンターにダウンロードしたフォントの ID	<ul style="list-style-type: none"> ・fontID が全て数字の場合 ビットマップダウンロードフォント ・fontID に数字以外が含まれる場合 TrueType ダウンロードフォント
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
hexp	[IN]	水平拡大率	1～24
vexp	[IN]	垂直拡大率	1～24
point	[IN]	文字サイズ	001～999
x	[IN]	印字位置(X 座標)	0000～9999
y	[IN]	印字位置(Y 座標)	※x,y は、左下を基準(0,0)として位置とする。

説明

プリンターにダウンロードしたフォントを使用して、回転・縦横拡大率・印字位置・文字サイズ等・指定条件で、入力された内容の文字を描画します。

ビットマップダウンロードフォントにつきましては、フォントをプリンターへダウンロードした際に 2 つの ID が割り当てられますが、値が小さい方の ID を指定してください。



文字コードに UTF-8 (CLS_ENC_CDPG_UTF_8) が指定できるモデルは CL-S700III のみです。(2022 年 10 月現在)

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
// TrueTypeダウンロードフォント
design.drawTextDLFont("TrueType",
    encoding: CLS_ENC_CDPG_IBM850, fontID: "S50",
    rotation: CLS_RT_NORMAL, hexp: 1, vexp: 1, point: 12,
    x: 100, y: 100)

// ビットマップダウンロードフォント
design.drawTextDLFont("Bitmap",
    encoding: CLS_ENC_CDPG_IBM850, fontID: "800",
    rotation: CLS_RT_NORMAL, hexp: 1, vexp: 1, point: 12,
    x: 100, y: 200)
```

2.3.4 drawTextLocalFont メソッド

形式

- 1) func drawTextLocalFont(data: String!, fontName: String!, rotation: Int32, hRatio: Int32, vRatio: Int32, point: Int32, style: Int32, x: Int32, y: Int32) -> Int32
- 2) func drawTextLocalFont(data: String!, fontName: String!, rotation: Int32, hRatio: Int32, vRatio: Int32, point: Int32, style: Int32, x: Int32, y: Int32, resolution: Int32) -> Int32
- 3) func drawTextLocalFont(data: String!, fontName: String!, rotation: Int32, hRatio: Int32, vRatio: Int32, point: Int32, style: Int32, x: Int32, y: Int32, resolution: Int32, measurementUnit: Int32) -> Int32

パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	印字文字列	
fontName	[IN]	印字文字列のフォント名	サポートされるフォントは、OS の実装によって異なります。
rotation	[IN]	回転方向	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
hRatio	[IN]	水平拡大率	1～1000(%単位)
vRatio	[IN]	垂直拡大率	1～1000(%単位)
point	[IN]	文字サイズ	
style	[IN]	文字スタイル(定数)	CLS_FNT_DEFAULT: 無し CLS_FNT_BOLD: 太字 CLS_FNT_REVERSE: 反転 CLS_FNT_UNDERLINE: アンダーライン CLS_FNT_ITALIC: イタリック CLS_FNT_STRIKEOUT: 取り消し線 ※複数指定時は「 」で連結する。
x	[IN]	印字位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	
resolution	[IN]	解像度(dpi)	CLS_PRT_RES_203(203dpi) CLS_PRT_RES_300(300dpi) ※省略時は、CLS_PRT_RES_203 となる。
measurementUnit	[IN]	プリンターの単位選択設定	CLS_UNIT_MILLI CLS_UNIT_INCH ※省略時は、CLS_UNIT_INCH となる。

説明

文字コード・回転・縦横拡大率・文字スタイル・文字サイズ・印字位置・解像度等・指定条件で、入力された内容の文字を描画します。

本メソッドは内部的に、入力された内容を端末で画像化し、画像化したデータをプリンターへ登録してから印刷を行っていますが、このとき、プリンターのメモリ容量の問題等で、印字画像をプリンターへ登録可能であるか・登録されたかのチェックは行いません。

指定した文字サイズで印刷を行うには、解像度をプリンターに合わせて指定してください。

イメージの回転指定をする場合は、基準位置補正のため解像度と単位選択設定をプリンターに合わせて指定する必要があります。

文字スタイルは、太字、反転、アンダーライン、イタリック、取り消し線を組み合わせて指定可能です。組み合

わせる場合は論理和を指定してください。なお、イタリック指定は iOS13 以降では無視されます。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
// 解像度指定 省略(203dpi)、単位指定 省略(inch)
design.drawTextLocalFont("drawTextLocalFont",
    fontName: "Arial Hebrew", rotation: CLS_RT_NORMAL,
    hRatio: 100, vRatio: 100, point: 10,
    style: CLS_FNT_BOLD | CLS_FNT_ITALIC, x: 10, y: 0)

// 解像度指定 300dpi、単位指定 省略(inch)
design.drawTextLocalFont("drawTextLocalFont",
    fontName: "Arial Hebrew", rotation: CLS_RT_NORMAL,
    hRatio: 100, vRatio: 100, point: 10,
    style: CLS_FNT_DEFAULT, x: 10, y: 50,
    resolution: CLS_PRT_RES_300)

// 解像度指定 300dpi、単位指定 mm
design.drawTextLocalFont("drawTextLocalFont",
    fontName: "Arial Hebrew", rotation: CLS_RT_NORMAL,
    hRatio: 100, vRatio: 100, point: 10,
    style: CLS_FNT_DEFAULT, x: 10, y: 100,
    resolution: CLS_PRT_RES_300, measurementUnit: CLS_UNIT_MILLI)
```

2.3.5 drawNVBitmap メソッド

形式

```
func drawNVBitmap(name: String!, hexp: Int32, vexp: Int32, x: Int32, y: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
name	[IN]	画像ファイル名	以下の条件を除いた ASCII コードが使用可能 ・先頭がアンダースコアは不可 ・以下の文字は使用不可(禁則文字) ¥ / : * ? " < >
hexp	[IN]	水平拡大率	1～24
vexp	[IN]	垂直拡大率	1～24
x	[IN]	印字位置(X 座標)	0000～9999
y	[IN]	印字位置(Y 座標)	※x,y は、左下を基準(0,0)として位置とする。

説明

縦横拡大率・印字位置等・指定条件で、プリンターに登録されている画像データを描画します。
指定された画像ファイルが、プリンターに登録されているかのチェックは行いません。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

[storeNVBitmap メソッド](#)を参照

2.3.6 drawBitmap メソッド

形式

- 1) func drawBitmap(filePath: String!, rotation: Int32, width: Int32, height: Int32, x: Int32, y: Int32) -> Int32
- 2) func drawBitmap(filePath: String!, rotation: Int32, width: Int32, height: Int32, x: Int32, y: Int32, resolution: Int32, measurementUnit: Int32) -> Int32

パラメータ

値	[IN/OUT]	意味	設定可能範囲
filePath	[IN]	対象のファイルパス	
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
width	[IN]	横サイズ(ピクセル単位)	
height	[IN]	縦サイズ(ピクセル単位)	
x	[IN]	印字位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	
resolution	[IN]	解像度(dpi)	CLS_PRT_RES_203(203dpi) CLS_PRT_RES_300(300dpi) ※省略時は、CLS_PRT_RES_203 となる。
measurementUnit	[IN]	プリンターの単位選択設定	CLS_UNIT_MILLI CLS_UNIT_INCH ※省略時は、CLS_UNIT_INCH となる。

説明

回転・縦横拡大率・印字位置等の指定条件で、指定された画像ファイルを描画します。

本メソッドは内部的に、指定された画像ファイルをプリンターへ登録してから描画を行っていますが、このとき、プリンターのメモリ容量の問題等で、画像をプリンターへ登録可能であるか・登録されたかのチェックは行いません。

イメージの回転指定をする場合は、基準位置補正のため解像度と単位選択設定をプリンターに合わせて指定する必要があります。

描画可能なビットマップ形式は、BMP/GIF/EXIF/JPG/PNG/TIFF です。

描画される画像サイズは、指定された縦横サイズに合わせて、アスペクト比を維持した状態でリサイズします。

例: 以下の場合、リサイズ後の画像サイズは「200x50」ピクセルとなる。

画像サイズ: 400x100 ピクセル

横サイズ: 200

縦サイズ: 200

指定のサイズが 0 の場合、対象の画像サイズを基準とする。

例: 以下の場合、リサイズ後の画像サイズは「800x200」ピクセルとなる。

画像サイズ: 400x100 ピクセル

横サイズ: 0

縦サイズ: 200

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
// Resource内ファイル取得
let FileName = NSBundle.mainBundle().pathForResource("sample_1.jpg",
    ofType: nil)

// 解像度指定 省略(203dpi)、単位指定 省略(inch)
design.drawBitmap(FileName, rotation: CLS_RT_NORMAL,
    width: 0, height: 0, x: 10, y: 10)

// 解像度指定 300dpi、単位指定 mm
design.drawBitmap("Sample.bmp", rotation: CLS_RT_NORMAL,
    width: 0, height: 0, x: 10, y: 10,
    resolution: CLS_PRT_RES_300, measurementUnit: CLS_UNIT_MILLI)
```

2.3.7 drawBarCode メソッド

形式

```
func drawBarCode(data: String!, symbology: Int32, rotation: Int32, thick: Int32, narrow: Int32,
    height: Int32, x: Int32, y: Int32, showText: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	バーコードデータ文字列	ASCII コード文字列
symbology	[IN]	バーコードの種類(定数)	CLS_BCS_CODE39: Code 3 of 9 CLS_BCS_UPCA: UPC-A CLS_BCS_UPCE: UPC-E CLS_BCS_INTERLEAVED25: Interleaved 2 of 5 CLS_BCS_CODE128: Code 128 CLS_BCS_EAN13: EAN-13(JAN-13) CLS_BCS_EAN8: EAN-8(JAN-8) CLS_BCS_HIBC: HIBC CLS_BCS_CODABAR: CODABAR(NW-7) CLS_BCS_INT25: Int 2 of 5 CLS_BCS_PLESSEY: Plessey CLS_BCS_CASECODE: CASE CODE CLS_BCS_UPC2DIG: UPC 2DIG ADD CLS_BCS_UPC5DIG: UPC 5DIG ADD CLS_BCS_CODE93: Code93 CLS_BCS_ITF14: ITF-14 CLS_BCS_ZIP: ZIP CLS_BCS_ITF16: IFT-16 CLS_BCS_UCCEAN128: UCC/EAN-128 CLS_BCS_INDUSTRIAL25: Industrial 2 of 5 CLS_BCS_UCCEAN128KMART: UCC/EAN-128(for K-MART) CLS_BCS_COOP25: COOP 2 of 5 CLS_BCS_UCCEAN128RANDOMWEIGHT: UCC/EAN-128 Random Weight CLS_BCS_TELEPEN: Telepen
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
thick	[IN]	太バーの幅	1～24
narrow	[IN]	細バーの幅	1～24
height	[IN]	バーコードデータの高さ	001～999
x	[IN]	印字位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	
showText	[IN]	バーコード文字の表示有無(定数)	CLS_BCS_TEXT_HIDE: 非表示 CLS_BCS_TEXT_SHOW: 表示

説明

このメソッドは、一次元のバーコードを描画します。また、バーコードに対して、バーコード種類・太/細バーの幅・バーコードデータの高さ・印字位置・バーコード文字の表示有無等の条件を指定することができます。

※バーコードの種類によって、thick/narrow 比率や、有効なバーコードデータ文字列が異なります。詳細は、ラベルプリンターのコマンドリファレンス「バーコードフィールドの定義」、「各バーコードの説明」の解説をご確認ください。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
// Code3of9
var code39 = "ABC123456789"
design.drawBarcode(code39,
    symbology: CLS_BCS_CODE39, rotation: CLS_RT_NORMAL,
    thick: 6, narrow: 2, height: 50, x: 10, y: 10,
    showText: CLS_BCS_TEXT_SHOW)

// UPC-A
var upca = "01234567890"
design.drawBarcode(upca,
    symbology: CLS_BCS_UPCA, rotation: CLS_RT_NORMAL,
    thick: 2, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// UPC-E
var upce = "123456"
design.drawBarcode(upce,
    symbology: CLS_BCS_UPCE, rotation: CLS_RT_NORMAL,
    thick: 2, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// Interleaved2of5
var interleaved25 = "1234567890"
design.drawBarcode(interleaved25,
    symbology: CLS_BCS_INTERLEAVED25, rotation: CLS_RT_NORMAL,
    thick: 5, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// Code128
var code128 = "1234567890"
design.drawBarcode(code128,
    symbology: CLS_BCS_CODE128, rotation: CLS_RT_NORMAL,
    thick: 2, narrow: 4, height: 50, x: 10, y: 10,
    showText: CLS_BCS_TEXT_SHOW)

// EAN-13
var ean13 = "123456789012"
design.drawBarcode(ean13,
    symbology: CLS_BCS_EAN13, rotation: CLS_RT_NORMAL,
    thick: 3, narrow: 3, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)
```

```
// EAN-8
var ean8 = "1234567"
design.drawBarcode(ean8,
    symbology: CLS_BCS_EAN8, rotation: CLS_RT_NORMAL,
    thick: 4, narrow: 4, height: 50, x: 10, y: 10,
    showText: CLS_BCS_TEXT_SHOW)

// HIBC
var hIBC = "1234567890"
design.drawBarcode(hIBC,
    symbology: CLS_BCS_HIBC, rotation: CLS_RT_NORMAL,
    thick: 6, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// CODABAR
var codabar = "a1234567890b"
design.drawBarcode(codabar,
    symbology: CLS_BCS_CODABAR, rotation: CLS_RT_NORMAL,
    thick: 6, narrow: 2, height: 40, x: 10, y: 10,
    showText: CLS_BCS_TEXT_SHOW)

// Interleaved2of5 W/BARS
var int25 = "1234567890"
design.drawBarcode(int25,
    symbology: CLS_BCS_INT25, rotation: CLS_RT_NORMAL,
    thick: 5, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// PLESSEY
var plessey = "1234567890"
design.drawBarcode(plessey,
    symbology: CLS_BCS_PLESSEY, rotation: CLS_RT_NORMAL,
    thick: 4, narrow: 2, height: 50, x: 10, y: 10,
    showText: CLS_BCS_TEXT_SHOW)

// CASE CODE
var casecode = "1234567890123"
design.drawBarcode(casecode,
    symbology: CLS_BCS_CASECODE, rotation: CLS_RT_NORMAL,
    thick: 5, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// UPC 2DIG ADD
var upca = "01234567890"
var upc2dig = "12"

design.drawBarcode(upca,
    symbology: CLS_BCS_UPCA, rotation: CLS_RT_NORMAL,
    thick: 2, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

design.drawBarcode(upc2dig,
    symbology: CLS_BCS_UPC2DIG, rotation: CLS_RT_NORMAL,
```

```
        thick: 4, narrow: 2, height: 50, x: 130, y: 100,
        showText: CLS_BCS_TEXT_SHOW)

// UPC 5DIG ADD
var upca = "01234567890"
var upc5dig = "12345"

design.drawBarcode(upca,
    symbology: CLS_BCS_UPCA, rotation: CLS_RT_NORMAL,
    thick: 2, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

design.drawBarcode(upc5dig,
    symbology: CLS_BCS_UPC5DIG, rotation: CLS_RT_NORMAL,
    thick: 4, narrow: 2, height: 50, x: 130, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// Code93
var code93 = "1234ABCD"
design.drawBarcode(code93,
    symbology: CLS_BCS_CODE93, rotation: CLS_RT_NORMAL,
    thick: 6, narrow: 6, height: 50, x: 10, y: 10,
    showText: CLS_BCS_TEXT_SHOW)

// ITF-14
var itf14 = "1234567890123"
design.drawBarcode(itf14,
    symbology: CLS_BCS_ITF14, rotation: CLS_RT_NORMAL,
    thick: 5, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// ZIP
var zip = "123456789"
design.drawBarcode(zip,
    symbology: CLS_BCS_ZIP, rotation: CLS_RT_NORMAL,
    thick: 1, narrow: 1, height: 10, x: 10, y: 10,
    showText: CLS_BCS_TEXT_SHOW)

// ITF-16
var itf16 = "123456789012345"
design.drawBarcode(itf16,
    symbology: CLS_BCS_ITF16, rotation: CLS_RT_NORMAL,
    thick: 5, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// UCC/EAN-128
var uccean128 = "1234567890123456789"
design.drawBarcode(uccean128,
    symbology: CLS_BCS_UCCEAN128, rotation: CLS_RT_NORMAL,
    thick: 4, narrow: 4, height: 50, x: 10, y: 10,
    showText: CLS_BCS_TEXT_SHOW)
```

```
// Industrial2of5
var industrial25 = "1234567890"
design.drawBarcode(industrial25,
    symbology: CLS_BCS_INDUSTRIAL25, rotation: CLS_RT_NORMAL,
    thick: 5, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// UCC/EAN-128 (for K-MART)
var uccean128kmart = "123456789012345678"
design.drawBarcode(uccean128kmart,
    symbology: CLS_BCS_UCCEAN128KMART, rotation: CLS_RT_NORMAL,
    thick: 3, narrow: 3, height: 50, x: 10, y: 10,
    showText: CLS_BCS_TEXT_SHOW)

// COOP 2 of 5
String coop25 = "0123456789"
design.drawBarcode(coop25,
    symbology: CLS_BCS_COOP25, rotation: CLS_RT_NORMAL,
    thick: 10, narrow: 4, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)

// UCC/EAN-128 Random Weight
var uccean128randomweight = "12345678901234567890123456789099999"
design.drawBarcode(uccean128randomweight,
    symbology: CLS_BCS_UCCEAN128RANDOMWEIGHT, rotation: CLS_RT_NORMAL,
    thick: 2, narrow: 2, height: 50, x: 10, y: 10,
    showText: CLS_BCS_TEXT_SHOW)

// Telepen
var telepen = "1234567890"
design.drawBarcode(telepen,
    symbology: CLS_BCS_TELEPEN, rotation: CLS_RT_NORMAL,
    thick: 2, narrow: 2, height: 50, x: 10, y: 100,
    showText: CLS_BCS_TEXT_SHOW)
```

2.3.8 drawMaxiCode メソッド

形式

```
func drawMaxiCode(data: [Any]!, rotation: Int32, x: Int32, y: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	バーコードデータ文字列	ASCII コード文字列
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
x	[IN]	印字位置(X 座標)	0000~9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	

説明

このメソッドは、UPS MaxiCode 形式のバーコードを描画します。また、バーコードに対して、回転・印字位置の条件を指定することができます。

バーコードデータ文字列には、以下 5 つの要素を、①から順にセットしてください。

- ① 5 桁の Zip コード
- ② 4 桁の+4Zip コード
- ③ 3 桁の国別コード
- ④ 3 桁の class of service code
- ⑤ 84 桁以内のデータ文字列

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
let data: [String] = ["90501", "6282", "840", "001", "1200004951"]
design.drawMaxiCode(data, rotation: CLS_RT_NORMAL, x: 100, y: 0)
```


2.3.9 drawPDF417 メソッド

形式

```
func drawPDF417(data: String!, encoding: Int32, rotation: Int32, exp: Int32, ecLevel: Int32,
                x: Int32, y: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	バーコードデータ文字列	※指定の文字コードで使用可能な文字
encoding	[IN]	バーコードデータの文字コード(定数)	「 3.3.パラメータ 」の No9 を参照
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
exp	[IN]	拡大率	1～5
ecLevel	[IN]	エラー修正レベル(定数)	CLS_PDF417_EC_LEVEL_0: レベル 0 CLS_PDF417_EC_LEVEL_1: レベル 1 CLS_PDF417_EC_LEVEL_2: レベル 2 CLS_PDF417_EC_LEVEL_3: レベル 3 CLS_PDF417_EC_LEVEL_4: レベル 4 CLS_PDF417_EC_LEVEL_5: レベル 5 CLS_PDF417_EC_LEVEL_6: レベル 6 CLS_PDF417_EC_LEVEL_7: レベル 7 CLS_PDF417_EC_LEVEL_8: レベル 8
x	[IN]	印字位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	

説明

このメソッドは、PDF-417 形式のバーコードを描画します。また、バーコードに対して、エンコード・回転・拡大率・エラー修正レベル・印字位置の条件を指定することができます。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
design.drawPDF417("0123456789",
                encoding: CLS_ENC_CDPG_US_ASCII, rotation: CLS_RT_NORMAL,
                exp: 3, ecLevel: CLS_PDF417_EC_LEVEL_0, x: 0, y: 0)
```

2.3.10 drawDataMatrix メソッド

形式

```
func drawDataMatrix(data: String!, encoding: Int32, rotation: Int32, exp: Int32, ecLevel: Int32,
    x: Int32, y: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	バーコードデータ文字列	・数字のみの場合、3116 文字以内 ・英数字混在の場合、2335 文字以内 ※指定の文字コードで使用可能な文字
encoding	[IN]	バーコードデータの文字コード(定数)	「 3.3.パラメータ 」の No9 を参照
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
exp	[IN]	拡大率	1～15
ecLevel	[IN]	エラー修正レベル(定数)	CLS_DATAMATRIX_EC_LEVEL_200: 200
x	[IN]	印字位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	

説明

このメソッドは、DataMatrix 形式のバーコードを描画します。また、バーコードに対して、エンコード・回転・拡大率・印字位置の条件を指定することができます。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
design.drawDataMatrix("0123456789",
    encoding: CLS_ENC_CDPG_US_ASCII, rotation: CLS_RT_NORMAL,
    exp:15, ecLevel: CLS_DATAMATRIX_EC_LEVEL_200, x: 0, y: 0)
```

2.3.11 drawQRCode メソッド

形式

```
func drawQRCode(data: String!, encoding: Int32, rotation: Int32, exp: Int32, ecLevel: Int32,
                x: Int32, y: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	バーコードデータ文字列	※指定の文字コードで使用可能な文字
encoding	[IN]	バーコードデータの文字コード(定数)	「 3.3.パラメータ 」の No9 を参照
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
exp	[IN]	拡大率	1～15
ecLevel	[IN]	エラー修正レベル(定数)	CLS_QRCODE_EC_LEVEL_L: レベル L(7%) CLS_QRCODE_EC_LEVEL_M: レベル M(15%) CLS_QRCODE_EC_LEVEL_Q: レベル Q(25%) CLS_QRCODE_EC_LEVEL_H: レベル H(30%)
x	[IN]	印字位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	

説明

このメソッドは、QRコードを描画します。また、QRコードに対して、エンコード・回転・拡大率・エラー修正レベル・印字位置の条件を指定することができます。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
design.drawQRCode("アイエオ12345",
                encoding: CLS_ENC_CDPG_SHIFT_JIS, rotation: CLS_RT_NORMAL,
                exp:10, ecLevel: CLS_QRCODE_EC_LEVEL_H, x: 100, y: 100)
```

2.3.12 drawAztec メソッド

形式

```
func drawAztec(data: String!, encoding: Int32, rotation: Int32, exp: Int32, ecLevel: Int32,
               x: Int32, y: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	バーコードデータ文字列	※指定の文字コードで使用可能な文字
encoding	[IN]	バーコードデータの文字コード(定数)	「 3.3.パラメータ 」の No9 を参照
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
exp	[IN]	拡大率	1～15
ecLevel	[IN]	エラー修正レベル(定数)	CLS_AXTEC_EC_LEVEL_000: レベル 0(誤り訂正率 23%)
x	[IN]	印字位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	

説明

このメソッドは、Aztec 形式のバーコードを描画します。また、バーコードに対して、エンコード・回転・拡大率・印字位置の条件を指定することができます。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
design.drawAztec("0123456789",
                encoding: CLS_ENC_CDPG_US_ASCII, rotation: CLS_RT_NORMAL,
                exp:10, ecLevel: CLS_AXTEC_EC_LEVEL_000, x: 0, y: 0)
```

2.3.13 drawGS1DataBar メソッド

形式

```
func drawGS1DataBar(data: [Any]!, type _type: Int32, rotation: Int32, exp: Int32,
    x: Int32, y: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	バーコードデータ文字列	ASCII コード文字列
type	[IN]	バーコードタイプ(定数)	CLS_GS1_DATABAR_OMNI_DIRECTIONAL: Omni-directional CLS_GS1_DATABAR_COMPOSITE: Composite CLS_GS1_DATABAR_TRUNCATION: Truncation CLS_GS1_DATABAR_STACKED: Stacked CLS_GS1_DATABAR_STACKED_OMNI_DIRECTIONAL: Stacked Omni-directional CLS_GS1_DATABAR_LIMITED: Limited CLS_GS1_DATABAR_EXPANDED: Expanded
rotation	[IN]	回転方向(定数)	CLS_RT_NORMAL: 回転なし CLS_RT_RIGHT90: 右 90 度回転 CLS_RT_ROTATE180: 右 180 度回転 CLS_RT_LEFT90: 左 90 度回転
exp	[IN]	拡大率	1～15
x	[IN]	印字位置(X 座標)	0000～9999
y	[IN]	印字位置(Y 座標)	※x,y は、左下を基準(0,0)として位置とする。

説明

このメソッドは、GS1 DataBar 形式のバーコードを描画します。また、バーコードに対して、バーコードタイプ・回転・拡大率・印字位置等の条件を指定することができます。

※バーコードの種類によって、指定可能なバーコードデータ文字列などが異なります。詳細は、ラベルプリンターのコマンドリファレンス「バーコード W1k(国内モデル、海外モデル共通): GS1 DataBar (RSS)」の解説をご確認ください。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
// GS1 DataBar Omni-Directional
let omnidirectional: [String] = ["1234567890123"]
design.drawGS1DataBar(omnidirectional,
    type: CLS_GS1_DATABAR_OMNI_DIRECTIONAL,
    rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 10)

// GS1 DataBar Composite
let composit: [String] = ["1234567890123", "1234567890-07/07/07"]
design.drawGS1DataBar(composit,
    type: CLS_GS1_DATABAR_COMPOSITE,
    rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 110)
```

```
// GS1 DataBar Truncation
let truncation: [String] = ["1234567890123"]
design.drawGS1DataBar(truncation,
    type: CLS_GS1_DATABAR_TRUNCATION,
    rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 210)

// GS1 DataBar Stacked
let stacked: [String] = ["1234567890123"]
design.drawGS1DataBar(stacked,
    type: CLS_GS1_DATABAR_STACKED,
    rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 310)

// GS1 DataBar Stacked-Omni-Directional
let stackedOD: [String] = ["1234567890123"]
design.drawGS1DataBar(stackedOD,
    type: CLS_GS1_DATABAR_STACKED_OMNI_DIRECTIONAL,
    rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 410)

// GS1 DataBar Limited
let limited: [String] = ["1234567890123"]
design.drawGS1DataBar(limited,
    type: CLS_GS1_DATABAR_LIMITED,
    rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 610)

// GS1 DataBar Expanded
let expanded: [String] = ["041234567890123"]
design.drawGS1DataBar(expanded,
    type: CLS_GS1_DATABAR_EXPANDED,
    rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 710)
```

2.3.14 drawLine メソッド

形式

```
func drawLine(x1: Int32, y1: Int32, x2: Int32, y2: Int32, thickness: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
x1	[IN]	印字開始位置(X 座標、中心点)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y1	[IN]	印字開始位置(Y 座標、中心点)	
x2	[IN]	印字終了位置(X 座標、中心点)	
y2	[IN]	印字終了位置(Y 座標、中心点)	
thickness	[IN]	線幅(中心点を基準)	0000～9999

説明

設定された幅の線を描画します。

印字開始・終了位置は、中心線の座標を示します。線幅を考慮した線の座標が、0～9999 の範囲外になる場合、CLS_E_ILLEGAL(1101)エラーになります。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
// 印字領域が重なる部分を上書きに指定
printer!.setFormatAttribute(1)
```

```
// 縦線
design.drawLine (20, y1: 30, x2: 20, y2: 300, thickness: 10)
```

```
// 横線
design.drawLine (16, y1: 34, x2: 200, y2: 34, thickness: 10)
```

2.3.15 drawRect メソッド

形式

```
func drawRect(x: Int32, y: Int32, width: Int32, height: Int32, thickness: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
x	[IN]	印字開始位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字開始位置(Y 座標)	
width	[IN]	水平幅	0000～9999
height	[IN]	垂直幅	
thickness	[IN]	線幅	0000～9999

説明

設定された寸法の矩形を描画します。

印字開始位置は、線の外周の左下を示します。線幅を太くした際は、印字開始位置から内側に線が太くなります。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
design.drawRect(20, y: 30, width: 180, height: 280, thickness: 10)
```


2.3.16 fillRect メソッド

形式

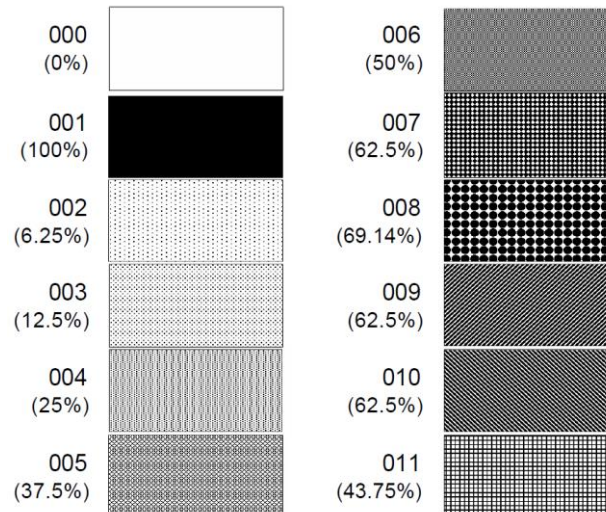
```
func fillRect(x: Int32, y: Int32, width: Int32, height: Int32, pattern: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
x	[IN]	印字開始位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字開始位置(Y 座標)	
width	[IN]	水平幅	0000～9999
height	[IN]	垂直幅	
pattern	[IN]	網掛けパターン(定数)	CLS_SHADED_PTN_0: 網掛けパターン 000 CLS_SHADED_PTN_1: 網掛けパターン 001 CLS_SHADED_PTN_2: 網掛けパターン 002 CLS_SHADED_PTN_3: 網掛けパターン 003 CLS_SHADED_PTN_4: 網掛けパターン 004 CLS_SHADED_PTN_5: 網掛けパターン 005 CLS_SHADED_PTN_6: 網掛けパターン 006 CLS_SHADED_PTN_7: 網掛けパターン 007 CLS_SHADED_PTN_8: 網掛けパターン 008 CLS_SHADED_PTN_9: 網掛けパターン 009 CLS_SHADED_PTN_10: 網掛けパターン 010 CLS_SHADED_PTN_11: 網掛けパターン 011

説明

設定された寸法の矩形を描画し、設定されたパターンで内部を網掛けします。



戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
design.fillRect(20, y: 30, width: 180, height: 280,  
               pattern: CLS_SHADED_PTN_10)
```

2.3.17 drawCircle メソッド

形式

```
func drawCircle(x: Int32, y: Int32, radius: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
x	[IN]	印字開始位置(X 座標、中心点)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字開始位置(Y 座標、中心点)	
radius	[IN]	円の半径	0000～0398

説明

設定された中心と半径で円を描画します。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
design.drawCircle(50, y: 50, radius: 15)
```

2.3.18 fillCircle メソッド

形式

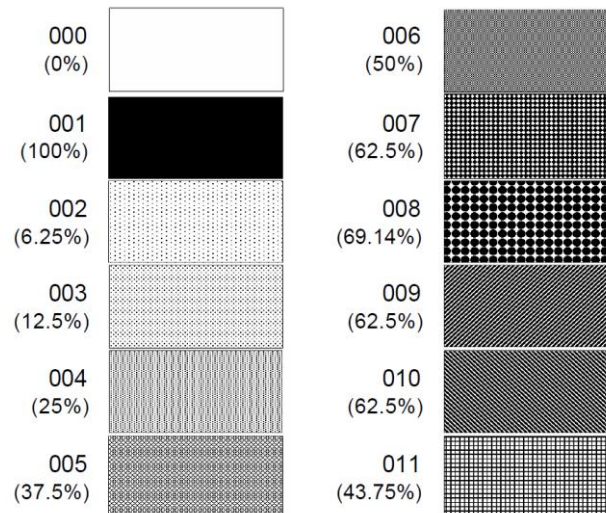
```
func fillCircle(x: Int32, y: Int32, radius: Int32, pattern: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
x	[IN]	印字開始位置(X 座標、中心点)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字開始位置(Y 座標、中心点)	
radius	[IN]	円の半径	0000～0398
pattern	[IN]	網掛けパターン(定数)	CLS_SHADED_PTN_0: 網掛けパターン 000 CLS_SHADED_PTN_1: 網掛けパターン 001 CLS_SHADED_PTN_2: 網掛けパターン 002 CLS_SHADED_PTN_3: 網掛けパターン 003 CLS_SHADED_PTN_4: 網掛けパターン 004 CLS_SHADED_PTN_5: 網掛けパターン 005 CLS_SHADED_PTN_6: 網掛けパターン 006 CLS_SHADED_PTN_7: 網掛けパターン 007 CLS_SHADED_PTN_8: 網掛けパターン 008 CLS_SHADED_PTN_9: 網掛けパターン 009 CLS_SHADED_PTN_10: 網掛けパターン 010 CLS_SHADED_PTN_11: 網掛けパターン 011

説明

設定された中心と半径で円を描画し、設定されたパターンで内部を網掛けします。



戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
design.fillCircle(100, y: 100, radius: 40, pattern: CLS_SHADED_PTN_2)
```

2.3.19 drawPolygon メソッド

形式

```
func drawPolygon(x: Array<Any>!, y: Array<Any>!) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
x	[IN]	印字位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	

説明

設定されたポイントで多角形を描画します。多角形の頂点の数だけ、x と y の座標を指定する。x と y の組み合わせは、最低 3 つ必要となり、3 つ未満のときは、CLS_E_ILLEGAL(1101)エラーとなる。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
let x: [Any] = [NSNumber(value: 100), NSNumber(value: 200),
               NSNumber(value: 250), NSNumber(value: 150)]
let y: [Any] = [NSNumber(value: 100), NSNumber(value: 100),
               NSNumber(value: 200), NSNumber(value: 200)]
design.drawPolygon(x, y: y) // 平行四辺形を描画
```

2.3.20 fillPolygon メソッド

形式

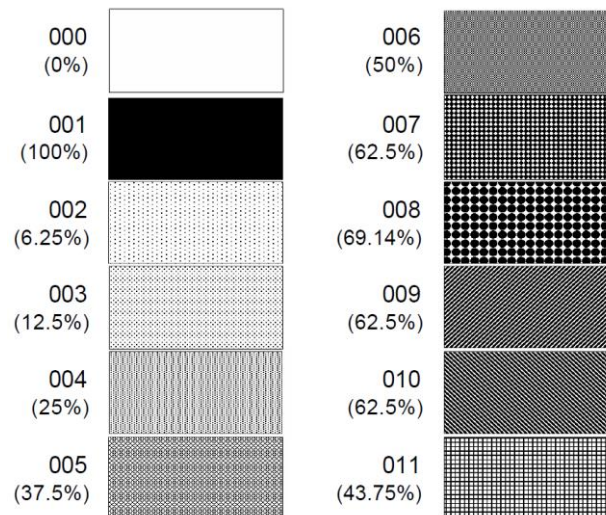
```
func fillPolygon(x: Array<Any>!, y: Array<Any>!, pattern: Int32) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
x	[IN]	印字位置(X 座標)	0000～9999 ※x,y は、左下を基準(0,0)として位置とする。
y	[IN]	印字位置(Y 座標)	
pattern	[IN]	網掛けパターン(定数)	CLS_SHADED_PTN_0: 網掛けパターン 000 CLS_SHADED_PTN_1: 網掛けパターン 001 CLS_SHADED_PTN_2: 網掛けパターン 002 CLS_SHADED_PTN_3: 網掛けパターン 003 CLS_SHADED_PTN_4: 網掛けパターン 004 CLS_SHADED_PTN_5: 網掛けパターン 005 CLS_SHADED_PTN_6: 網掛けパターン 006 CLS_SHADED_PTN_7: 網掛けパターン 007 CLS_SHADED_PTN_8: 網掛けパターン 008 CLS_SHADED_PTN_9: 網掛けパターン 009 CLS_SHADED_PTN_10: 網掛けパターン 010 CLS_SHADED_PTN_11: 網掛けパターン 011

説明

設定されたポイントで多角形を描画し、設定されたパターンで内部を網掛けします。多角形の頂点の数だけ、x と y の座標を指定する。x と y の組み合わせは、最低 3 つ必要となり、3 つ未満のときは、CLS_E_ILLEGAL(1101)エラーとなる。



戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

```
let x: [Any] = [NSNumber(value: 100), NSNumber(value: 200),
               NSNumber(value: 250), NSNumber(value: 150)]
let y: [Any] = [NSNumber(value: 100), NSNumber(value: 100),
               NSNumber(value: 200), NSNumber(value: 200)]
design.fillPolygon(x, y: y, pattern: CLS_SHADED_PTN_3); // 平行四辺形を描画
```

2.3.21 embedRawDesignCommand メソッド

形式

```
func embedRawDesignCommand(data: UnsafeMutablePointer<Int8>,
    withLength length: UInt) -> Int32
```

パラメータ

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	ラベルデザインコマンド	
length	[IN]	データ長	

説明

このメソッドは、ラベルデザインコマンドとして、バイトデータを挿入することができます。
ご使用の際は、他のメソッドに影響を与えない様に注意する必要があります。

戻り値

成功時は CLS_SUCCESS(0) を返します。それ以外のエラーコードは「[2.1 戻り値](#)」を参照してください。

使用例

「drawRect(100, 200, 300, 250, 100)」と同様のコマンドを、本メソッドを使用して挿入するとき。

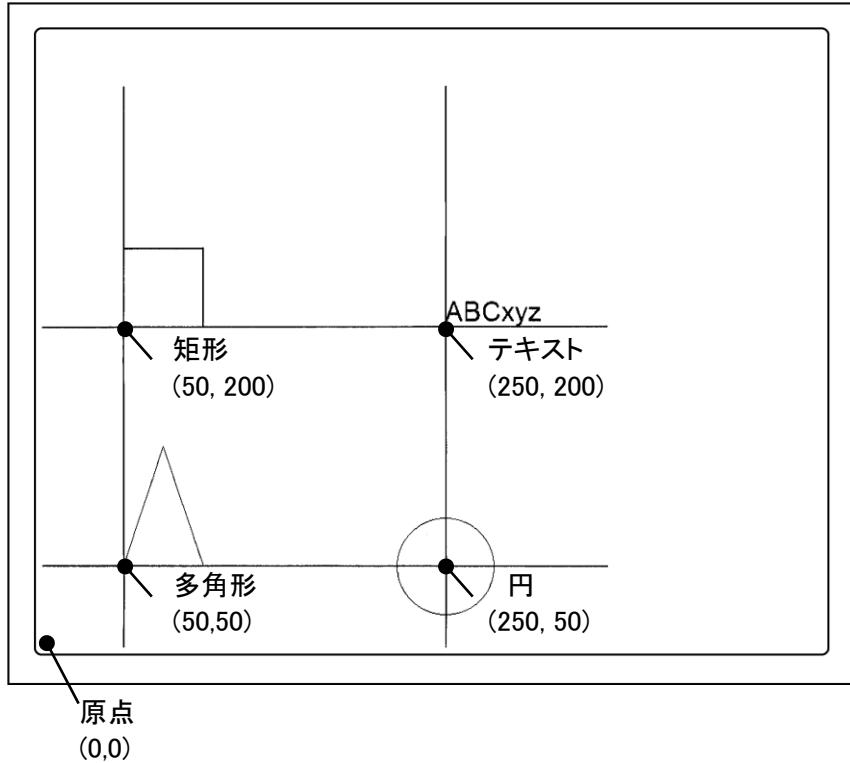
```
var hexdata: [Int8] = [
    0x31, 0x58, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x32, 0x30,
    0x30, 0x30, 0x31, 0x30, 0x30, 0x62, 0x30, 0x33, 0x30, 0x30,
    0x30, 0x32, 0x35, 0x30, 0x30, 0x31, 0x30, 0x30, 0x30, 0x31,
    0x30, 0x30, 0x0D, 0x0A ]

design.embedRawDesignCommand(&hexdata, withLength: 34);
```

3. 補足

3.1. 印字位置指定

ラベルにバーコードや文字を印字する場合の位置はラベルの左下が原点となり、原点からの距離を用いて印字位置指定を行います。原点より上方向の距離を Y 座標、右方向の距離を X 座標といいます。



使用例

```
// 多角形
let x: [Any] = [NSNumber(value: 50), NSNumber(value: 75),
               NSNumber(value: 100)]
let y: [Any] = [NSNumber(value: 50), NSNumber(value: 125),
               NSNumber(value: 50)]
design.drawPolygon(x, y: y)

// 矩形
design.drawRect(50, y: 200, width: 50, height: 50, thickness: 1)

// 円
design.drawCircle(250, y: 50, radius: 30)

// テキスト
design.drawTextLocalFont("ABCxyz",
                        fileName: "Arial Hebrew", rotation: CLS_RT_NORMAL,
                        hRatio: 100, vRatio: 100, point: 12,
                        style: CLS_FNT_DEFAULT, x: 250, y: 200)

// 罫線
design.drawLine(0, y1: 200, x2: 350, y2: 200, thickness: 1) // 罫線1
design.drawLine(50, y1: 0, x2: 50, y2: 350, thickness: 1) // 罫線2
design.drawLine(0, y1: 50, x2: 350, y2: 50, thickness: 1) // 罫線3
design.drawLine(250, y1: 0, x2: 250, y2: 350, thickness: 1) // 罫線4
```

3.2. ログ機能について

本 SDK は、メソッドの実行やプロパティの読み書きを記録するログ機能をサポートしています。ログ機能を設定する際は、[setLog メソッド](#)を使用するか、次の書式の設定ファイル「CSJLabelLib.cfg」をアプリケーションの共有フォルダに配置してください。

＜CSJLabelLib.cfg の例＞

[LogSetting]	…セクション名(固定)
LogMode=1	…ログモードを指定
LogPath=/	…ログファイルを格納するアプリケーション共有フォルダ内のフォルダを指定
LogMaxSize=10	…ログファイルの最大サイズを MB 単位で指定

設定項目

・ログモード

ログを記録するモードを指定します。

- 0: 記録なし
- 1: アクセス履歴の記録
- 2: エラーのみ記録

・格納フォルダ

ログファイルを格納するフォルダを指定します。本設定が指定されていない場合は、アプリケーション共有フォルダに格納されます。

・ログサイズ

ログファイルの最大容量を MB 単位で指定します。0 を指定した場合は容量制限が解除され可能な限り記録されます。

ログファイル名

ログファイルの拡張子は「.log」です。ファイル名は「CSJLabelLib」の後ろに曜日を表す数字が追加されます。曜日は日曜日を 0、月曜日を 1 として 0 から 6 のいずれかの数字が加えられます。

例) CSJLabelLib_1.log

ログファイルが既に存在し、それが当日以外の場合は、古いファイルを削除してからログを記録します。

ログフォーマット

ログ機能は、メソッド、プロパティの日付、時間、結果のアクセス情報を記録します。

--- メソッドの例1 (Connect) ---

```
2018/07/24 11:08:07.850 001 METHOD call    connect(1, 00:01:90:DF:CB:6D, 9100, 0)
2018/07/24 11:08:08.567 001 METHOD result connect() -> Success(0)
```

--- メソッドの例2 (Print) ---

```
-----Parameter Detail-----
drawTextPtrFont("Sample Print", 0, 10, 1, 1, 1, 8, 20, 300) -> 0
drawQRCode("DrawQRCode", 850, 1, 4, 3, 20, 220) -> 0
fillRect(20, 150, 350, 40, 11) -> 0
drawBarCode("0123456789", 104, 1, 3, 3, 30, 20, 70, 1) -> 0
-----
2018/07/24 11:08:25.519 001 METHOD result print() -> Success(0)
```


--- プロパティの設定例 ---

```
2018/07/24 11:08:29.716 001 PROPERTY set PrintDarkness <- 10 : Success(0)
```

--- プロパティの参照例 ---

```
2018/07/24 11:08:29.688 001 PROPERTY get PrintDarkness -> 999999
```

※ログ機能を使用する場合、全てのメソッドとプロパティのアクセス時にログファイルが更新されますので、SDK の処理が低下してしまうことがあります。

※次のような理由などでファイルの書き込みができない場合はログファイルの記録が行われません。このような場合エラーメッセージなどは表示されませんので、ご注意ください。

- ・書き込み禁止デバイスを指定した場合
- ・出力先に十分な領域が残っていない場合
- ・書き込み禁止のログファイルがある場合
- ・ファイルやフォルダのアクセス権がない場合
- ・他のアプリケーションがログファイルを使用している場合

3.3. パラメータ

No	項目	定数名	型	値	説明
1	処理結果	CLS_SUCCESS	int	0	正常終了
		CLS_E_CONNECTED	int	1001	接続済み
		CLS_E_DISCONNECT	int	1002	未接続
		CLS_E_NOTCONNECT	int	1003	接続失敗
		CLS_E_CONNECT_NOTFOUND	int	1004	未対応機種
		CLS_E_ILLEGAL	int	1101	未対応処理または無効パラメータ
		CLS_E_OFFLINE	int	1102	オフライン
		CLS_E_NOEXIST	int	1103	ファイルが存在しない
		CLS_E_FAILURE	int	1104	処理異常
		CLS_E_TIMEOUT	int	1105	書き込みタイムアウト
		CLS_E_NO_LIST	int	1106	プリンターが見つからない
		CLS_EPTR_BADFORMAT	int	1203	画像フォーマット異常
2	プロパティ デフォルト値	CLS_PROPERTY_DEFAULT	int	999999	プロパティのデフォルト値
3	プリンターの ステータス	CLS_STS_NO	int	0	ステータス:NO
		CLS_STS_YES	int	1	ステータス:YES
4	接続インター フェース	CLS_PORT_WiFi	int	0	ネットワーク接続
		CLS_PORT_Bluetooth	int	1	Bluetooth 接続
		CLS_PORT_USB	int	3	USB 接続
5	シリアル通信 条件	CLS_COM_BAUDRATE_1200	int	1200	ボーレート:1200
		CLS_COM_BAUDRATE_2400	int	2400	ボーレート:2400
		CLS_COM_BAUDRATE_4800	int	4800	ボーレート:4800
		CLS_COM_BAUDRATE_9600	int	9600	ボーレート:9600
		CLS_COM_BAUDRATE_19200	int	19200	ボーレート:19200
		CLS_COM_BAUDRATE_38400	int	38400	ボーレート:38400
		CLS_COM_BAUDRATE_57600	int	57600	ボーレート:57600
		CLS_COM_BAUDRATE_115200	int	115200	ボーレート:115200
		CLS_COM_PARITY_NONE	int	0	パリティ:NONE
		CLS_COM_PARITY_ODD	int	1	パリティ:ODD
		CLS_COM_PARITY_EVEN	int	2	パリティ:EVEN
		CLS_COM_HANDSHAKE_DTRDSR	int	0	フロー制御:DTR/DSR
		CLS_COM_HANDSHAKE_XONXOFF	int	1	フロー制御:XON/XOFF
6	ロケール	CLS_LOCALE_JP	int	0	日本モデル用ロケール
		CLS_LOCALE_OTHER	int	1	海外モデル用ロケール
		CLS_LOCALE_CN	int	2	中国モデル用ロケール
		CLS_LOCALE_KR	int	3	韓国モデル用ロケール
7	文字種	CLS_PRT_FNT_0	int	0	システムフォント:0
		CLS_PRT_FNT_1	int	1	システムフォント:1
		CLS_PRT_FNT_2	int	2	システムフォント:2
		CLS_PRT_FNT_3	int	3	システムフォント:3
		CLS_PRT_FNT_4	int	4	システムフォント:4
		CLS_PRT_FNT_5	int	5	システムフォント:5
		CLS_PRT_FNT_6	int	6	システムフォント:6
		CLS_PRT_FNT_7	int	7	システムフォント:7
		CLS_PRT_FNT_8	int	8	システムフォント:8

8	文字サイズ	CLS_PRT_FNT_TRIUMVIRATE	int	9	スムースフォント(Triumvirate)
		CLS_PRT_FNT_TRIUMVIRATE_B	int	10	スムースフォント(Triumvirate Bold)
		CLS_PRT_FNT_KANJI	int	11	漢字(横書き)
		CLS_PRT_FNT_KANJIT	int	12	漢字(縦書き)
		CLS_PRT_FNT_SIZE_4	int	0	文字種ポイント数(4pt)
		CLS_PRT_FNT_SIZE_5	int	1	文字種ポイント数(5pt)
		CLS_PRT_FNT_SIZE_6	int	2	文字種ポイント数(6pt)
		CLS_PRT_FNT_SIZE_8	int	3	文字種ポイント数(8pt)
		CLS_PRT_FNT_SIZE_10	int	4	文字種ポイント数(10pt)
		CLS_PRT_FNT_SIZE_12	int	5	文字種ポイント数(12pt)
		CLS_PRT_FNT_SIZE_14	int	6	文字種ポイント数(14pt)
		CLS_PRT_FNT_SIZE_18	int	7	文字種ポイント数(18pt)
		CLS_PRT_FNT_SIZE_24	int	8	文字種ポイント数(24pt)
		CLS_PRT_FNT_SIZE_30	int	9	文字種ポイント数(30pt)
		CLS_PRT_FNT_SIZE_36	int	10	文字種ポイント数(36pt)
		CLS_PRT_FNT_SIZE_48	int	11	文字種ポイント数(48pt)
		CLS_PRT_FNT_KANJI_SIZE_16	int	100	漢字文字種(16ドット)
		CLS_PRT_FNT_KANJI_SIZE_24	int	101	漢字文字種(24ドット)
		CLS_PRT_FNT_KANJI_SIZE_32	int	102	漢字文字種(32ドット)
		CLS_PRT_FNT_KANJI_SIZE_48	int	103	漢字文字種(48ドット)
9	エンコード	CLS_ENC_CDPG_DEFAULT	int	0	デフォルト
		CLS_ENC_CDPG_IBM037	int	37	IBM037 IBM EBCDIC (米国-カナダ)
		CLS_ENC_CDPG_IBM437	int	437	IBM437 OEM 米国
		CLS_ENC_CDPG_IBM500	int	500	IBM500 IBM EBCDIC (インターナショナル)
		CLS_ENC_CDPG_IBM737	int	737	ibm737 ギリシャ語 (DOS)
		CLS_ENC_CDPG_IBM775	int	775	ibm775 バルト言語 (DOS)
		CLS_ENC_CDPG_IBM850	int	850	ibm850 西ヨーロッパ語 (DOS)
		CLS_ENC_CDPG_IBM852	int	852	ibm852 中央ヨーロッパ言語 (DOS)
		CLS_ENC_CDPG_IBM855	int	855	IBM855 OEM キリル語
		CLS_ENC_CDPG_IBM857	int	857	ibm857 トルコ語 (DOS)
		CLS_ENC_CDPG_IBM860	int	860	IBM860 ポルトガル語 (DOS)
		CLS_ENC_CDPG_IBM861	int	861	ibm861 アイスランド語 (DOS)
		CLS_ENC_CDPG_IBM863	int	863	IBM863 フランス語 (カナダ)(DOS)
		CLS_ENC_CDPG_IBM865	int	865	IBM865 ノルウェー語 (DOS)
		CLS_ENC_CDPG_CP866	int	866	cp866 キリル語 (DOS)
		CLS_ENC_CDPG_IBM869	int	869	ibm869 ギリシャ語 モダン (DOS)
		CLS_ENC_CDPG_WINDOWS_874	int	874	windows-874 タイ語 (Windows)
		CLS_ENC_CDPG_CP875	int	875	cp875 IBM EBCDIC (ギリシャ語 モダン)
		CLS_ENC_CDPG_SHIFT_JIS	int	932	shift_jis 日本語 (シフト JIS)
		CLS_ENC_CDPG_GB2312	int	936	gb2312 簡体字中国語 (GB2312)
		CLS_ENC_CDPG_KS_C_5601_1987	int	949	ks_c_5601-1987 韓国語
		CLS_ENC_CDPG_BIG5	int	950	big5 繁体字中国語 (Big5)
		CLS_ENC_CDPG_IBM1026	int	1026	IBM1026 IBM EBCDIC (トルコ語 Latin-5)
		CLS_ENC_CDPG_UTF_16	int	1200	utf-16 Unicode
		CLS_ENC_CDPG_UNICODEFFFE	int	1201	unicodeFFFE Unicode (Big-Endian)

CLS_ENC_CDPG_WINDOWS_1250	int	1250	windows-1250 中央ヨーロッパ言語 (Windows)
CLS_ENC_CDPG_WINDOWS_1251	int	1251	windows-1251 キリル語 (Windows)
CLS_ENC_CDPG_WINDOWS_1252	int	1252	Windows-1252 西ヨーロッパ言語 (Windows)
CLS_ENC_CDPG_WINDOWS_1253	int	1253	windows-1253 ギリシャ語 (Windows)
CLS_ENC_CDPG_WINDOWS_1254	int	1254	windows-1254 トルコ語 (Windows)
CLS_ENC_CDPG_WINDOWS_1255	int	1255	windows-1255 ヘブライ語 (Windows)
CLS_ENC_CDPG_WINDOWS_1256	int	1256	windows-1256 アラビア語 (Windows)
CLS_ENC_CDPG_WINDOWS_1257	int	1257	windows-1257 バルト語 (Windows)
CLS_ENC_CDPG_WINDOWS_1258	int	1258	windows-1258 ベトナム語 (Windows)
CLS_ENC_CDPG_JOHAB	int	1361	Johab 韓国語 (Johab)
CLS_ENC_CDPG_MACINTOSH	int	10000	macintosh 西ヨーロッパ言語 (Mac)
CLS_ENC_CDPG_X_MAC_JAPANESE	int	10001	x-mac-japanese 日本語 (Mac)
CLS_ENC_CDPG_X_MAC_CHINESETRAD	int	10002	x-mac-chinesetrad 繁体字中国語 (Mac)
CLS_ENC_CDPG_X_MAC_KOREAN	int	10003	x-mac-korean 韓国語 (Mac)
CLS_ENC_CDPG_X_MAC_GREEK	int	10006	x-mac-greek ギリシャ語 (Mac)
CLS_ENC_CDPG_X_MAC_CYRILLIC	int	10007	x-mac-cyrillic キリル語 (Mac)
CLS_ENC_CDPG_X_MAC_CHINESESIMP	int	10008	x-mac-chinesesimp 簡体字中国語 (Mac)
CLS_ENC_CDPG_X_MAC_ROMANIAN	int	10010	x-mac-romanian ルーマニア語 (Mac)
CLS_ENC_CDPG_X_MAC_UKRAINIAN	int	10017	x-mac-ukrainian ウクライナ語 (Mac)
CLS_ENC_CDPG_X_MAC_CE	int	10029	x-mac-ce 中央ヨーロッパ語 (Mac)
CLS_ENC_CDPG_X_MAC_ICELANDIC	int	10079	x-mac-icelandic アイスランド語 (Mac)
CLS_ENC_CDPG_X_MAC_TURKISH	int	10081	x-mac-turkish トルコ語 (Mac)
CLS_ENC_CDPG_X_MAC_CROATIAN	int	10082	x-mac-croatian クロアチア語 (Mac)
CLS_ENC_CDPG_X_CHINESE_CNS	int	20000	x-Chinese-CNS 繁体字中国語 (CNS)
CLS_ENC_CDPG_US_ASCII	int	20127	us-ascii US-ASCII
CLS_ENC_CDPG_X_CP20261	int	20261	x-cp20261 T.61
CLS_ENC_CDPG_IBM290	int	20290	IBM290 IBM EBCDIC (日本語カカナ)
CLS_ENC_CDPG_KOI8_R	int	20866	koi8-r キリル語 (KOI8-R)
CLS_ENC_CDPG_EUC_JP_JIS	int	20932	EUC-JP 日本語 (JIS 0208-1990 and 0212-1990)
CLS_ENC_CDPG_X_CP20936	int	20936	x-cp20936 簡体字中国語 (GB2312-80)
CLS_ENC_CDPG_X_CP20949	int	20949	x-cp20949 韓国語 Wansung
CLS_ENC_CDPG_X_CP21027	int	21027	x-cp21027 Ext Alpha Lowercase
CLS_ENC_CDPG_KOI8_U	int	21866	koi8-u キリル語 (KOI8-R)
CLS_ENC_CDPG_ISO_8859_1	int	28591	iso-8859-1 西ヨーロッパ言語 (ISO)
CLS_ENC_CDPG_ISO_8859_2	int	28592	iso-8859-2 中央ヨーロッパ言語 (ISO)
CLS_ENC_CDPG_ISO_8859_4	int	28594	iso-8859-4 バルト語 (ISO)
CLS_ENC_CDPG_ISO_8859_5	int	28595	iso-8859-5 キリル語 (ISO)
CLS_ENC_CDPG_ISO_8859_7	int	28597	iso-8859-7 ギリシャ語 (ISO)
CLS_ENC_CDPG_ISO_8859_9	int	28599	iso-8859-9 トルコ語 (ISO)
CLS_ENC_CDPG_ISO_8859_13	int	28603	iso-8859-13 エストニア語 (ISO)
CLS_ENC_CDPG_ISO_8859_15	int	28605	iso-8859-15 ラテン語 9 (ISO)
CLS_ENC_CDPG_ISO_2022_JP	int	50220	iso-2022-jp 日本語 (JIS)

		CLS_ENC_CDPG_CSISO2022JP	int	50221	csISO2022JP 日本語 (JIS-Allow 1 byte Kana)
		CLS_ENC_CDPG_ISO_2022_JP_S	int	50222	iso-2022-jp 日本語 (JIS-Allow 1 byte Kana - SO/SI)
		CLS_ENC_CDPG_ISO_2022_KR	int	50225	iso-2022-kr 韓国語 (ISO)
		CLS_ENC_CDPG_X_CP50227	int	50227	x-cp50227 簡体字中国語 (ISO-2022)
		CLS_ENC_CDPG_EUC_JP	int	51932	euc-jp 日本語 (EUC)
		CLS_ENC_CDPG_EUC_CN	int	51936	EUC-CN 簡体字中国語 (EUC)
		CLS_ENC_CDPG_EUC_KR	int	51949	euc-kr 韓国語 (EUC)
		CLS_ENC_CDPG_HZ_GB_2312	int	52936	hz-gb-2312 簡体字中国語 (HZ)
		CLS_ENC_CDPG_GB18030	int	54936	GB18030 簡体字中国語 (GB18030)
		CLS_ENC_CDPG_UTF_7	int	65000	utf-7 Unicode (UTF-7)
		CLS_ENC_CDPG_UTF_8	int	65001	utf-8 Unicode (UTF-8)
10	文字スタイル	CLS_FNT_DEFAULT	int	0	無し
		CLS_FNT_BOLD	int	8	太字
		CLS_FNT_REVERSE	int	16	反転
		CLS_FNT_UNDERLINE	int	128	アンダーライン
		CLS_FNT_ITALIC	int	256	イタリック
		CLS_FNT_STRIKEOUT	int	512	取り消し線
11	回転方向	CLS_RT_NORMAL	int	1	回転無し(0°)
		CLS_RT_RIGHT90	int	2	右回転(90°)
		CLS_RT_ROTATE180	int	3	反転(180°)
		CLS_RT_LEFT90	int	4	左回転(270°)
12	バーコードの種類	CLS_BCS_CODE39	int	100	Code 3 of 9
		CLS_BCS_UPCA	int	101	UPC-A
		CLS_BCS_UPCE	int	102	UPC-E
		CLS_BCS_INTERLEAVED25	int	103	Interleaved 2 of 5
		CLS_BCS_CODE128	int	104	Code 128
		CLS_BCS_EAN13	int	105	EAN-13 (JAN-13)
		CLS_BCS_EAN8	int	106	EAN-8 (JAN-8)
		CLS_BCS_HIBC	int	107	HIBC
		CLS_BCS_CODABAR	int	108	CODABAR (NW-7)
		CLS_BCS_INT25	int	109	Int 2 of 5
		CLS_BCS_PLESSEY	int	110	Plessey
		CLS_BCS_CASECODE	int	111	CASE CODE
		CLS_BCS_UPC2DIG	int	112	UPC 2DIG ADD (UPC 用の 2 桁の補足コード)
		CLS_BCS_UPC5DIG	int	113	UPC 5DIG ADD (UPC 用の 5 桁の補足コード)
		CLS_BCS_CODE93	int	114	Code93
		CLS_BCS_ITF14	int	115	(国内モデル)ITF-14
		CLS_BCS_ZIP	int	116	(海外モデル)ZIP
		CLS_BCS_ITF16	int	117	(国内モデル)ITF-16
		CLS_BCS_UCCEAN128	int	118	(海外モデル)UCC/EAN-128
		CLS_BCS_INDUSTRIAL25	int	119	(国内モデル)Industrial 2 of 5
		CLS_BCS_UCCEAN128KMART	int	120	(海外モデル) UCC/EAN-128(for K-MART)
		CLS_BCS_COOP25	int	121	(国内モデル)COOP 2 of 5

		CLS_BCS_UCCEAN128RANDOMWEIGHT	int	122	(海外モデル) UCC/EAN-128 Random Weight
		CLS_BCS_TELEPEN	int	123	Telepen
13	バーコード文字の表示有無	CLS_BCS_TEXT_HIDE	int	0	非表示
		CLS_BCS_TEXT_SHOW	int	1	表示
14	エラー修正レベル (PDF417)	CLS_PDF417_EC_LEVEL_0	int	0	レベル 0
		CLS_PDF417_EC_LEVEL_1	int	1	レベル 1
		CLS_PDF417_EC_LEVEL_2	int	2	レベル 2
		CLS_PDF417_EC_LEVEL_3	int	3	レベル 3
		CLS_PDF417_EC_LEVEL_4	int	4	レベル 4
		CLS_PDF417_EC_LEVEL_5	int	5	レベル 5
		CLS_PDF417_EC_LEVEL_6	int	6	レベル 6
		CLS_PDF417_EC_LEVEL_7	int	7	レベル 7
		CLS_PDF417_EC_LEVEL_8	int	8	レベル 8
15	エラー修正レベル(Data Matrix)	CLS_DATAMATRIX_EC_LEVEL_200	int	200	
16	エラー修正レベル(QR Code)	CLS_QRCODE_EC_LEVEL_L	int	0	エラー修正レベル L(7%)
		CLS_QRCODE_EC_LEVEL_M	int	1	エラー修正レベル M(15%)
		CLS_QRCODE_EC_LEVEL_Q	int	2	エラー修正レベル Q(25%)
		CLS_QRCODE_EC_LEVEL_H	int	3	エラー修正レベル H(30%)
17	エラー修正レベル(Aztec)	CLS_AXTEC_EC_LEVEL_000	int	0	誤り訂正率 23%
18	バーコードタイプ(GS1 DataBar)	CLS_GS1_DATABAR_OMNI_DIRECTIONAL	int	0	GS1DataBar Omni-directional
		CLS_GS1_DATABAR_COMPOSITE	int	1	GS1DataBar Composite
		CLS_GS1_DATABAR_TRUNCATION	int	2	GS1DataBar Truncation
		CLS_GS1_DATABAR_STACKED	int	3	GS1DataBar Stacked
		CLS_GS1_DATABAR_STACKED_OMNI_DIRECTIONAL	int	4	GS1DataBar Stacked Omni-directional
		CLS_GS1_DATABAR_LIMITED	int	5	GS1DataBar Limited
		CLS_GS1_DATABAR_EXPANDED	int	6	GS1DataBar Expanded
19	網掛けパターン	CLS_SHADED_PTN_0	int	0	網掛けパターン:000
		CLS_SHADED_PTN_1	int	1	網掛けパターン:001
		CLS_SHADED_PTN_2	int	2	網掛けパターン:002
		CLS_SHADED_PTN_3	int	3	網掛けパターン:003
		CLS_SHADED_PTN_4	int	4	網掛けパターン:004
		CLS_SHADED_PTN_5	int	5	網掛けパターン:005
		CLS_SHADED_PTN_6	int	6	網掛けパターン:006
		CLS_SHADED_PTN_7	int	7	網掛けパターン:007
		CLS_SHADED_PTN_8	int	8	網掛けパターン:008
		CLS_SHADED_PTN_9	int	9	網掛けパターン:009
		CLS_SHADED_PTN_10	int	10	網掛けパターン:010
		CLS_SHADED_PTN_11	int	11	網掛けパターン:011
20	単位	CLS_UNIT_MILLI	int	0	ミリ
		CLS_UNIT_INCH	int	1	インチ
21	速度設定	CLS_SPEEDSETTING_1	int	1	1:1.0 インチ(25.4mm)／秒
		CLS_SPEEDSETTING_2	int	2	2:2.0 インチ(50.8mm)／秒
		CLS_SPEEDSETTING_3	int	3	3:3.0 インチ(76.2mm)／秒
		CLS_SPEEDSETTING_4	int	4	4:4.0 インチ(101.6mm)／秒

		CLS_SPEEDSETTING_5	int	5	5:5.0 インチ(127.0mm)/秒
		CLS_SPEEDSETTING_6	int	6	6:6.0 インチ(152.4mm)/秒
		CLS_SPEEDSETTING_7	int	7	7:7.0 インチ(177.8mm)/秒
		CLS_SPEEDSETTING_8	int	8	8:8.0 インチ(203.2mm)/秒
		CLS_SPEEDSETTING_9	int	9	9:9.0 インチ(228.6mm)/秒
		CLS_SPEEDSETTING_A	int	10	A:1.0 インチ(25.4mm)/秒
		CLS_SPEEDSETTING_B	int	11	B:1.0 インチ(25.4mm)/秒
		CLS_SPEEDSETTING_C	int	12	C:2.0 インチ(50.8mm)/秒
		CLS_SPEEDSETTING_D	int	13	D:2.0 インチ(50.8mm)/秒
		CLS_SPEEDSETTING_E	int	14	E:3.0 インチ(76.2mm)/秒
		CLS_SPEEDSETTING_F	int	15	F:3.0 インチ(76.2mm)/秒
		CLS_SPEEDSETTING_G	int	16	G:4.0 インチ(101.6mm)/秒
		CLS_SPEEDSETTING_H	int	17	H:4.0 インチ(101.6mm)/秒
		CLS_SPEEDSETTING_I	int	18	I:5.0 インチ(127.0mm)/秒
		CLS_SPEEDSETTING_J	int	19	J:5.0 インチ(127.0mm)/秒
		CLS_SPEEDSETTING_K	int	20	K:6.0 インチ(152.4mm)/秒
		CLS_SPEEDSETTING_L	int	21	L:6.0 インチ(152.4mm)/秒
		CLS_SPEEDSETTING_M	int	22	M:7.0 インチ(177.8mm)/秒
		CLS_SPEEDSETTING_N	int	23	N:7.0 インチ(177.8mm)/秒
		CLS_SPEEDSETTING_O	int	24	O:8.0 インチ(203.2mm)/秒
		CLS_SPEEDSETTING_P	int	25	P:8.0 インチ(203.2mm)/秒
		CLS_SPEEDSETTING_Q	int	26	Q:9.0 インチ(228.6mm)/秒
		CLS_SPEEDSETTING_R	int	27	R:9.0 インチ(228.6mm)/秒
		CLS_SPEEDSETTING_S	int	28	S:10.0 インチ(254.0mm)/秒
		CLS_SPEEDSETTING_T	int	29	T:10.0 インチ(254.0mm)/秒
		CLS_SPEEDSETTING_U	int	30	U:11.0 インチ(279.4mm)/秒
		CLS_SPEEDSETTING_V	int	31	V:11.0 インチ(279.4mm)/秒
		CLS_SPEEDSETTING_W	int	32	W:12.0 インチ(304.8mm)/秒
		CLS_SPEEDSETTING_X	int	33	X:12.0 インチ(304.8mm)/秒
22	印刷後動作 設定	CLS_MEDIAHANDLING_NONE	int	0	なし
		CLS_MEDIAHANDLING_TEAROFF	int	1	ティアオフ
		CLS_MEDIAHANDLING_DISPENSES	int	2	ディスペンス
		CLS_MEDIAHANDLING_PAUSE	int	3	ポーズ
		CLS_MEDIAHANDLING_CUT	int	4	カット
		CLS_MEDIAHANDLING_CUTANDPAUSE	int	5	カット、ポーズ
		CLS_MEDIAHANDLING_PEELOFF	int	6	剥離
		CLS_MEDIAHANDLING_REWIND	int	7	リワインダー
23	センサー選 択(透過/反 射/なし)	CLS_SELSENSOR_NONE	int	0	なし
		CLS_SELSENSOR_SEETHROUGH	int	1	透過
		CLS_SELSENSOR_REFLECT	int	2	反射
24	印字方法 (TT/DT)	CLS_PRTMETHOD_TT	int	0	熱転写
		CLS_PRTMETHOD_DT	int	1	感熱
25	センサー選 択(前方/後 方)	CLS_SENS_LOCATION_FRONT	int	0	フロントセンサ
		CLS_SENS_LOCATION_ADJUSTABLE	int	1	アジャスタブルセンサ

