TOSHIBA
Leading Innovation

Toshiba 4610 Printer
Android and iOS SDK Users Guide

Summary of Changes

Changes resulting in document revisions will be summarized in this table in reverse chronological sequence. Revision bars (|) will highlight the text changed in new document versions.

Version	Release Date	Change Description
Draft	04/22/2014	Beta release

Table of Contents

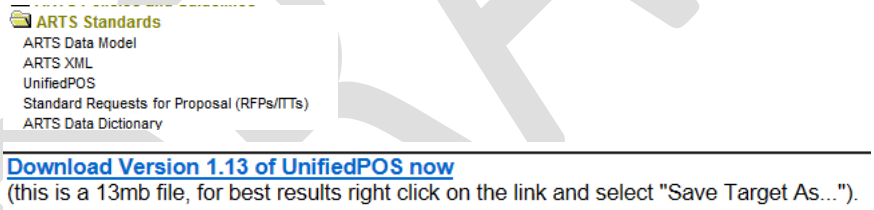
1.0 Overview

1.1 Introduction

In this document, the terms Toshiba or Toshiba Global Commerce Solutions (TGCS) are used interchangeably. This document provides information regarding Application Programming Interface (API) to directly communicate with Toshiba 4610 Ethernet Printers using Android and iOS devices. It also provides details on:

- Requirements for Android and iOS build environment
- How to integrate the API libraries into your project
- Build the project
- Sample code to demonstrate the use of Toshiba SDK

The Toshiba SDK for Android and iOS SDK is built to run as a **static library** on your mobile application, and allow your application to interact with Toshiba 4610 Multi-function Ethernet Printers. Toshiba Global Commerce Solution has chosen to implement SDK for Android and iOS based on the UnifiedPOS Standards. The OPOS and JavaPOS are implementations of UnifiedPOS Standards. If you are familiar with OPOS and JavaPOS, then the Toshiba SDK is natural extension of that for POSPrinter, MICR and CashDrawer device categories. For more details, please download Version 1.13 of the Standards from http://legacy.nrf-arts.org/, click on UnifiedPOS



1.2 Device Configuration

For iOS and Android SDK support, the Toshiba 4610 Ethernet Printers are configured through a configuration file. The configuration file can be manually created or it can be generated automatically generated by an external Toshiba 4610 Ethernet Printer Configuration utility. The configuration file details are described below.

Configuration file name:

aip4610.efg

Configuration File Format:

[4610Printer] // section 1 for 1st Ethernet printer

DeviceAddress: 9.44.162.101

DeviceName: 4610-46008C

LogicalName: ePrinter1

Description: 4610-2CR at Location 1

[4610Printer] // section 2 for 2nd Ethernet printer

DeviceAddress: 9.44.162.102

DeviceName: 4610-47009D

LogicalName: ePrinter2

Description: 4610-1NR at Locaton 2

…..

……

Where

- DeviceAddress = IP address of the Printer. The IP address can be used during open() call.
- DeviceName = host name assigned to the Printer. In addition to IP address, the printer can be accessed by host name. If host name is specified in the configuration file, the printer shall be accessed by host name rather than IP address. The device can be accessed using the host name as "hostname.xxx.yyy.com" convention.
- LogicalName = Logical Name of the printer is used in open( ) call
- Description = Description of the printer or location.

1.3 Supported Android and iOS Environment

iOS:

- OS version 6.x and later compatible versions
- Devices:
  - iPhone 4, 4S, 5S/C
  - iPod touch 4th generation and 5th generation
  - iPad 2/iPad 4th generation/iPad mini

Android:

- Devices with OS version 4.0 and later compatible versions

2.0 Development Environment Setup

2.1 iOS

The prerequisite for the iOS version of the SDK is XCode 4.5 or later. Installing XCode will set up the development environment required for the API to be integrated with your mobile application.

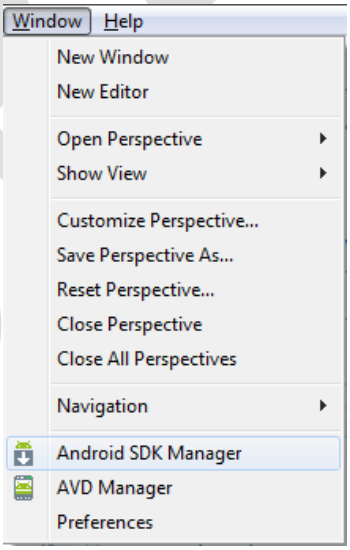
2.2 Android

To setup the Eclipse-based Android Development environment required for this project, follow the installation instructions for the given links below.

SDK/IDE	URL
Android Developer Tools (ADT)	http://developer.android.com/sdk/index.html
Eclipse C/C++ Development Tooling (CDT)	http://www.eclipse.org/cdt/
Android Native Development Kit (NDK)	http://developer.android.com/tools/sdk/ndk/index.html

2.3 Install Android SDK Version(s)

Install the required SDK version(s) via the Android SDK Manager. Select *Window* → *Android SDK Manager* from Eclipse menu or use the `tools/android` program from your Android SDK installation.



This project supports Android SDK 4.0 and above.

Configure NDK Build Configuration

For building convenience, configure an NDK Build Configuration in Eclipse. Instructions for doing so may be found here: http://mobilepearls.com/labs/ndk-builder-in-eclipse/.

2.4 Enabling Device for Development

Android devices are required to have Developer Options enabled for development. Enabling Developer Tools on Android devices vary from vendor to vendor. Detailed instructions on enabling Android devices for development may be found here - http://developer.android.com/tools/device.html.

3.0 Sample Code

To jump start development, working sample is provided for both iOS and Android.

3.1 iOS Sample Code

The sample code along with project workspace is included in samplecode\src directory structure. You should be able to build the sample code and exercise Ethernet Printer in no time.

Below is a screen shot of the sample code utility. The utility provides sample code to demonstrate the use of basic commands, printing receipts in Synchronous and Asynchronous mode, and printing a receipt in Transaction Mode.

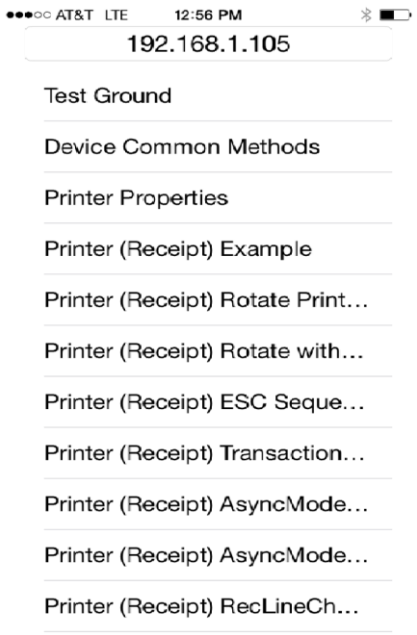


Figure 1 Screen shot of iOS Sample Application

3.2 Android Sample Code

The sample code along with project workspace is included in samplecode\src directory structure. You should be able to build the sample code and exercise the Ethernet printer within a short amount of time.

Pre-built apk:

In addition, under samplecode\bin directory structure you will find ready to install apk. It can be loaded onto Android device to test the Ethernet printer/

Below is a screen shot of the sample code utility. The utility provides sample code to demonstrate the use of basic commands, printing receipts in Synchronous and Asynchronous mode, and printing a receipt in Transaction Mode.

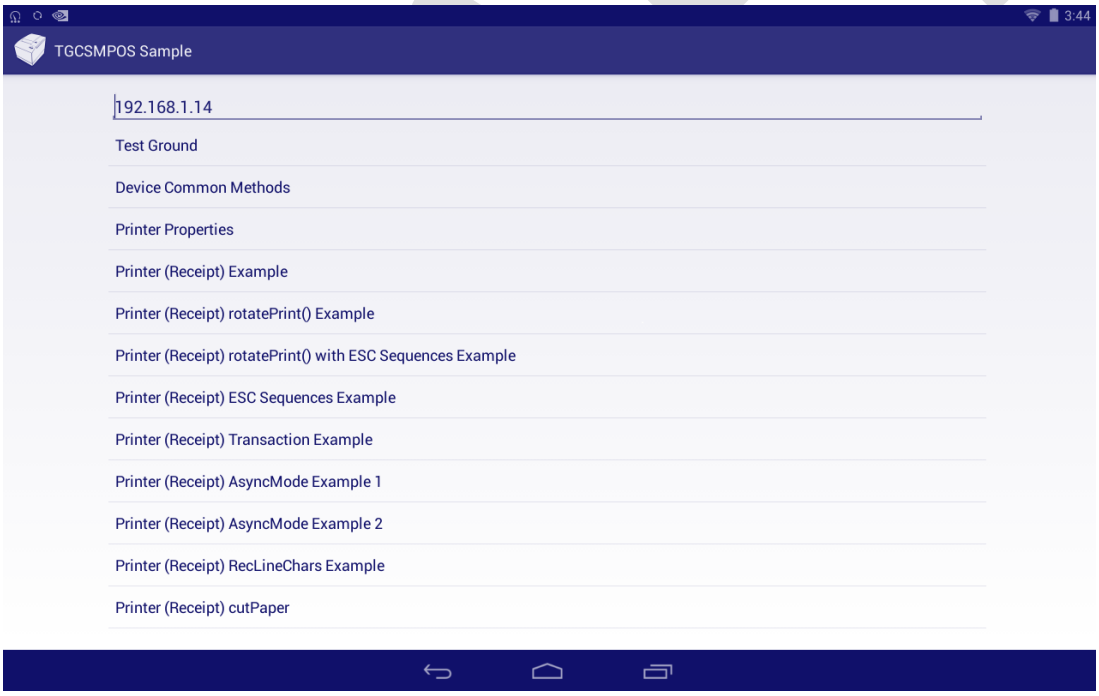


Figure 2 Screen shot of Android Sample Application

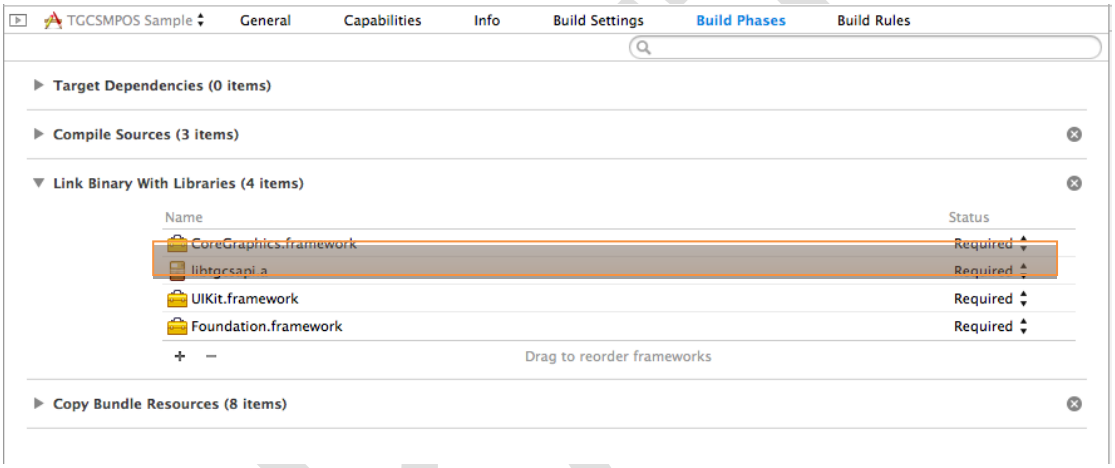
4.0 Importing TGCS API Library

4.1 iOS

The following diagram illustrates the integration of the API as a static library in your project.

Herein demonstrates the steps to import the API library into your project.

1) Obtain the API binary file and header files from lib\ios\<arch>
2) Drag and drop the libtgcsapi.a file and include files into your project.
3) Under Build Phases ➔ Link Binary with Libraries, verify that the library is linked with the project.



4) Include `TGCSMPOS.h` in your `.h`, `.m` or `.mm` file.

```
#import "TGCSMPOS.h"
```

5) In the first ViewController's `viewDidLoad` method, call the necessary setup methods to initialize the API core engine.

```
////////////////////////////////////////////
// Necessary setup calls
////////////////////////////////////////////

// Initialize log path
_base_path = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES) objectAtIndex:0];
[TGCSMPOSDevice setupLogPath:_base_path];

// initialize resources
 [TGCSMPOSDevice initialize];
```

6) Instantiate the virtual device instance and make the necessary method* calls. Refer to *TGCSMPOS Sample App* for more examples.

```
// instantiate a virtual instance of the device
TGCSMPOS4610Printer *device = [[TGCSMPOS4610Printer alloc] init];


///// ommitted code /////
```

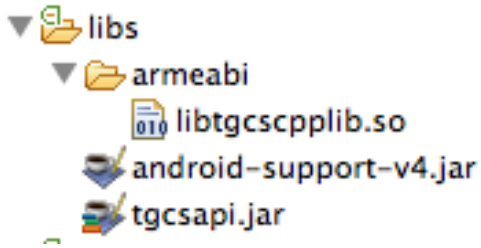
```
try {
        // open the device to instantiate resources
        [device open:logicalDeviceName];

        // claim the device for exclusive access; 3000ms timeout
        [device claim:3000];

        // start transaction
       [device transactionPrintStation:PTR_S_RECEIPT control:PTR_TP_TRANSACTION];

        // logo
        [device printBitmapWithImage:logo Station:PTR_S_RECEIPT width:576
    alignment:PTR_BC_CENTER];

        // content
        [device printNormalStation:PTR_S_RECEIPT data:receipt_content];

        // barcode
        [device printBarCodeStation:PTR_S_RECEIPT data:@"0123456789"
    symbology:PTR_BCS_Code39 height:100 width:2 alignment:PTR_BC_CENTER
    textPosition:PTR_BC_TEXT_BELOW];

       [device printNormalStation:PTR_S_RECEIPT data:@"\n\n\n\n\n\n"];

       // dispense
        [device cutPaper:100];

       // end transaction
       [device transactionPrintStation:PTR_S_RECEIPT control:PTR_TP_NORMAL];

       // relinquish exclusive access to the device
       [device _release];

       // close the device to release resources
       [device close];


} catch(NSException *e) {
        error = [e reason];
        NSLog(@"%@", [e reason]);
        try {
       [device close];
    } catch(NSException *e) {
      // in the event open() fails.
      NSLog(@"%@", [e reason]);
    }

}
```

*** It is good practice to surround method calls with `try {} catch {}` braces to catch any exceptions that the API may throw.**

4.2 Android

The following instructions demonstrate how to import the Android SDK binary to a project.

1. Copy, paste and merge the contents of the `libs\android` folder in your Android Project.
2. Your final import should look like this.



3. Import the library class files.

```
import com.toshiba.tgcsapi.*;
```

4. Instantiate the virtual device instance and make the necessary method* calls. Refer to *TGCSMPOS Sample App* for more examples.

```
// instantiate the device instance
TGCSMPOSDevice device = new TGCSMPOSDevice();

// omitted code
try {
      // open the device
      device.open(logicalDeviceName);

      // claim the device for exclusive access; 3000ms timeout
      device.claim(3000);

      // enable the device
      device.setDeviceEnabled(true);

      // Omitted code

      // relinquish exclusive access to the device
      device.release();

      // close the device
      device.close();
} catch (RuntimeException e) {
      res += e.getMessage() + "\n";
}
```

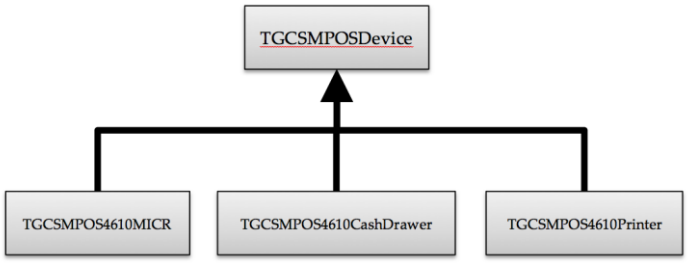
*** It is good practice to surround method calls with `try {} catch {}` braces to catch any exceptions that the API may throw.**

5. Build and Run the project.

   NOTE: The `.jar` and the `.so` files must be built to the right architecture and Android SDK version.

5.0 Toshiba SDK API

The following diagram illustrates the class relationship between the TGCSMPOS classes.



The following list contains the methods supported by the SDK and their respective owner classes. The behavior of each of the method follows the UnifiedPOS Version 1.13 very closely. Any accommodation for the 4610 Printer device is done transparently within the method.

5.1 SDK API Reference:

You will find browsed based comprehensive documentation in “doc” directory.

5.2 Methods supported

The APIs are based on UnifiedPOS Standards, and are common to both iOS and Android SDK. The APIs are subset of methods/properties supported by UnifiedPOS Standards for POSPrinter, MICR and CashDrawer device categories.

Virtual Device Class	Method	Global	Async	Event Driven
TGCSMPOSDevice	ConfiguredDevice[] getConfiguredDevices()	□		
	void:loadConfigurationFile(configurationFile:string)	□		
	void:loadMICRExceptionFile(exceptionFile:string)	□		
	void:open(logicalDeviceName:string)			
	void:close()			
	void:claim(timeout:int32)			
	void:release()			
	void:directIO(command:int32, data:int32, obj:object)			
	void:clearInput()			
	void:clearOutput()			
	bool:getClaimed()			
	bool:getDeviceEnabled()			
	void:setDeviceEnabled(value:bool)			

	bool:getDataEventEnabled()			
	void:setDataeventEnabled(value:bool)			
	int32:getOutputID()			
	string:getPhysicalDeviceName()			
TGCSMPOS4610Printer	void:beginInsertion(timeout:int32)			
	void:endInsertion()			
	void:beginRemoval(timeout:int32)			
	void:endRemoval()			
	void:changePrintSide(side:int32)			
	void:cutPaper(percentage:int32)		□	
	void:printBarCode(station:int32, data:string, symbology:int32, height:int32, width:int32, alignment:int32, textPosition:int32)		□	
	void:printBitmap(station:int32, fileName:string, width:int32, alignment:int32)		□	
	void:printNormal(station:int32, data:string)		□	
	void:rotatePrint(station:int32, rotation:int32)		□	
	void:transactionPrint(station:int32, control:int32)			
	bool:getAsyncMode()			
	void:setAsyncMode(value:bool)			
	bool:getCapSlpLeft90()			
	bool:getCapSlpPresent()			
	bool:getCapSlpRight90()			
	bool:getCapSlpRotate180()			
	int32:getCharacterSet()			
	void:setCharacterSet(value:int32)			
	string:getCharacterSetList()			
	bool:getCoverOpen()			
	int32:getErrorLevel()			
	bool:getFlagWhenIdle()			
	void:setFlagWhenIdle(value:bool)			
	int32:getMapMode()			
	void:setMapMode(value:int32)			
	int32:getRecLineChars()			
	void:setRecLineChars(value:int32)			
	string:getRecLineCharsList()			
	int32:getRecLineHeight()			
	int32:getRecLIneWidth()			
	bool:getRecEmpty()			
	int32:getSlpLineChars()			
	void:setSlpLineChars(value:int32)			
	string:getSlpLineCharsList()			

	int32:getSlpLineHeight()			
	void:setSlpLineHeight(value:int32)			
	int32:getSlpLineSpacing()			
	void:setSlpLIneSpacing(value:int32)			
	bool:getSlpEmpty()			
	int32:getSlpPrintSide()			
	void:setSlpPrintSide(side:int32)			
TGCSMPOS 4610MICR	void:beginInsertion(timeout:int32)			
	void:endInsertion()			
	void:beginRemoval(timeout:int32)			
	void:endRemoval()			
	string:getAccountNumber()			□
	string:getBankNumber()			□
	int32:getCheckType()			□
	int32:getCountryCode()			□
	string:getEPC()			□
	string:getRawData()			□
	string:getSerialNumber()			□
	string:getTransitNumber()			□
TGCSMPOS4610Cash Drawer	void:openDrawer()			
	void:waitForDrawerClose(beepTimeout:int32, beepFrequency:int32, beepDuration:int32, beepDelay:int32)			
	bool:getDrawerOpened()			

Legend

Async – Methods support asynchronous mode.

Global – Methods impact the API on a global level.

Event Driven – Methods are dependent on POSDataEvent for validity.

5.3 4610 Printer Method Override

Methods that may differ marginally from the UnifiedPOS Specification are listed in this section. Methods listed here differ due to the device's support for the operation or are worthy of mention due to special attention they need. Methods listed in the UPOS Supported Methods but not mentioned here follow the operational description as depicted in the UnifiedPOS Specification (version 1.13).

5.4 TGCSMPOSDevice::open(logicalDeviceName:string)

This method instantiates the virtual resources for required by the device instance to communicate, control and track the status of the POS Device. The logicalDeviceName can be either the web address, IP4 address of the device OR the logical name specified in the Configuration File (.efg) file supported by the API.

Herein illustrates sample content of an .efg file. Refer to the ethernetPrinterSample.efg file for more information.

```
#Example to configure a printer with a DeviceAddress with minimum
information
[4610Printer]
DeviceAddress: 192.168.1.19
DeviceName:       test
DeviceTcpPort:    9100
DeviceUdpPort:    9100
PossPortNumber:   1
```

5.5 TGCSMPOS4610Printer::cutPaper(percentage:int32)

This initiates a cut paper action on the receipt station of the 4610 Multifunction Receipt Printer. The percentage attribute, though necessary, is ignored.

5.6 TGCSMPOS4610Printer::printBarCode(station:int32, data:string, symbology:int32, height:int32, width:int32, alignment:int32, textPosition:int32)

This method sends data to be converted to barcode format to print. There are limitations to the valid values passed into this method. Should any value not be within range of the supported values, an exception may be thrown to indicate so.

station:

PTR_S_RECEIPT

PTR_S_SLIP

data:

Depends on the selected symbology. Error is handled on the printer.

symbology:

PTR_BCS_UPCA

PTR_BCS_UPCE

PTR_BCS_JAN13

PTR_BCS_JAN8

PTR_BCS_Code39

PTR_BCS_ITF

PTR_BCS_Codabar

PTR_BCS_Code128

PTR_BCS_Code93

PTR_BCS_GS1DATABAR

PTR_BCS_GS1DATABAR_S

PTR_BCS_GS1DATABAR_E

PTR_BCS_GS1DATABAR_E_S

PTR_BCS_PDF417

PTR_BCS_QRCODE

height:

For Receipt Station, 1 <= height <= 255

For Slip Station, 3 <= height <= 5

width:

2 <= width <= 4

alignment:

PTR_BC_LEFT

PTR_BC_CENTER

PTR_BC_RIGHT

textPosition:

PTR_BC_TEXT_NONE

PTR_BC_TEXT_ABOVE

PTR_BC_TEXT_BELOW

PTR_BC_TEXT_BOTH

5.7 TGCSMPOS4610Printer::printBitmap(station:int32, filename:string, width:int32, alignment:int32)

This method initiates the print bitmap operation on the printer. The supported formats are: JPEG, BMP, PNG and TIFF. This method enforces the restraints that the printer can support onto the images passed to this method. As such, the passed image will be scaled to the best fit for the printer and dithered for the best print result on a 2-tone printer.

Bitmap printing limitation of the device:

For station = PTR_S_RECEIPT (Receipt Station),

Max Width: 576 or 400 pixels (depending on the receipt paper loaded)

Max Height: 904 or 1304 pixels (depending on the receipt paper loaded)

For station = PTR_S_SLIP (Document Insert)

Max Width: 472 pixels

Max Height: 40 pixels

Alignment:

PTR_BM_LEFT

PTR_BM_CENTER

PTR_BM_RIGHT

5.8 TGCSMPOS4610Printer::printNormal(station int32, data:string)

This method is used to print text, escape sequences, and barcodes. This method is performed synchronously if AsyncMode is false and asynchronously if AsyncMode is true. This method's operation follows that as documented in the UnifiedPOS Specifications version 1.13.

5.9 TGCSMPOS4610Printer::rotatePrint(station:int32, rotation:int32)

This method will alter the cached tracking of the rotation to be performed. The use of this method involves switching between PTR_RP_LEFT90, PTR_RP_RIGHT90, PTR_RP_ROTATE180 and PTR_RP_NORMAL. The SDK will buffer text data to be printed until PTR_RP_NORMAL is sent.

Eg.

```
device.rotatePrint(POS.PTR_S_RECEIPT, POS.PTR_RP_RIGHT90);
device.printNormal(POS.PTR_S_RECEIPT, "Line 1 rotPrintR90 test\n");
device.printNormal(POS.PTR_S_RECEIPT, "Line 2 rotPrintR90 test\n");
device.rotatePrint(POS.PTR_S_RECEIPT, POS.PTR_RP_NORMAL);

device.rotatePrint(POS.PTR_S_RECEIPT, POS.PTR_RP_LEFT90);
device.printNormal(POS.PTR_S_RECEIPT, "Line 1 rotPrintL90 test\n");
device.printNormal(POS.PTR_S_RECEIPT, "Line 2 rotPrintL90 test\n");
device.rotatePrint(POS.PTR_S_RECEIPT, POS.PTR_RP_NORMAL);

device.rotatePrint(POS.PTR_S_RECEIPT, POS.PTR_RP_ROTATE180);
device.printNormal(POS.PTR_S_RECEIPT, "Line 1 rotPrint180 test\n");
device.printNormal(POS.PTR_S_RECEIPT, "Line 2 rotPrint180 test\n");
device.rotatePrint(POS.PTR_S_RECEIPT, POS.PTR_RP_NORMAL);
device.printNormal(POS.PTR_S_RECEIPT, "regular text\n");
```

5.10 Support for Escape Sequences

The following table illustrates the Escape Sequences supported by the TGCS Mobile SDK. For more information about the usage of Escape Sequences, refer to Data Characters and Escape Sequences of the UnifiedPOS Specification version 1.13.