Cordova Plugin for Thermal Printer's

npm v1.0.6 downloads 233/month

This plugin is a wrapper for the Android library for ESC/POS Thermal Printer. Forked from paystory-de/thermal-printer-cordova-plugin Added a method bitmapToHexadecimalStringLarge for large image.

Install

Cordova

```
$ cordova plugin add thermal-printer-cordova-plugin
```

Ionic

```
$ ionic cordova plugin add thermal-printer-cordova-plugin
```

Capacitor

```
$ npm install thermal-printer-cordova-plugin
$ npx cap sync
```

Don't forget to add BLUETOOTH and INTERNET (for TCP) permissions and for USB printers the `android.hardware.usb.host` feature to the `AndroidManifest.xml`.

```
<uses-feature android:name="android.hardware.usb.host" />
<uses-permission android:maxSdkVersion="30"
android:name="android.permission.BLUETOOTH" />
<uses-permission android:maxSdkVersion="30"
android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
```

Run this for getting Bluetooth access permission if needed

```
ThermalPrinter.requestBTPermissions({type: 'bluetooth'}, function(result){
console.log(result) }, function(error){ console.log(error) });
```

Examples

Notice for TypeScript-Developers

You can easily import and use the ThermalPrinter plugin in your TypeScript-Projects.

```
import { ThermalPrinterPlugin } from 'thermal-printer-cordova-plugin/src';

declare let ThermalPrinter: ThermalPrinterPlugin;
```

And then use the following examples in your code.

Print via Bluetooth

Printing via Bluetooth is as easy as possible.

```
ThermalPrinter.printFormattedText({
    type: 'bluetooth',
    id: 'first', // You can also use the identifier directly i. e.
00:11:22:33:44:55 (address) or name
    text: '[C]<u><font size='big'>Hello World</font></u>' // new lines
with "\n"
}, function() {
    console.log('Successfully printed!');
}, function(error) {
    console.error('Printing error', error);
});
```

Notice: If not working please ensure that you have the printer connected. (Settings -> Bluetooth -> Pairing) If you have other issues maybe you have not granted the `android.permission.BLUETOOTH` permission.

Print via TCP

Printing via TCP is as easy as possible.

```
ThermalPrinter.printFormattedText({
    type: 'tcp',
    address: '192.168.1.123',
    port: 9100,
    id: 'tcp-printer-001', // Use an unique identifier for each printer i.
e. address:port or name
    text: '[C]<u><font size='big'>Hello World</font></u>' // new lines
with "\n"
}, function() {
    console.log('Successfully printed!');
}, function(error) {
    console.error('Printing error', error);
});
```

Notice: If not working please ensure that your device can ping the printer. And the printer must be a POSPrinter! Also ensure that you're using the correct port. 9100 is default for the thermal printers.

Print via USB (incl. listPrinters and requestPermissions)

1. First we get our printer because we don't know the printer's ID.
2. Then we request permissions for printing. This is needed because Android will not allow us to access all devices.
3. And finally we can print with our device.

```
ThermalPrinter.listPrinters({type: 'usb'}, function(printers) {
    if (printers.length > 0) {
        var printer = printers[0];
        ThermalPrinter.requestPermissions(printer, function() {
            // Permission granted - We can print!
            ThermalPrinter.printFormattedText({
                type: 'usb',
                id: printer.id,
                text: '[C]<u><font size='big'>Hello World</font></u>' //
new lines with "\n"
            }, function() {
                console.log('Successfully printed!');
            }, function(error) {
                console.error('Printing error', error);
            });
        }, function(error) {
            console.error('Permission denied - We can\'t print!');
        });
    } else {
        console.error('No printers found!');
    }
}, function(error) {
    console.error('Ups, we cant list the printers!', error);
});
```

listPrinters(data, successCallback, errorCallback)

List available printers

Param	Type	Description
data	`Object`	Data object
data.type	`"bluetooth"` | `"usb"`	Type of list: bluetooth or usb
successCallback	`function`	Result on success
errorCallback	`function`	Result on failure

printFormattedText(data, successCallback, errorCallback)

Print a formatted text and feed paper

See: https://github.com/DantSu/ESCPOS-ThermalPrinter-Android#formatted-text--syntax-guide

Param	Type	Description
data	`Array.<Object>`	Data object
data.type	`"bluetooth"` | `"tcp"` | `"usb"`	List all bluetooth or usb printers
[data.id]	`string` | `number`	ID of printer to find (Bluetooth: address, TCP: Use address + port instead, USB: deviceId)
[data.address]	`string`	If type is "tcp" then the IP Address of the printer
[data.port]	`number`	If type is "tcp" then the Port of the printer
[data.mmFeedPaper]	`numberoptional`	Millimeter distance feed paper at the end
[data.dotsFeedPaper]	`numberoptional`	Distance feed paper at the end
[data.printerDpi]	`numberoptional`	Printer DPI
[data.printerWidthMM]	`numberoptional`	Paper Width in mm
[data.printerNbrCharactersPerLine]	`numberoptional`	Number of characters per line
data.text	`string`	Formatted text to be printed
successCallback	`function`	Result on success
errorCallback	`function`	Result on failure

printFormattedTextAndCut(data, successCallback, errorCallback)

Print a formatted text, feed paper and cut the paper

See: https://github.com/DantSu/ESCPOS-ThermalPrinter-Android#formatted-text--syntax-guide

Param	Type	Description
data	`Array.<Object>`	Data object
data.type	`"bluetooth"` | `"tcp"` | `"usb"`	List all bluetooth or usb printers
[data.id]	`string` | `number`	ID of printer to find (Bluetooth: address, TCP: Use address + port instead, USB: deviceId)
[data.address]	`string`	If type is "tcp" then the IP Address of the printer

Param	Type	Description
[data.port]	`number`	If type is "tcp" then the Port of the printer
[data.mmFeedPaper]	`numberoptional`	Millimeter distance feed paper at the end
[data.dotsFeedPaper]	`numberoptional`	Distance feed paper at the end
[data.printerDpi]	`numberoptional`	Printer DPI
[data.printerWidthMM]	`numberoptional`	Paper Width in mm
[data.printerNbrCharactersPerLine]	`numberoptional`	Number of characters per line
data.text	`string`	Formatted text to be printed
successCallback	`function`	Result on success
errorCallback	`function`	Result on failure

getEncoding(data, successCallback, errorCallback)

Get the printer encoding when available

Param	Type	Description
data	`Array.<Object>`	Data object
data.type	`"bluetooth"` | `"tcp"` | `"usb"`	List all bluetooth or usb printers
[data.id]	`string` | `number`	ID of printer to find (Bluetooth: address, TCP: Use address + port instead, USB: deviceId)
[data.address]	`string`	If type is "tcp" then the IP Address of the printer
[data.port]	`number`	If type is "tcp" then the Port of the printer
successCallback	`function`	Result on success
errorCallback	`function`	Result on failure

disconnectPrinter(data, successCallback, errorCallback)

Close the connection with the printer

Param	Type	Description
data	`Array.<Object>`	Data object
data.type	`"bluetooth"` | `"tcp"` | `"usb"`	List all bluetooth or usb printers
[data.id]	`string` | `number`	ID of printer to find (Bluetooth: address, TCP: Use address + port instead, USB: deviceId)
[data.address]	`string`	If type is "tcp" then the IP Address of the printer

Param	Type	Description
[data.port]	`number`	If type is "tcp" then the Port of the printer
successCallback	`function`	Result on success
errorCallback	`function`	Result on failure

requestPermissions(data, successCallback, errorCallback)

Request permissions for USB printers

Param	Type	Description
data	`Array.<Object>`	Data object
data.type	`"bluetooth"` | `"tcp"` | `"usb"`	List all bluetooth or usb printers
[data.id]	`string` | `number`	ID of printer to find (Bluetooth: address, TCP: Use address + port instead, USB: deviceId)
[data.address]	`string`	If type is "tcp" then the IP Address of the printer
[data.port]	`number`	If type is "tcp" then the Port of the printer
successCallback	`function`	Result on success
errorCallback	`function`	Result on failure

requestBTPermissions(data, successCallback, errorCallback)

Request permissions for bluetooth

Param	Type	Description
data	`Array.<Object>`	Data object
data.type	`"bluetooth"`	List all bluetooth or usb printers
successCallback	`function`	Result on success
errorCallback	`function`	Result on failure

bitmapToHexadecimalString(data, successCallback, errorCallback)

Convert Drawable instance to a hexadecimal string of the image data

Param	Type	Description
data	`Array.<Object>`	Data object
data.type	`"bluetooth"` | `"tcp"` | `"usb"`	List all bluetooth or usb printers

Param	Type	Description
[data.id]	`string` | `number`	ID of printer to find (Bluetooth: address, TCP: Use address + port instead, USB: deviceId)
[data.address]	`string`	If type is "tcp" then the IP Address of the printer
[data.port]	`number`	If type is "tcp" then the Port of the printer
[data.mmFeedPaper]	`numberoptional`	Millimeter distance feed paper at the end
[data.dotsFeedPaper]	`numberoptional`	Distance feed paper at the end
[data.printerDpi]	`numberoptional`	Printer DPI
[data.printerWidthMM]	`numberoptional`	Paper Width in mm
data.base64	`string`	Base64 encoded picture string to convert
successCallback	`function`	Result on success
errorCallback	`function`	Result on failure

bitmapToHexadecimalStringLarge(data, successCallback, errorCallback)

Convert Drawable instance to a hexadecimal string of the large image data

Param	Type	Description
data	`Array.<Object>`	Data object
data.type	`"bluetooth"` | `"tcp"` | `"usb"`	List all bluetooth or usb printers
[data.id]	`string` | `number`	ID of printer to find (Bluetooth: address, TCP: Use address + port instead, USB: deviceId)
[data.address]	`string`	If type is "tcp" then the IP Address of the printer
[data.port]	`number`	If type is "tcp" then the Port of the printer
[data.mmFeedPaper]	`numberoptional`	Millimeter distance feed paper at the end
[data.dotsFeedPaper]	`numberoptional`	Distance feed paper at the end
[data.printerDpi]	`numberoptional`	Printer DPI
[data.printerWidthMM]	`numberoptional`	Paper Width in mm
data.base64	`string`	Base64 encoded picture string to convert
successCallback	`function`	Result on success
errorCallback	`function`	Result on failure