

Boca Swift SDK

Bluetooth Class

Overview:

The Bluetooth class provides a simple interface to communicate with devices over Bluetooth. It supports opening, closing, reading, and writing to the printer via Bluetooth.

OpenSessionBT()

Attempts to connect to the printer via classic Bluetooth.

Returns:

- `true` if the connection is successful.
- `false` otherwise.

Behavior:

- Calls the Bluetooth connection handler.
- Set the mode to `"BT"`.

OpenSessionBTPicker(string serialNum)

Attempts to connect to a Bluetooth Low Energy printer by serial number.

Parameters:

- `serialNum (string)`: The serial number of the printer.

Returns:

- `true` if a device is found and connected.
- `false` otherwise.

Behavior:

- Starts a BLE scan and connects to the matching peripheral.
 - Set the mode to "BTLE".
-

CloseSessionBT()

Closes the Bluetooth connection (classic or BLE).

Behavior:

- Call the appropriate disconnection function depending on the mode.
 - Resets mode and connection status.
-

Please find the example in the next page.

Usage Example:

```
let btManager = BluetoothManager()
```

```
// For Classic Bluetooth
```

```
if btManager.openSessionBT() {
```

```
    // write
```

```
    btManager.closeSessionBT()
```

```
}
```

```
// For BLE
```

```
if btManager.openSessionBTPicker(serialNum: "DEVICE_SERIAL") {
```

```
    // Wait for connection in delegate callbacks
```

```
    // Then write/read data
```

```
}
```

TCP/IP Class

Overview:

The TCP/IP class provides a simple interface to communicate with devices over a TCP/IP connection. It supports connecting, reading, writing, and closing the connection.

OpenSessionNetwork(string ipAddress)

Connects to a printer over the local network using a specified IP.

Parameters:

- `ipAddress (string)`: The IP address of the printer.
-

CloseSessionNetwork()

Closes the active network session.

Behavior:

- Send shutdown command to TCP stack.
 - Resets mode and connection status.
-

Please find the example in the next page.

Usage Example:

```
let printer = OpenSessionNetwork(ipAddress: "192.168.1.100")
```

```
do {
```

```
    // Send bytes
```

```
    try printer.SendBytes([0x1B, 0x40])
```

```
    // Send text
```

```
    try printer.SendString("Hello Printer!\n")
```

```
} catch {
```

```
    print("Printer error: \(error)")
```

```
    printer.CloseSessionNetwork()
```

```
}
```

Utilities Class

loadGraphic(string filePath, string fileType)

Loads a graphic file into memory without printing it.

Parameters:

- *filePath (string)*: The full path of the graphic file to be loaded.
- *fileType (string)*: The file type specifier (e.g., "bmp", "pcx", "png").

Behavior:

- Reads the binary graphic file into memory.
 - Returns the raw binary data of the image.
-

loadAndConvertGraphic(string filePath, string fileType, string convertToType)

Loads a graphic file into memory and optionally converts its format.

Parameters:

- *filePath (string)*: The full path of the graphic file to be loaded.
- *fileType (string)*: The original file type (e.g., "bmp", "pcx", "png").
- *convertToType (string)*: The target format (e.g., "pcx", "bmp").

Behavior:

- Reads the binary graphic file into memory.
 - Converts the image to the specified format (if supported).
 - Returns the converted binary data.
-

DownloadGraphicToMemory(string filePath, string fileType)

Loads a graphic file into the printer's memory without immediate printing.

Parameters:

- *filePath* (string): The full path of the graphic file to be loaded
- *fileType* (string): The file type specifier (e.g., "bmp", "pcx", "png")

Behavior:

- Reads the binary graphic file from the specified path
- Transfers the graphic data to the printer's internal memory
- Does not initiate printing
- Returns true if the download was successful

Example - Downloading a Graphic:

```
let success = DownloadGraphicToMemory("logo.pcx", "pcx");  
  
if (success) {  
    // Graphic is now stored in printer memory  
}
```

PrintLogo(int idNum, int row, int column)

Sends a command to print a logo stored in memory at a specified position.

Parameters:

- `idNum (int)`: The ID number of the logo.
- `row (int)`: The vertical position on the ticket.
- `column (int)`: The horizontal position on the ticket.

Behavior:

- Constructs the FGL command `<SP{row}, {column}><LD{idNum}>`.
- Send the command to the printer.

Returns:

- `true` if the command is sent successfully.

Example – Printing a Stored Logo:

```
let printer = BocalpadSDK()
```

```
printer.printLogo(3, 10, 10)
```

SendString(string text)

Sends a plain text string to the printer using the current communication mode.

Parameters:

- `text (string)`: The text to send.

Behavior:

- Sends via the active mode: TCP/IP or Bluetooth.

Example:

```
printer.sendString("Hello, Boca Printer")
```

SendBytes(byte[] data)

Sends a raw byte array to the printer using the current communication mode.

Parameters:

- `data (byte[])`: The array of bytes to send to the printer.

Behavior:

- Sends the byte stream over the active communication mode: TCP/IP or Bluetooth.
- Intended for sending binary commands or graphic data.

Example:

```
let rawCommand: [UInt8] = [0x1B, 0x63]
```

```
printer.sendBytes(rawCommand)
```

PrintCut()

Send the cut command to the printer.

Behavior:

- Send the command `<p>` which cuts the paper after printing.
-

PrintNoCut()

Sends the no-cut eject command to the printer.

Behavior:

- Send the command `<q>` which ejects the paper without cutting.
-

SetStartPosition(int row, int column)

Sets the starting cursor position for printing.

Parameters:

- `row (int)`: Row position.
- `column (int)`: Column position.

Behavior:

- Updates the internal tracking position for text or graphic placement.
-

ChangeConfiguration(string path, string orientation)

Updates the printer configuration settings.

Parameters:

- `path (string)`: The printer path (e.g., "<P1>").
- `orientation (string)`: Orientation string ("<LM>" for landscape or "<PM>" for portrait).

Behavior:

- Updates the printer's internal path and orientation.
-

ClearMemory()

Clears the printer's graphic memory.

Behavior:

- Sends the command `\x1Bc` to clear downloaded logos or image memory.
-

ReadStatusAsync()

Asynchronously reads printer status bytes from all available communication interfaces.

Returns:

- A `byte[]` (Swift `[UInt8]`) array containing the status response from the printer.
- Additionally

Behavior:

- Check the currently active communication mode (TCP/IP or Bluetooth).

Example – Reading Status Bytes:

```
let statusBytes = try await printer.readStatusAsync()
```

GetStatusMeaning(byte data)

Interprets printer status code bytes.

Parameters:

- `data (byte)`: The raw status byte.

Returns:

- A string representing the meaning of the status code.

Behavior:

- Recognizes common status bytes (e.g., Ticket ACK, Ticket NAK, Cutter Jam, etc.).
-

Status Codes

The following status codes are supported:

Byte Value	Meaning
0x06	Ticket ACK
0x08	Invalid Checksum
0x09	Valid Checksum
0x10	Out of Tickets
0x11	X-On
0x12	Power On
0x13	X-Off
0x15	Ticket NAK
0x18	Ticket Jam
0x1D	Cutter Jam