EPSON®
EXCEED YOUR VISION

ePOS-fiscal.device

Development Guide

REVISION HISTORY

0	All	Preliminary.	September 2014	F.Chiechi
A	All	First release	November, 17th 2014	C.Radaelli

Cautions

• No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Seiko Epson Corporation.
• The contents of this document are subject to change without notice. Please contact us for the latest information.
• While every precaution has taken in the preparation of this document, Seiko Epson Corporation assumes no responsibility for errors or omissions.
• Neither is any liability assumed for damages resulting from the use of the information contained herein.
• Neither Seiko Epson Corporation nor its affiliates shall be liable to the purchaser of this product or third parties for damages, losses, costs, or expenses incurred by the purchaser or third parties as a result of: accident, misuse, or abuse of this product or unauthorized modifications, repairs, or alterations to this product, or (excluding the U.S.) failure to strictly comply with Seiko Epson Corporation's operating and maintenance instructions.
• Seiko Epson Corporation shall not be liable against any damages or problems arising from the use of any options or any consumable products other than those designated as Original EPSON Products or EPSON Approved Products by Seiko Epson Corporation.

Trademarks

EPSON® and ESC/POS® are registered trademarks of Seiko Epson Corporation in the U.S. and other countries.

Windows® and Internet Explorer® are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

Safari™ and TrueType® are either registered trademarks or trademarks of Apple Inc. in the United States and other countries.

Android™ and Google Chrome™ are either registered trademarks or trademarks of Google Inc. in the United States and other countries.

Mozilla® and Firefox® are either registered trademarks or trademarks of Mozilla Foundation in the United States and other countries.

IOS® is registered trademarks or trademarks of Cisco in the United States and other countries.

ESC/POS® Command System

EPSON has been taking industry's initiatives with its own POS printer command system (ESC/POS).

ESC/POS has a large number of commands including patented ones. Its high scalability enables users to build versatile POS systems. The system is compatible with all types of EPSON POS printers (excluding the TM-C100) and displays. Moreover, its flexibility makes it easy to upgrade the future. The functionality and the user-friendliness is valued around the world.

Index

1 Overview

1.1 Description

The aim of this manual is to provide information to software designers who wish to make use of the ePOS fiscal device in order to manage HID (Human Interface Device) like scanner or keyboard

1.2 List of Related Documents

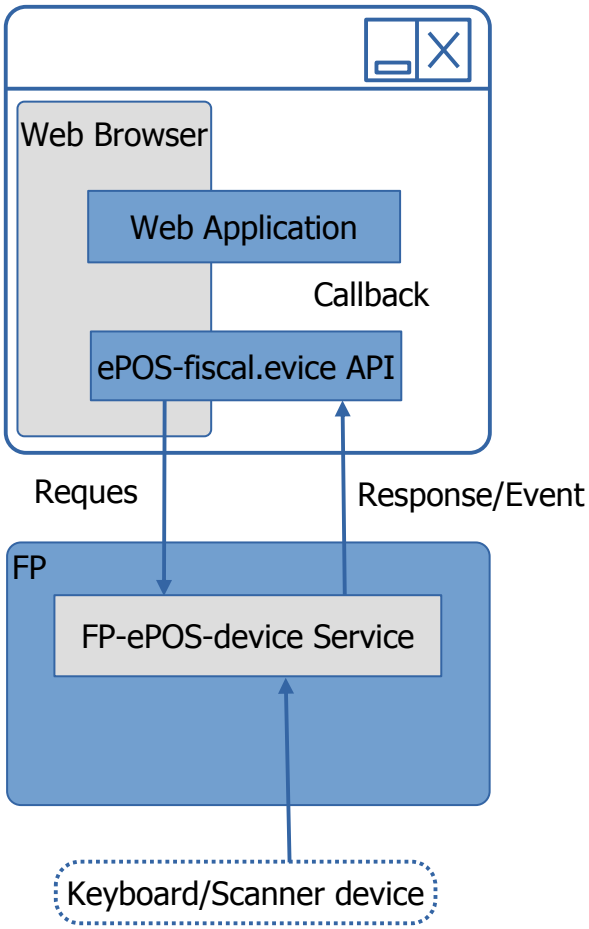
TBD

1.3 Abbreviations

- FP: Fiscal Printer

2 ePOS-fiscal.device API

This chapter describes the features and the specifications for ePOSfiscal-Device API



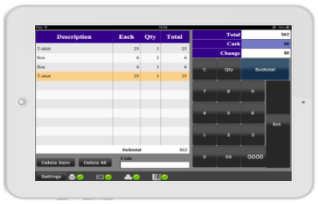
ePOS-fiscal.device API is a function used to control the peripheral devices connected to an FP in a multi-platform environment. Using ePOS-fiscal.device API, you can directly control the peripheral devices from Web browsers on personal computers, smart phones, or tablet computer.

2.1 Features

- Available from anywhere when used with a terminal with a Web browser installed in a network environment.
- No need to prepare any device (such as a PC) to act as a controller to control the peripheral devices.
- Devices that run with the OS-standard driver can be used with a device control script without any driver installed. As the device control script is written in JavaScript, it can be developed using the same language as the Web application.
- Accessing a device using FP-ePOS-Device API automatically locks the device exclusively. Even if accessed from multiple terminals simultaneously, the device is not under multiple controls. When the terminal that has controlled the device releases the device, the device becomes controllable from another terminal.

2.2 System Configuration Example

System to Use FP as Application Server



- Place a Web application in the server.
- Display the Web application with a Web browser.
- The Web browser sends requests to FP.
- The devices connected to FP send messages to the FP.
- FP returns messages to the Web browser.

2.3 Operating Enviroment

Web Browser on Terminal: Google Chrome 37 or later
Fiscal Printer model: FP90III or FP81II ETH from FW version 4.01
Peripheral Device: Input device tha can run with an OS standard HID driver

3 ePOS-fiscal.device API

This chapter describes how to write programs using FP-ePOS-Device.

3.1 Embedding the ePOS-fiscal.device API

The ePOS-fiscal.device API is provided so that ePOS-fiscal.device can be used from the JavaScript on the client side.

It is provided as JavaScript, and its file name is "fp_eposdevice-*.js".

The ePOS-fiscal.Device API is used by embedding fp_eposdevice-*.js into applications.

Embedding into Web pages

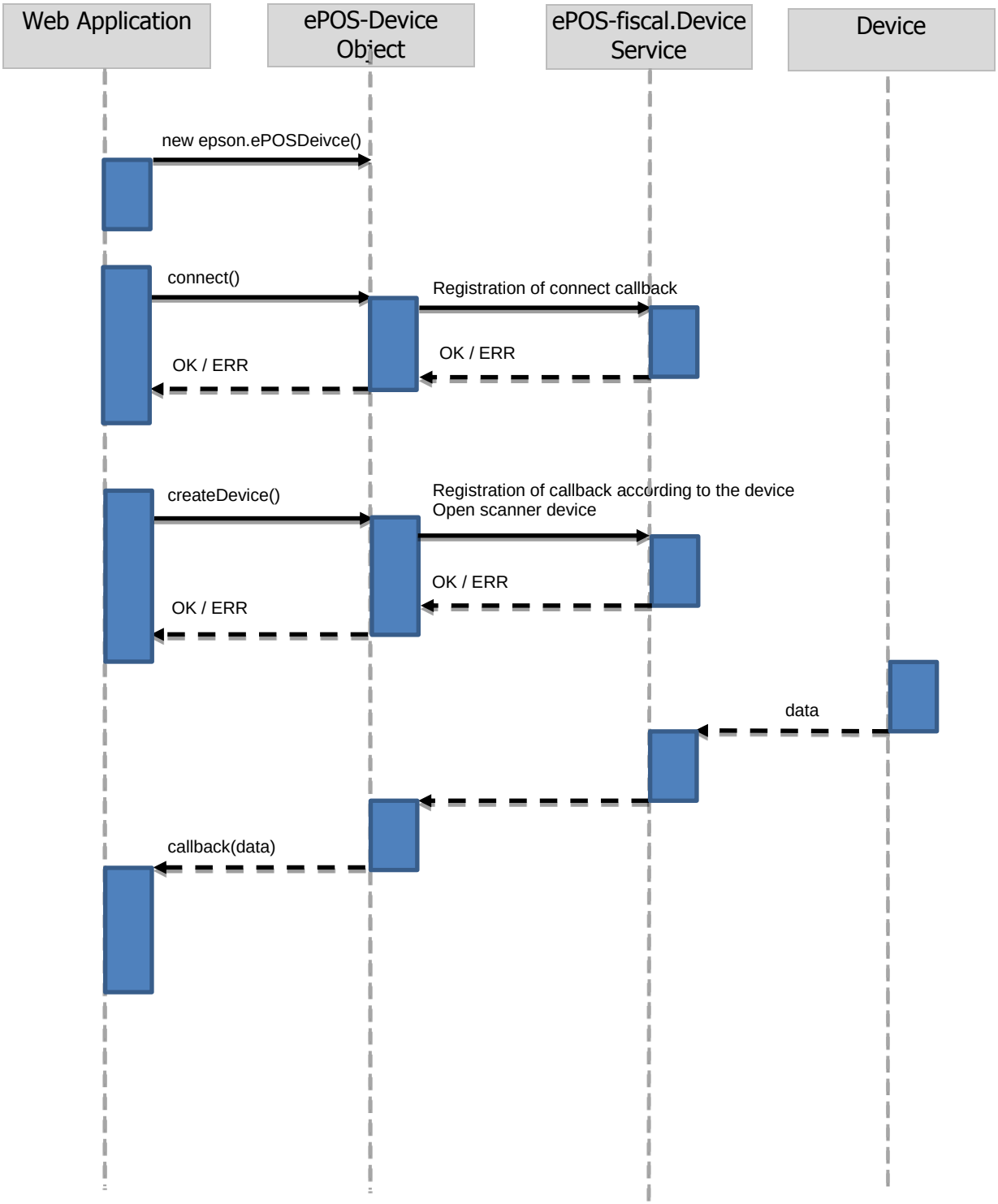
Embed the script into the Web page by using the HTML <script> tags.

At this stage a javascript json library is also needed. A realistic example is then the following:

```
<script type="text/javascript" src="fp_eposdevice-1.0.0.js"></script>
<script type="text/javascript" src="json2.js"></script>
```

3.2 Operating Enviroment

The basic programming sequence of FP-ePOS-Device API is as follows:



3.3 Operating Enviroment

FP-ePOSDevice object (window.epson.fp-ePOSDevice)

HID object

3.4 ePOS-fiscal.device Object

3.4.1 Constructor

Constructor for an FP-ePOS-Device object. Creates a new FP-ePOS-Device object and initializes it.

Syntax

ePOSDevice();

Return value

- FP-ePOS-Device object

3.4.2 Connect method

Establishes a communication path to FP-ePOS-Device Service I/F of FP.

Syntax

connect(ipAddress, port, callback);

Parameters

1 ipAddress :Object type : (String)

Specifies the IP address of TM-DT.

3 port :Object type : (Number)

Specifies "8008" (port number of FP-ePOS-Device Service I/F).

4 callback :Object type : (Function)

Specifies the callback to receive the processing results. To the first parameter of the callback, the following string is returned:

"OK" Connected successfully.

"ERR" Connection error.

3.4.3 Disconnect method

Disconnects the communication path connected with connect method.

Syntax

disconnect();

3.4.4 isConnect method

Obtains status of the communication path connected with connect method.

Syntax

isConnected();

Return value

- The connection status of the communication path

  true Connected

  false Not connected

3.4.5 createDevice method

Obtains a device object to act as the interface with the device.

When this method is successfully executed, the specified device is exclusively locked and calling "createDevice" from a different browser returns "DEVICE_IN_USE". The printer is not locked exclusively and can obtain device objects with "createDevice" from multiple browsers. If "createDevice" is executed immediately during a return process after an ondisconnect event, "DEVICE_IN_USE" occurs. In this case, retry until a value other than "DEVICE_IN_USE" is obtained.

Syntax

createDevice(deviceId, deviceType, options, callback);

Parameters

- deviceId :Object type: (String)

  Specifies the device ID which is a user defined string that is indicated from now on as <device_id>

- deviceType :Object type : (String)

  Specifies the following device type:

  type_scanner Specifies the scanner for the device type.

4 Options :Object type : (Object)

  Not used (specify false)

- callback :Object type : (Function)

  Specifies the callback to receive the processing results. When the second parameter if the callback is "OK", the device object is returned into the first parameter.
  The following string is returned into the second parameter:

  "OK" Obtained the device object successfully.

  "ERR" Error obtaining the device object.

3.4.6 deleteDevice method

Syntax

deleteDevice(deviceObject, callback);

Parameters

3 deviceObject :Object type : (Object)

Specifies the device object.

- callback :Object type: (Function)

Specifies the callback to receive the processing results.

The following string is returned into the parameter of the callback:

"OK" Device close succeeded.

"ERR" Error closing the device object.

3.5 Keyboard Object/Scanner Object

3.5.1 Ondata event

Syntax

Function(data);

Parameters

- data :Object type : (Object)

is a JSON object that contains info about one key press at a time with the following format:

```
["device_data", "1", "<device_id>", {
    "type":    "ondata",
    "data1":   0,
    "data2":   0,
    "data3":   0,
    "data4":   6
}]
```

the meaning of data is:

- data1 is normally 0

- data2 is 1 or 0 if shift is pressed or not respectively

- data3 and data4 contain the scancode which must be mapped to input data (see chapeter Websocket Protocol for an example)

4 Web Socket protocol

4.1 Connect to printer over websocket

use url:

ws://<ipAdress>:8008/connect

4.2 Open scanner device

To open the scanner device connected to the printer send following JSON text:

```
["open_device", "<device_id>", {
        "type":         "type_scanner",
        "crypto":       false
        "session_id": "<application_defined_string>"
    }]
```

session_id is an extension and is set to a default if not specified

Receive Answer (positive):

```
["open_device", "<device_id>", "OK", {
        "session_id": <application_defined_string>
    }]
```

Receive answer (negative), for example if another terminal is already connected:

```
["open_device", "<device_id>", "ERR"", {
        "session_id": <application_defined_string>
    }]
```

4.3 Receive scanner data

This can be done in a "ondata" callback.

The data has the following format:

```
["device_data", "1", "<device_id>", {
        "type":         "ondata",
        "data1":        0,
        "data2":        0,
        "data3":        0,
        "data4":        6
    }]
```

and must be **replied** by an acknowledge with the following format:

["ack", "dummy"]

the meaning of data is:

- data1 is normally 0

- data2 is 1 or 0 if shift is pressed or not respectively

- data3 and data4 contain the scancode which must be mapped in tables like the following:

```
var TNT = 999;
var TCL = 12;
var TCONT = 0x0d;
var TENTER = TCONT;
```

In case of **device disconnection** a message is sent with the following format:

```
["device_data", "1", "<device_id>", {
                "type":         "onerror",
                "data1":        0,
                "data2":        0,
                "data3":        0,
                "data4":        0
        }]
```

and must be replied by an acknowledge with the following format:

```
["ack", "dummy"]
```

4.4 Conversion tablet from scan code to ascii

```
var tab_scan_code_lower_case = [
TNT, TNT, '1', '2', '3', '4', '5', '6', '7', '8', // 00
'9', '0', '\'', TNT, TNT, TNT, 'q', 'w', 'e', 'r', // 10
't', 'y', 'u', 'i', 'o', 'p', TNT, '*', TENTER, TNT, // 20
'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', '@', // 30
TNT, TNT, TNT, '>', 'z', 'x', 'c', 'v', 'b', 'n', // 40
'm', ';', '.', '-', TNT, '*', TNT, ' ', TNT, TNT, // 50
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, // 60
TNT, '7', '8', '9', '-', '4', '5', '6', '+', '1', // 70
'2', '3', '0', '.', TNT, TNT, TNT, TNT, TNT, TNT, // 80
TNT, TNT, TNT, TNT, TNT, TNT, TENTER, TNT, TNT, TNT, // 90
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //100
TNT, TCL, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //110
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //120
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //130
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //140
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //150
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //160
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //170
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //180
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //190
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //200
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //210
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //220
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //230
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //240
TNT, TNT, TNT, TNT, TNT, TNT, //250
];
```

var tab_scan_code_upper_case = [
TNT, TNT, '!', '"', '#', '$', '%', '&', '/', '(', // 00
')', '=', '?', TNT, TNT, TNT, 'Q', 'W', 'E', 'R', // 10
'T', 'Y', 'U', 'I', 'O', 'P', TNT, '+', TENTER, TNT, // 20
'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', '@', // 30
TNT, TNT, TNT, '<', 'Z', 'X', 'C', 'V', 'B', 'N', // 40
'M', ',', ':', '_', TNT, '*', TNT, ' ', TNT, TNT, // 50
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, // 60
TNT, '7', '8', '9', '-', '4', '5', '6', '+', '1', // 70
'2', '3', '0', '.', TNT, TNT, TNT, TNT, TNT, TNT, // 80
TNT, TNT, TNT, TNT, TNT, TNT, TENTER, TNT, TNT, TNT, // 90
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //100
TNT, TCL, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //110
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //120
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //130
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //140
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //150
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //160
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //170
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //180
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //190
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //200
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //210
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //220
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //230
TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, TNT, //240
TNT, TNT, TNT, TNT, TNT, TNT, //250
];

4.5 Close scanner device

to close the scanner device connected to the printer send following JSON text:

["close_device", "<device_id>", {}]

receive answer (positive):

["close_device", "<device_id>", "OK"]

receive answer (negative):

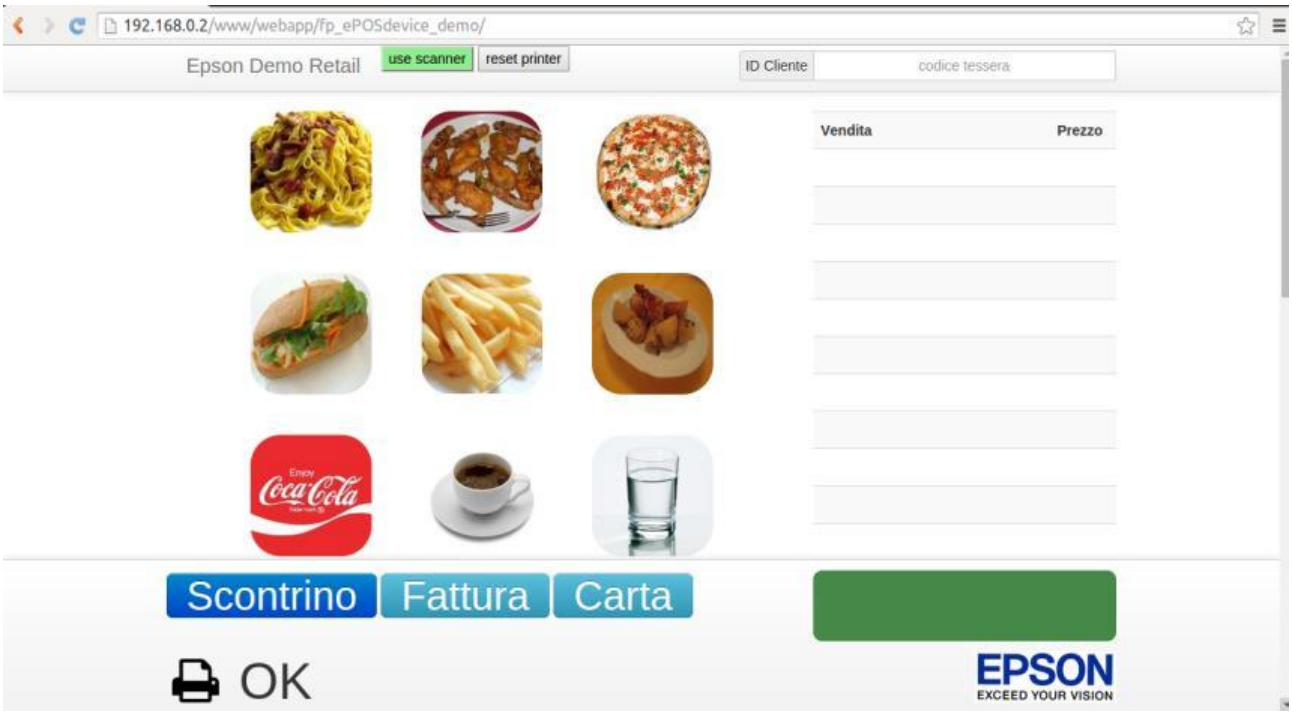
["close_device", "<device_id>", "ERR"]

5 Sample program

This chapter describes how to use the sample program.

5.1 Overview

The sample program allows you to test the FP-ePOS-device functionality by simulating a food shop.



5.2 Devices required

(Note that you do not have to make all of the following devices available to use the sample program.)

- Fiscal Printer (FP90-III ....)
- Customer display
- POS keyboard
- Barcode scanner

5.3 Other

- LAN cable
- Sample program
- HTML5-supported Web browser on Terminal Device (PC Tablet or SmartPhone)

5.4 Enviroment Setting

- Power ON the Printer and connect a LAN cable
  Printer display is required for proper FP functioning, also connect the usb keyboard to configure the printer
- Setup the FP for LAN communication in the same subnet of the Terminal Device
  (see EPSON_SerieFP_Manuale_operatore, section 9.5 "Programmazione LAN")
- Setup the FP to enable Web Application
  (see EPSON_SerieFP_Manuale_operatore, 9.32 "Programmazione Parametri Intelligent")
  NOTE: after this step it is necessary to power OFF and power ON the printer
- Enable web socket
  (to update in EPSON_SerieFP_Manuale_operatore)
- Upload the sample program
  - in the URL address bar of a Browser window type
    <FP ip address>/cgi-bin/upload.cgi
  - choose the provided file fp_ePOSdevice_demo.zip an upload it

5.5 Starting and Using sample program

1. Connect a Barcode scanner to the FP
2. In the URL address bar of a Browser window type
   <FP ip address>/www/webapp/fp_ePOSdevice_demo/
3. Use the scanner with the products table provided in the next paragraph

5.6 Enviroment Setting

)pri

)pat

)sec

)con

)piz

)coc

)pan

)caf

)acq