



# TREMOL Ltd.

BULGARIA  
5000 Veliko Tarnovo, 6 Toledo str.  
tel.: +359 62 600 844, fax: +359 62 604 208  
website: [www.tremol.bg](http://www.tremol.bg)

## Tremol fiscal device libraries and file server for main programming languages and platforms

### 1. Overview

#### 1.1. The Solution

The bundle provides support for accelerated development cycle of creating end-user software interacting with **TREMOL** fiscal devices.

It is a result of an automated process, generating the required libraries, tools and documentation from **latest official technical specifications**.

The package hides the complexities and internals of the communication protocol in the customer software. The lower-level implementation details, the underlying communication channels and platforms are abstracted away, instead higher level API is exposed to the end user software.

❑ **This grants the customer software the ability to:**

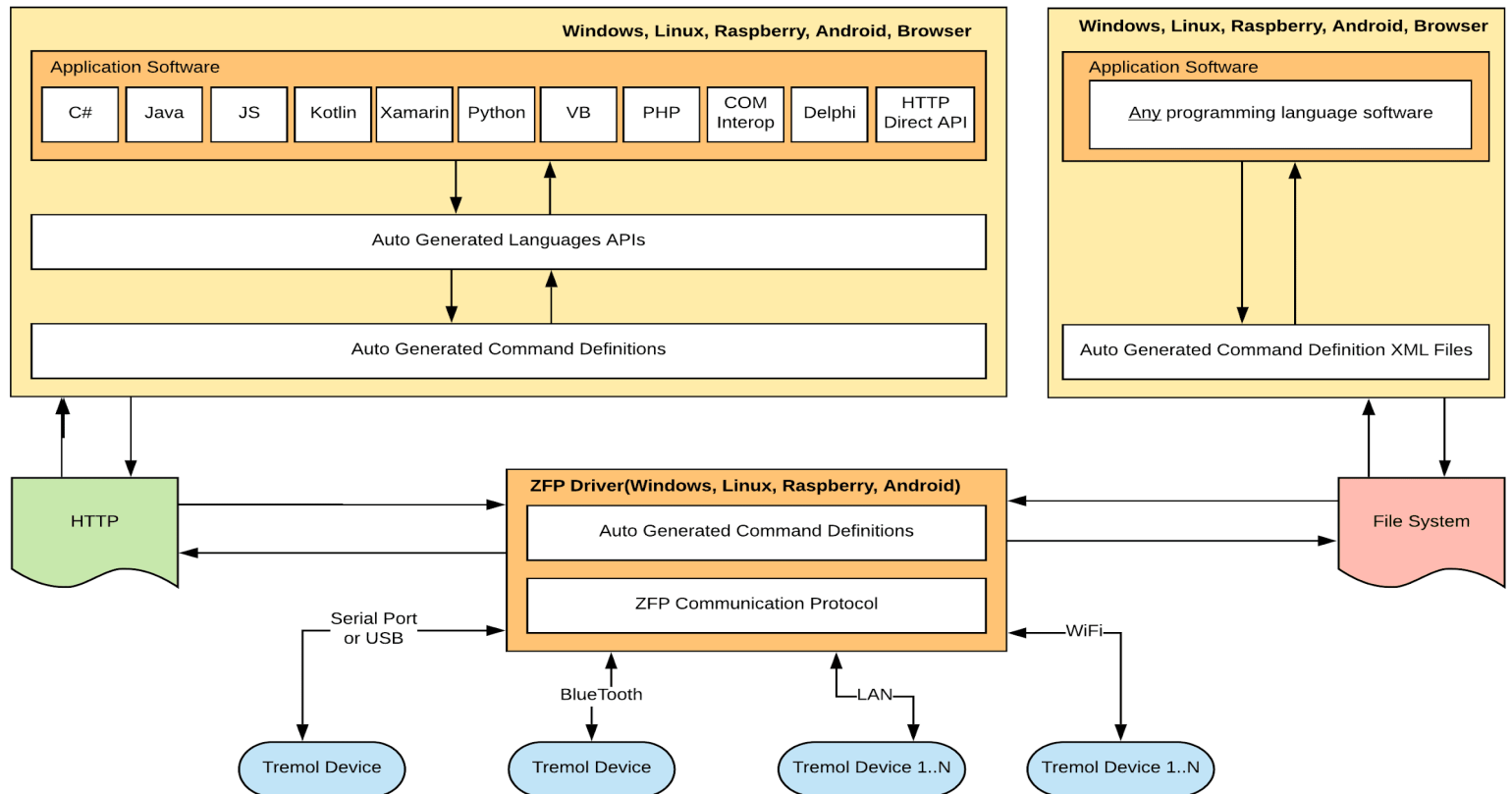
- Run on various platforms: Windows, Android, Raspberry, Linux, Web browser.
- Communicate with the fiscal device over Serial port, Bluetooth, LAN, WIFI.
- Control multiple fiscal devices from one workstation and/or share one fiscal device between multiple workstations.
- Use comprehensive and documented APIs for the available languages.
- Get support, updates and enhancements.
- Use the latest implementation of protocol specification.

❑ **The bundle contains:**

- Fully documented commands required for the operation of the fiscal device, implemented in various programming languages.
- Driver performing the low level communication with the fiscal device.
- Demo applications and tools demonstrating the usage of common commands.

## 1.2. Software Architecture

Interaction between applications, libraries, server (driver) and the devices:



- ❑ Client applications send commands to local or remote server through the API.
  - Communication is done over HTTP or File System.
  - The transmitted data is structured according to the definitions.
  - APIs for sending commands or files are auto-generated according to the definitions.
  - Command's request and response are fully documented.
  - Possibility to control multiple fiscal devices in the local network.
- ❑ The driver communicates with the available fiscal devices.
  - Uses the same definitions as the clients.
  - Converts the commands to ZFP protocol.
  - Establishes the connection with Fiscal Device over the requested transport channel.
  - Executes the commands and returns the response or error code.

## 2. Quick Start

### 2.1. Installation

Install [ZFPLabServerSetup.exe](#) from the package. After installation is completed, the server starts automatically at system boot. Other options are to redistribute server files with your software or install it as a service.

For Linux/Raspberry installation see [3.2.2](#). For Android installation see [3.2.3](#).

### 2.2. Using **Libraries**

Add to your project related libraries from **Libs** folder, according to the programming language. Example usage of the C# source code:

```
private void CreateReceipt()
{
    try
    {
        FP fp = new FP() {ServerAddress = "http://localhost:4444/"};
        fp.ServerSetDeviceSerialPortSettings("COM5", 115200);
        fp.OpenReceipt(1, "0000", OptionFiscalReceiptPrintType.Step_by_step_printing);
        fp.SellPLUwithSpecifiedVAT("Tomato", OptionVATClass.VAT_class_A, 1.29m, 3.21m);
        fp.CashPayCloseReceipt();
    }
    catch (SEException ex)
    {
        HandleException(ex);
    }
}
```

### 2.3. Using **File Server**

- Prepare commands for execution from **Libs/FileServer** by filling desired argument values or use prepared examples from **Demos/FileServer**
- Use "\_Settings.xml" to set up the connection
- Execute them by copying to "FILE\_IN" folder
- Check the result in [FILE\\_OUT](#) folder

#### Notes:

- More details can be seen in the provided Demos
- Detailed information about using the libraries can be found in [4.Libraries](#)
- Details about configuration of File Server, usage and examples can be found in [5. File server](#)

## 3. Package Content

### 3.1. Protocol\_description\_XX\_yyMMddHHmm.pdf

The document is fully-described communication protocol of *TREMOL* fiscal devices, working in fiscal printer mode. In the document's name are encoded:

- The Country - XX
- The timestamp of the last revision (VERSION) of the protocol. It is the same as the value of VersionDefinitions field, property or method in the generated libraries.
- (yy - year, MM - month, dd - day, HH - hour, mm -minute)

### 3.2. ZFPLabServer

Server (driver), which translates and executes the requests from Demo or Customer app to the fiscal device, and translates back the responses from the fiscal device to the app. The translation is based on server definitions, which are also auto-generated by the protocol. By default it is working on TCP port 4444. One server can serve multiple clients and devices. The communication with the device can be done by Serial, Bluetooth or TCP port.

- Although the various language APIs include all supported commands, they can be viewed and directly executed at <http://localhost:4444> .
- Log can be obtained from the Log tab (Windows with GUI) or at <http://localhost:4444/log>
- Only a single instance of the server can run in the system on the same port. When a new instance starts, **it shuts down the previous one.**
- To change the default server port an additional file *HttpServerSettings.xml* should be created in the ZfpLabServer executable directory with the following content:

```
<HttpServerSettings>
  <Port> 4445 </Port>
</HttpServerSettings>
```

#### 3.2.1 ZFPLabServer (Windows)

Possible usage scenarios:

- **Installation as a Windows Service.**  
From version 1.5.7 onwards the server supports running as a service.  
This kind of installation is suitable for configurations where the server is running on dedicated server machine and is shared between PCs. No user login in the system is required for normal server operation.
- **Integrating server into existing applications.** The server binaries can be redistributed with the application files and **no additional installation will be required.** When the application starts, it needs to start the server, no explicit shutdown is necessary.
- **Normal installation.** After OS boot - the server hides in the PC system tray (when used with GUI). Usually the GUI is not required, but can be used to view the communication log between the apps and the devices, examine problems and set some global settings.

The server supports **headless mode** - without GUI. This mode is the default for Linux based systems. The mode is optional for Windows and can be turned on by starting the server with argument 'nogui' from the command prompt e.g.:

*C:\Program Files\ZFPLabServer>ZFPLabServer nogui*

or creating an empty file "NOGUI.txt" in the same directory as the executable.

### 3.2.2 ZFPLabServer (Linux)

Linux driver, sharing the same functionality like the Windows version. The folder contains the executable and the definitions. The files **zfplabserver**, **zfplabserver\_x86** and **zfplabserver\_arm** from ZFPLABServerLinux folder are executables respectively for x64, x86 and ARM platforms. They have **no other dependencies** and are self contained. They may be configured for auto-start in the Linux machine. The driver runs without GUI.

#### ❑ Installation

- The file should be marked as executable on the target Distribution OS:  
*sudo chmod u+x zfplabserver*
- In case the communication is not done over LAN or WIFI, setup serial port (USB) settings:

- To get the serial ports and change permissions use:

*dmesg | grep -i tty*

*sudo chmod 0777 /dev/ttyACM0*

- In some cases it may be required to change the following settings of serial/usb port and baud rate for the serial port:

*stty -F /dev/ttyS0 -inlcr 115200*

*stty -F /dev/ttyS0 -opost -onlcr 115200*

*stty -F /dev/ttyS0 -isig -icanon -iexten -echo -echoe -echok -echoctl -echoke 115200*

- Start the executable. It expects the XML folder with the definitions to be in the same directory.

- (Optional) To run the server in background without interruptions, it can be started in **nohup** (No Hangups) mode: "**nohup ./zfplabserver &**"

**Nohup** mode can also log the full server activity in a chosen file by changing the output stream of the server to that file:

*nohup ./zfplabserver >log.txt &*

### 3.2.3 ZFPLabServer (Android)

Android application which shares the same functionality as it's Windows and Linux versions. The server works on Android version 4.1 (Jelly Bean) up to the latest and has features like:

- **Working as a background service.** The server does not need any interactions
- **Automatic service startup** after device reboot, shutdown, RAM clear or application stop
- **Automatic Bluetooth socket release** after 60 seconds of device inactivity
- **Automatic reconnection** to the fiscal device after device wakeup, Bluetooth/Wifi restart or any other case of a broken connection
- **Integrated Log** in the server UI in case of diagnostic needs
- **Integrated API** for direct commands to the fiscal device

#### 3.2.3.1 Installation

- ZFPLABServerAndroid folder contains the APK file. It can be:
  - **transferred** to the Android device(via USB, Bluetooth, cloud, etc.) and then installed.
  - **included as a resource in the client application and automatically installed** on the first run. For questions, details or code example contact [zfplab@tremol.bg](mailto:zfplab@tremol.bg) or check ZfpLabServerManager.java in the provided demo.
- During installation, the OS asks for the option “Install app from Unknown Source”. It should be accepted.
- On the initial run, the server sets up the required permissions and continues to work as a service. No further interaction with it is required.

**Note:** Some mobile device manufacturers integrate their own battery optimizations in the OS that can abruptly stops the service. Although the service restarts itself, this could cause communication problems. In these rare situations disabling the battery optimization allows the service to run normally.

### 3.2.3.2 Setup and connection

#### ❑ Bluetooth connection

- Establishing a connection to the device requires both devices to be paired
- After successful pairing, the fiscal device's Bluetooth name should be set as:

***"fp.ServerSetAndroidBluetoothDeviceSettings(deviceName="ZK123456")"***

or

***"fp.ServerSetDeviceSerialPortSettings(Com="ZK123456",Baud=115200)"***

#### ❑ WiFi Connection

- Both devices should be connected to the same WiFi network
- IP Address, Port and the ZFP Password of the fiscal device should be set
- Use settings command: ***fp.ServerSetDeviceTcpSettings()***

#### ❑ (Optional) Remote access

- In order to connect to the server from a different device, both devices (the device with the server and the intended device) should be present in the same network (WiFi)
- To access the server remotely, the IP Address of the server device is needed followed by the port, e.g.: <http://192.168.1.1:4444>

## 3.3. Libs

The folder contains auto-generated libraries by the protocol for common programming languages. The libraries provide the developers easy access to fiscal device functions and save them the effort to read and understand the protocol.

More details about supported languages can be found in [4. Libraries](#)

### 3.3.1 FileServerCommands

The folder contains all available, supported protocol-based commands in xml format, which can be executed separately or combined in one file by ZfpLabServer, after filling in the values of command parameters if required.

### 3.4. Demos

The folder contains demo applications in different programming languages, based on the auto-generated libraries. **The demos demonstrate the usage of the most common fiscal device operations and intercepting the exceptions** which can be caused by the device or the ZFPLabServer.

The folder also contains **file server examples**, describing some common fiscal receipts scenarios.

### 3.5. TOOLS

#### 3.5.1 ZFPLabSetup.exe

Standalone tool, which **maintains the overall functionality** of the protocol and provides the user access to all commands and possibility to execute, test and examine them.

#### 3.5.2 Report

Report tool, which can be used for reporting problems, bugs, questions or to. The tool collects automatically information about Windows, .Net framework, ZFPLabServer, Definition version, Communication Log and sends an email to [zfplab@tremol.bg](mailto:zfplab@tremol.bg).

### 3.6 ChangeLog.txt

The file contains the changes in the communication protocol, grouped by package version. These changes are also listed in VERSIONS section of **Protocol\_description.pdf**



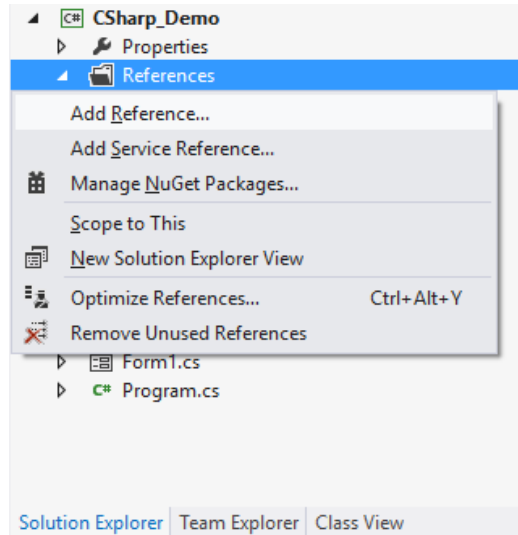
## 4. Libraries

### 4.1. Common usage

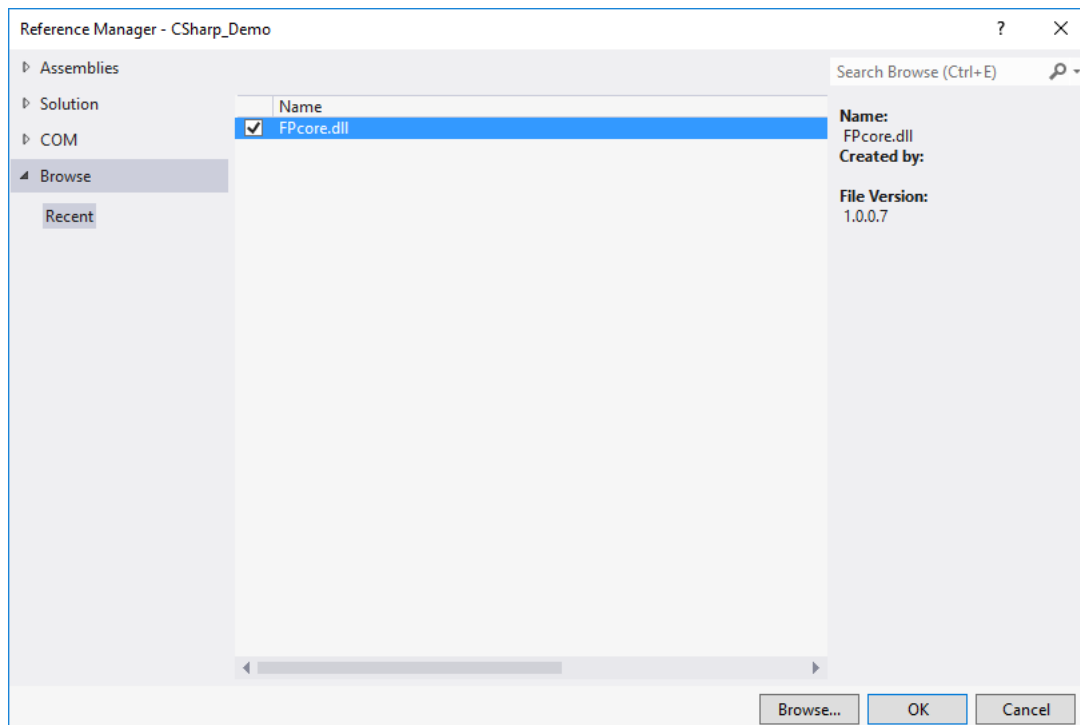
- The developer needs the specific library and the server to be installed to start developing.
- The commands should be used observing the following rules:
  - Do not send a subsequent command prior to receiving a response of the preceding one.
  - Observe the codes of Server error object for detailed error description.
  - When the received information is insufficient, request detailed status information – *ReadStatus()*
- Common basic usage of the libraries is as follows.
  - Setting of connection:
    - Server HTTP local or remote address - *ServerAddress*
    - Optional: automatically find fiscal device on serial port - *ServerFindDevice()*
    - Device TCP/Serial Connection - *ServerSetDeviceTcpSettings/ServerSetDeviceSerialPortSettings*  
It is enough to set settings only once - on application start
  - Sample sequence of commands for issuing a receipt:
    - Fiscal receipt opening – *OpenReceipt()*
    - Sale registrations – *SellPLUwithSpecifiedVAT() / SellPLUfromDep()*
    - Subtotal amount – *Subtotal()*
    - Payment – *Payment()*
    - Fiscal receipt closure - *CloseFiscReceipt()*
  - Performing error checks when executing the commands.
  - The commands are executed synchronously, each of them blocking until the fiscal device performs the operation or returns response data.

## 4.2. C# / Xamarin

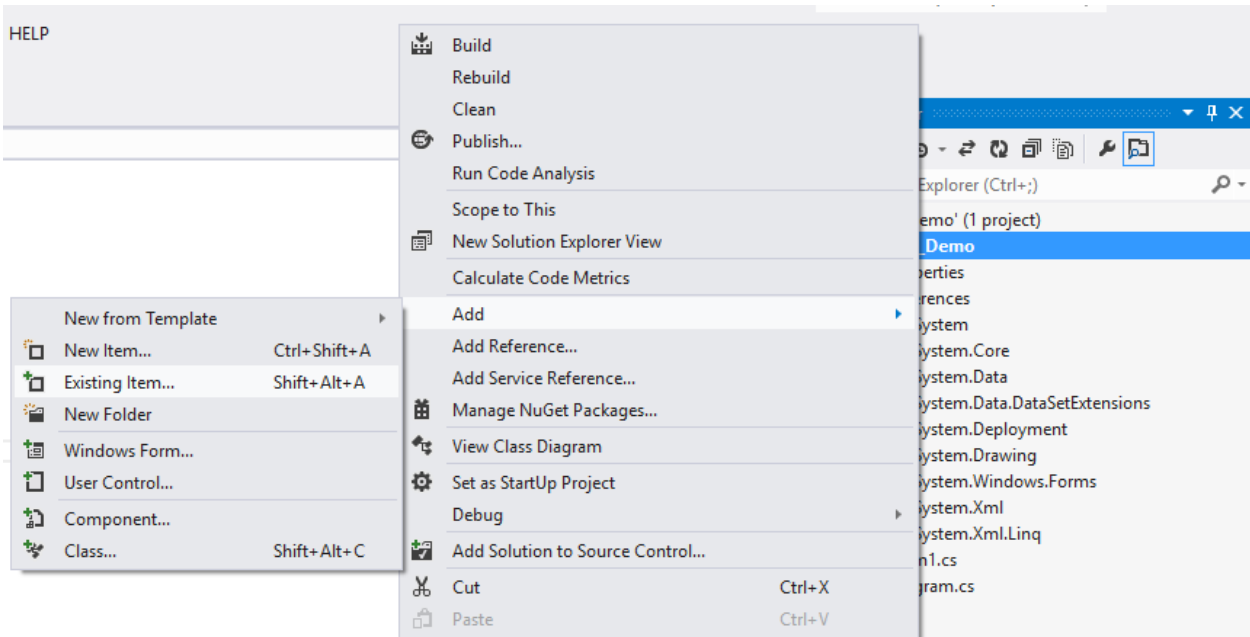
Create new C# project, go to **Solution Explorer** in MS Visual Studio, right click on **References** and select **Add Reference**:



Browse Libs\C# folder and select FPcore.dll:



Right click on project name (e.g. CSharp\_Demo), select **Add/Existing Item** from **Libs\C#** folder to add **FP.cs** file:



**Note:**

- **FP.cs file is generated by a tool. Changes to this file may cause incorrect behavior and will be lost, if the code is regenerated.**

Example C# code snippet with receipt related commands:

```
private void CreateReceipt()
{
    try
    {
        //Set server address
        FP fp = new FP() { ServerAddress = "http://localhost:4444/" };

        //Set device serial port settings
        fp.ServerSetDeviceSerialPortSettings("COM5", 115200);

        //Open receipt with Operator Number 1, Password 0000
        fp.OpenReceipt(1, "0000", OptionFiscalReceiptPrintType.Step_by_step_printing);

        void FP.OpenReceipt(decimal operNum, string operPass, OptionFiscalReceiptPrintType optionFiscalReceiptPrintType)
        Opens a fiscal receipt assigned to the specified operator and print type depends of FiscalReceiptPrintType parameter.

        //Sale article
        fp.SellPLUwithSpecifiedVAT("Tomato", OptionVATClass.VAT_Class_A, 1.29m, 3.21m, null,
            null, null, null, "", "");

        //Subtotal
        fp.Subtotal(OptionPrinting.Yes, OptionDisplay.Yes, null, null);

        //Payment
        fp.Payment(OptionPaymentType.Payment_0, 4.15m);

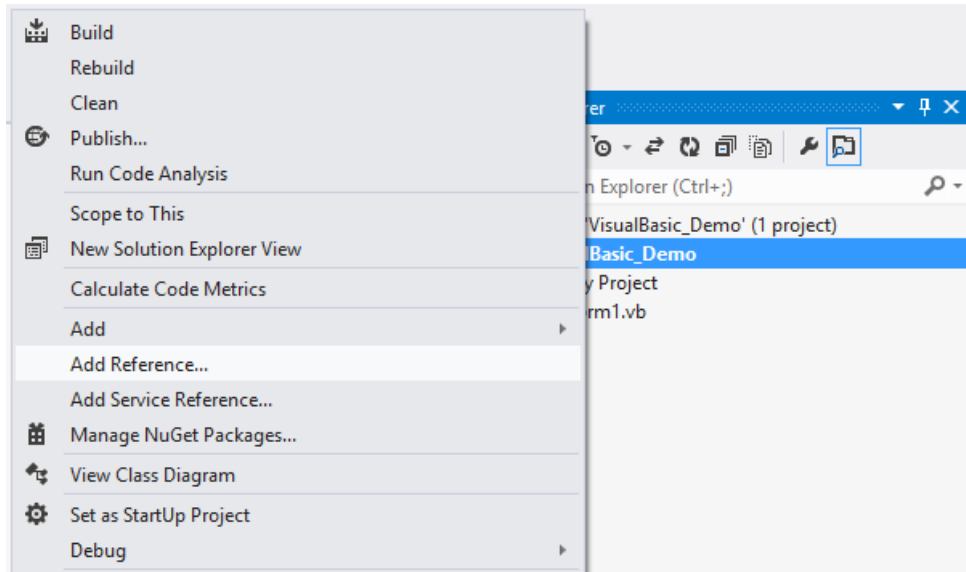
        //Close the fiscal receipt
        fp.CloseReceipt();
    }
    catch (SException ex)
    {
        HandleException(ex);
    }
}
```

Note:

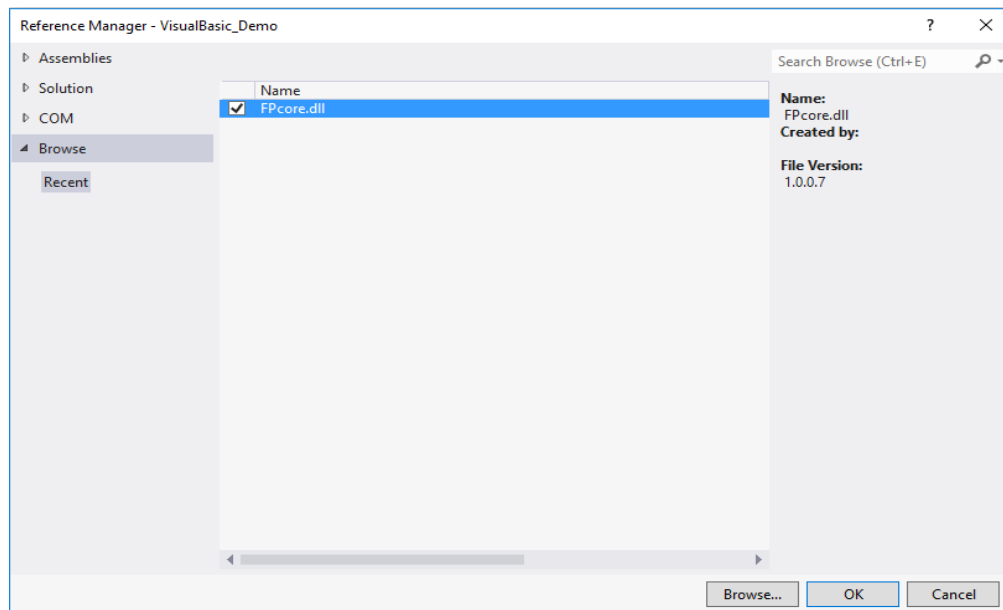
- Exception handling details see in [4.11 Exception Handling](#)
- More detailed demonstration of Library APIs can be found in the demo application in DEMOS/C#/ZFPDemo folder

### 4.3. Visual Basic

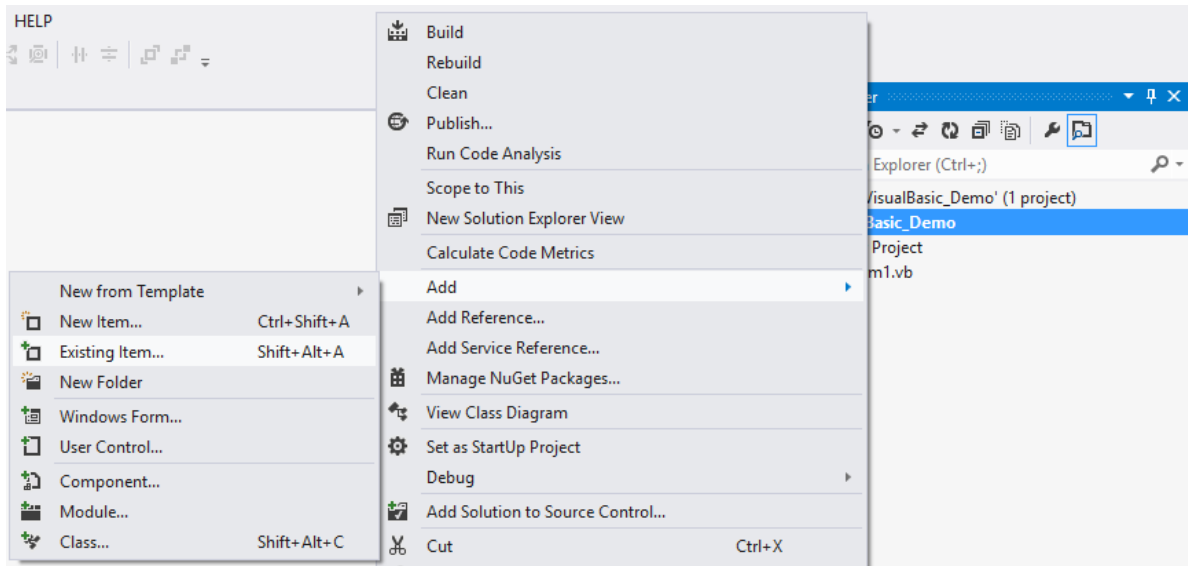
Create new VB project, go to **Solution Explorer** in MS Visual Studio, right click on project name (e.g. VisualBasic\_Demo) and select **Add Reference**:



Browse **Libs\VB** folder and select *FPcore.dll*:



Right click on project name (e.g. VisualBasic\_Demo), select Add/Existing Item from **Libs\VB** folder to add **FP.vb** file:



**Note:**

- **FP.vb file is generated by a tool. Changes to this file may cause incorrect behavior and will be lost, if the code is regenerated.**

Example Visual Basic code snippet with receipt related commands:

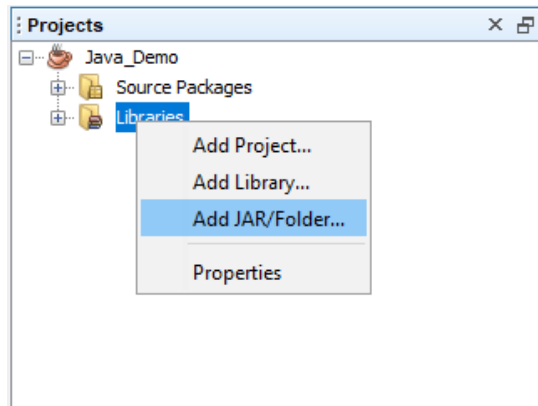
```
Public Sub CreateReceipt()  
    Try  
        'Set server address  
        Dim fp = New FP() With {.ServerAddress = "http://localhost:4444/"}  
  
        'Set device serial port settings  
        fpr.ServerSetDeviceSerialPortSettings("COM5", 115200)  
  
        'Open receipt with Operator Number 1, Password 0000  
        fpr.OpenReceipt(1, "0000", OptionFiscalReceiptPrintType.Step_by_step_printing)  
  
        Public Sub OpenReceipt(operNum As Decimal, operPass As String, optionFiscalReceiptPrintType As ZFPDemoVB.TremolZFP.OptionFiscalReceiptPrintType)  
            Opens a fiscal receipt assigned to the specified operator  
        End Sub  
  
        'Sale article  
        fp.SellPLUwithSpecifiedVAT("Tomato", OptionVATClass.VAT_class_A, 1.29D, 3.21D, Nothing,  
            Nothing, Nothing, Nothing, "", "")  
  
        'Subtotal  
        fpr.Subtotal(OptionPrinting.Yes, OptionDisplay.Yes, Nothing, Nothing)  
  
        'Payment  
        fpr.Payment(OptionPaymentType.Payment_0, 4.15D)  
  
        'Close the fiscal receipt  
        fpr.CloseReceipt()  
    Catch ex As SException  
        HandleException(ex)  
    End Try  
End Sub
```

**Note:**

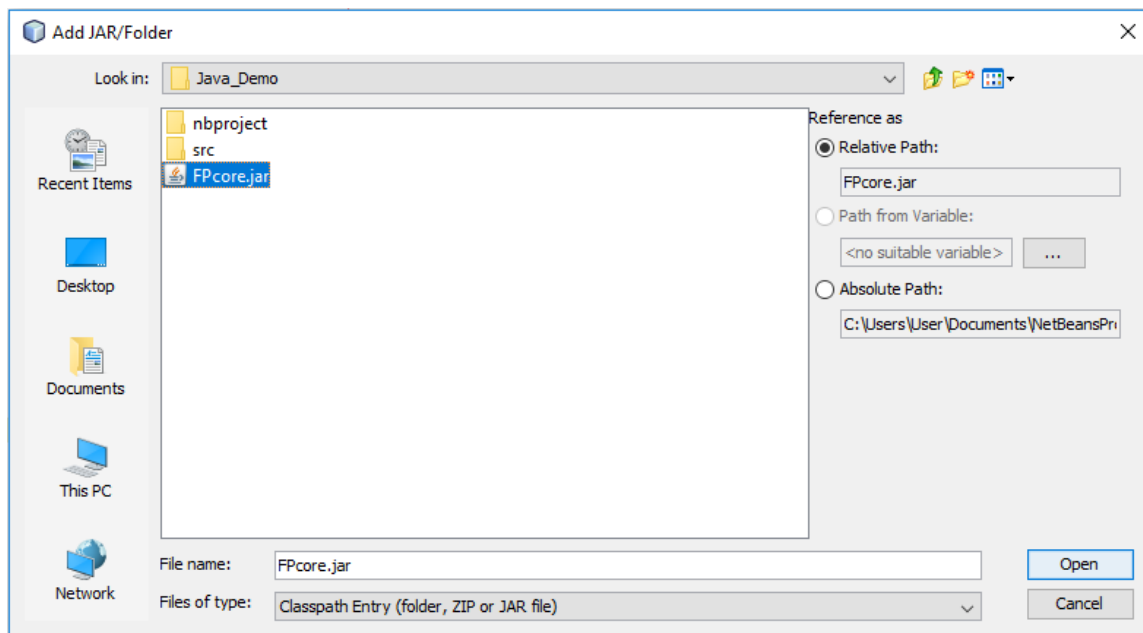
- Exception handling details see in [4.11 Exception Handling](#)
- More detailed demonstration of Library APIs can be found in the demo application in DEMOS/VB/ZFPDemo folder

## 4.4. Java

Create new Java project, go to **Projects** panel, right click on **Libraries** and select **Add JAR/Folder**:



Browse **Libs\Java** folder and select *FPcore.jar*:





Copy **Libs\Java\TremolZFP** folder to **src** folder:

This PC > Documents > NetBeansProjects > Java_Demo > src			^	
Name	Type	Size		
java_demo	File folder			
TremolZFP	File folder			

Example Java code snippet with receipt related commands:

```
public void CreateReceipt() {
    try {
        //Set server address
        FP fp = new FP();
        fp.ServerAddress = "http://LocalHost:4444/";

        //Set Device serial port settings
        fp.ServerSetDeviceSerialPortSettings("COM5", 115200);

        //Open receipt
        fp.OpenReceipt(1d, "0000", OptionFiscalReceiptPrintType.Step_by_step_printing);

        //Sell article
        fp.SellPLUwithSpecifiedVAT("Tomato", OptionVATClass.VAT_Class_A, 1.25d, 3.21d, null,
            null, null, null, "", "");

        //Subtotal
        fp.Subtotal(OptionPrinting.Yes, OptionDisplay.Yes, null, null);

        //Payment
        fp.Payment(OptionPaymentType.Payment_0, 4.15d);

        //Close the fiscal receipt
        fp.CloseReceipt();
    }
    catch(Exception ex){
        HandleException(ex);
    }
}
```

● **void TremolZFP.FP.OpenReceipt(Double operNum, String operPass, OptionFiscalReceiptPrintType optionFiscalReceiptPrintType) throws Exception**

Opens a fiscal receipt assigned to the specified operator and print type depends of FiscalReceiptPrintType parameter.

**Parameters:**

**operNum** Symbols from 1 to 20 corresponding to operator's number

**operPass** 4 symbols for operator's password

**optionFiscalReceiptPrintType** 1 symbol with value: - '0' - Step by step printing - '2' - Postponed printing - 'A' - Buffered Printing

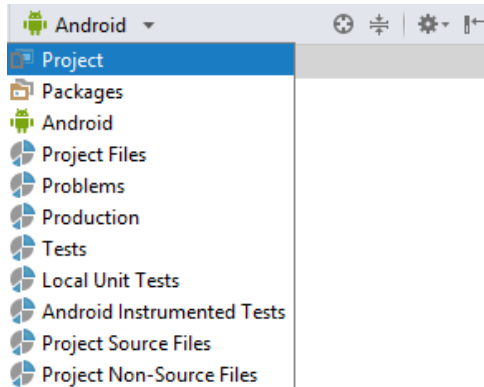
Press 'F2' for focus

**Note:**

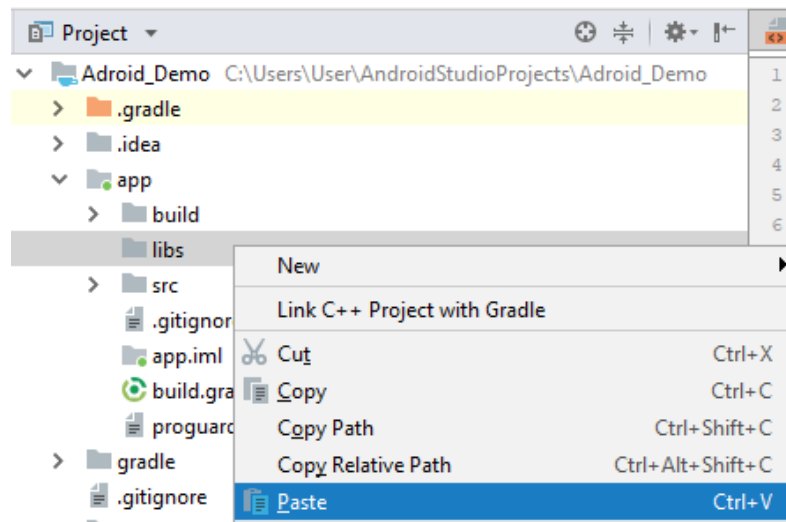
- Exception handling details see in [4.11 Exception Handling](#)
- More detailed demonstration of Library APIs can be found in the demo application in **DEMOS/Java/ZFPDemo** folder

## 4.5. Android

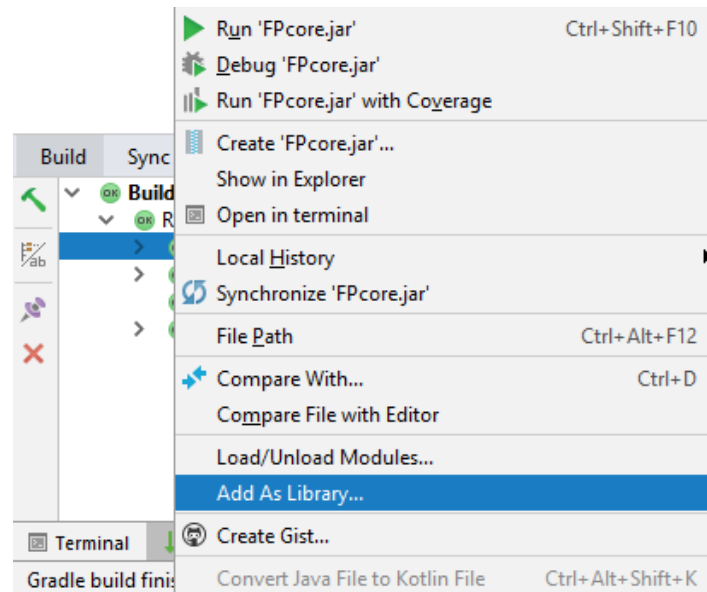
Create new Android project, go to **Project** panel, switch folder structure from Android to **Project**:



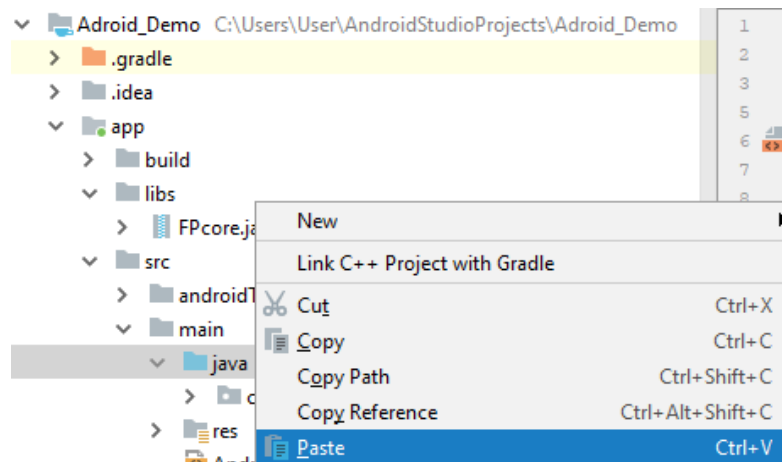
Copy *FPcore.jar* from **Libs\Android**, search for the **app/libs** folder and paste it.



Right click on the *FPcore.jar* file and select **Add as library**:



Copy **Libs\Android\TremolZFP** folder to **app/src/main/java** folder:



Example Android code snippet with receipt related commands:

```
public void createReceipt(){
    try{
        //Set server address
        FP fp = new FP();
        fp.ServerAddress = "http://localhost:4444/";

        //Set device bluetooth settings
        fp.ServerSetAndroidBluetoothDeviceSettings( deviceName: "ZK123456");

        //Open receipt
        fp.OpenReceipt( operNum: 1d, operPass: "0000", OptionFiscalReceiptPrintType.Step_by_step_printing);

        //Sell article
        fp.SellPLUwithSpecifiedVAT( namePLU: "Tomato", OptionVATClass.VAT_Class_A, price: 1.25d, quantity: 3.21d,
            discAddP: null, discAddV: null, discNamed: null, category: null, namePLUextension: "", additionalNamePLU: "" );

        //Subtotal
        fp.Subtotal(OptionPrinting.Yes, OptionDisplay.Yes, discAddV: null, discAddP: 4.15d);

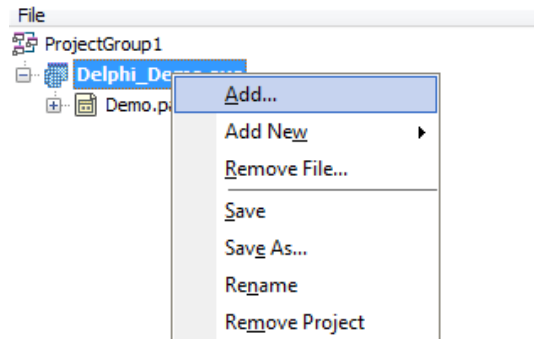
        //Close the receipt
        fp.CloseReceipt();
    }
    catch (Exception ex){
        handleException(ex);
    }
}
```

#### Notes:

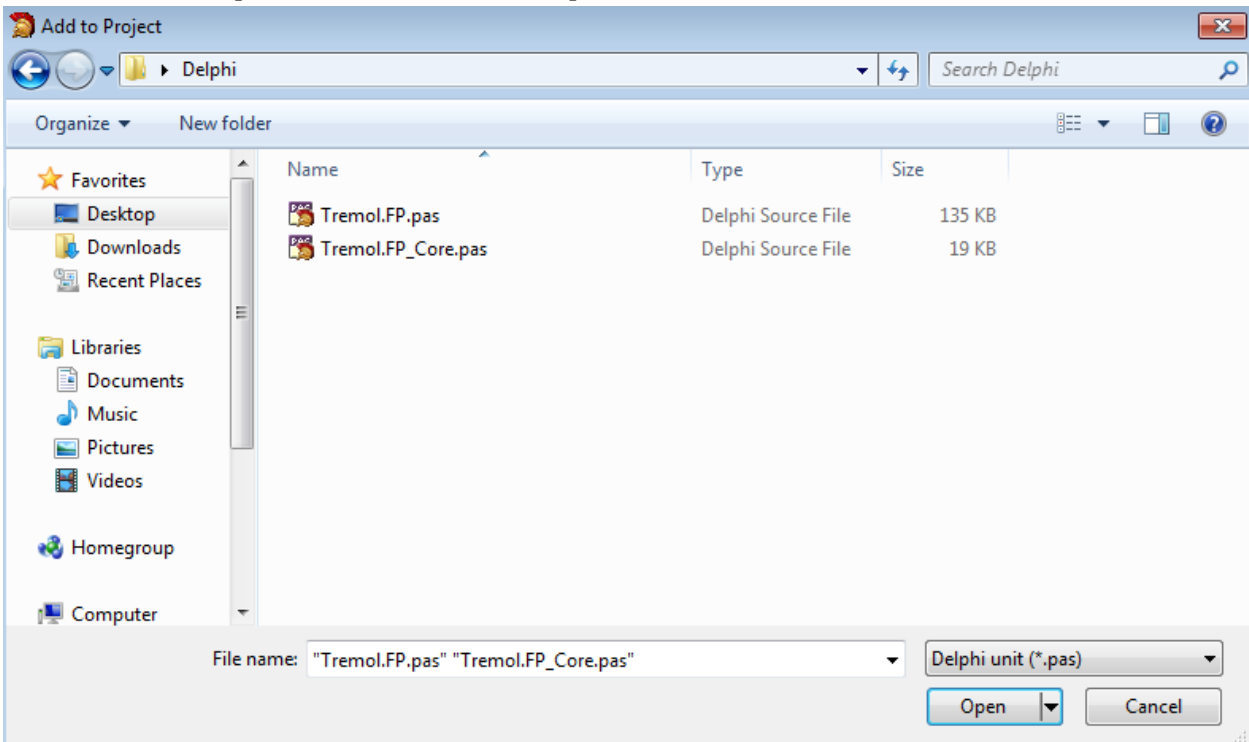
- [Exception handling details see in 4.11 Exception Handling](#)
- When setting “ServerAddress” property, it should be device IP, where ZFPLabServer is installed on.
- All FP methods should be executed in another thread, because the usage of network operations.
- More detailed demonstration of Library APIs can be found in the demo application in DEMOS/Android/ZFPDemo folder.

## 4.6. Delphi

Create new project, copy from **Libs\Delphi** *Tremol.FP.pas* and *Tremol.FP\_Core.pas* files to project folder. Go to **Project manager**, right click on project name (e.g. Delphi\_Demo) select **Add**:



Select *Tremol.FP.pas* and *Tremol.FP\_Core.pas* files:



Example Delphi 7 code snippet with receipt related commands:

```
procedure CreateReceipt();
begin
    // Set server address
    fpc.ServerAddress := 'http://localhost:4444/';

    //Set device serial port settings
    fpc.ServerSetDeviceSerialPortSettings('COM5', 115200);

    //Open receipt
    fpc.OpenReceipt(1, '0000', frpt_Step_by_step_printing);

    //Sell article
    fpc.SellPLUwithSpecifiedVAT('Tomato', vatc_VAT_Class_A, 1.29, 3.21,
    fpc.EmptyDouble, fpc.EmptyDouble, fpc.EmptyDouble, fpc.EmptyDouble, '', '');

    //Subtotal
    fpc.Subtotal(p_Yes, d_Yes, fpc.EmptyDouble, fpc.EmptyDouble);

    //Payment
    fpc.Payment(pt_Payment_0, 4.15);

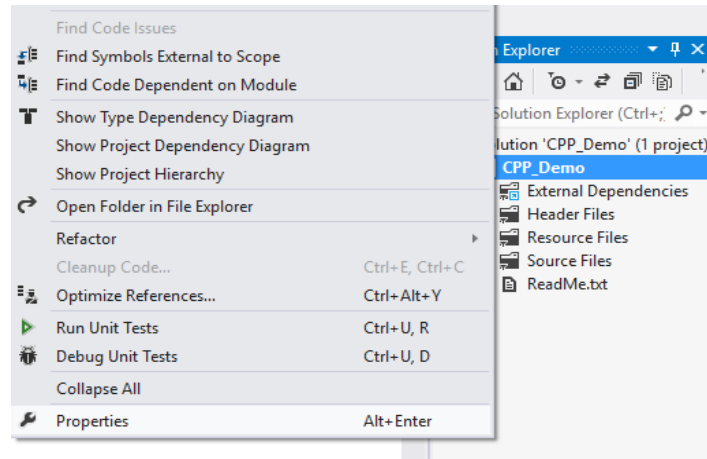
    //Close receipt
    fpc.CloseReceipt();
end;
```

Note:

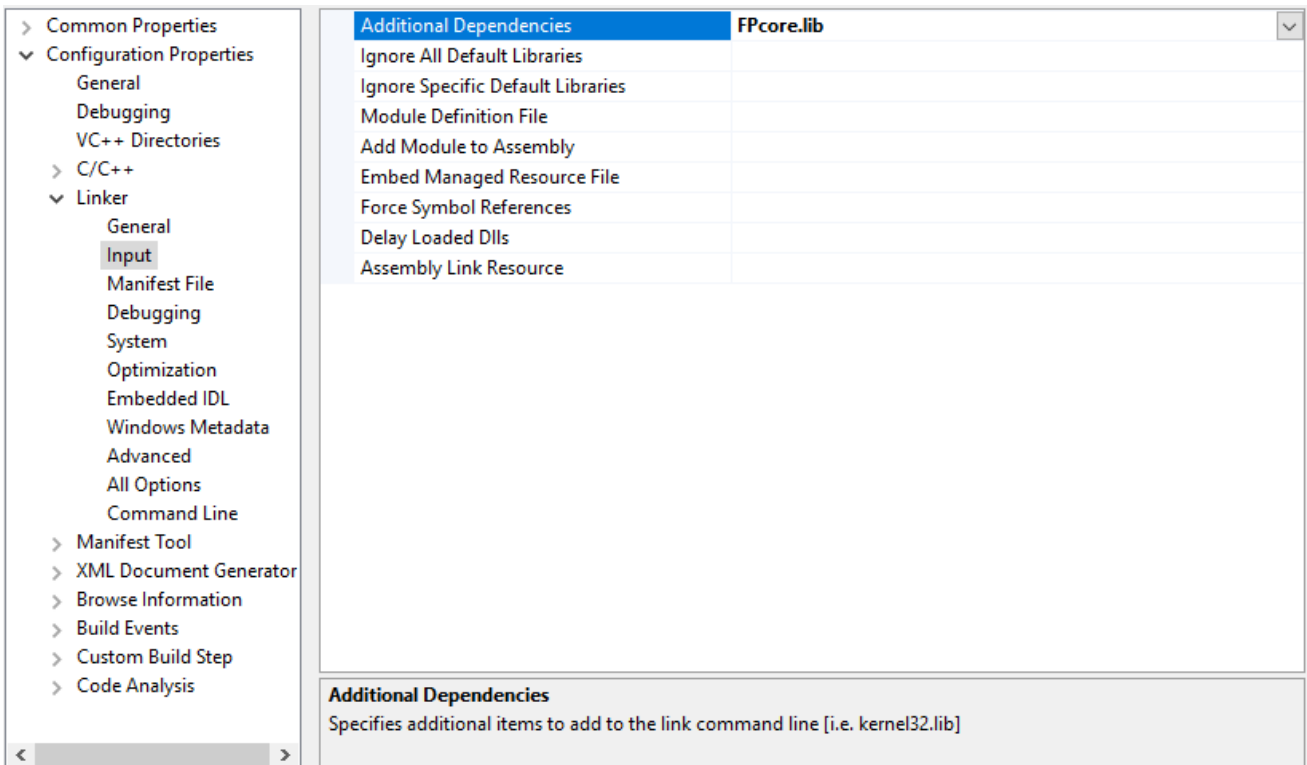
- To use *Tremol.FP.pas* and *Tremol.FP\_Core.pas* install the Indy9 package
- Exception handling details see in [4.11 Exception Handling](#)
- More detailed demonstration of Library APIs can be found in the demo application in DEMOS/Delphi/ZFPDemo folder

## 4.7. C++

Create new project, copy from **Libs\CPP** *FP.cpp*, *FP.h*, *FPcore.dll* and *FPcore.lib* files to project folder. Go to **Project manager**, right click on project name and select **Properties**:



In section **Configuration properties\Linker\Input** add **FPcore.lib** in field **Additional Dependencies**:



Example C++ code snippet with receipt related commands:

```
void CreateReceipt()
{
    //Set server address
    zfp::ServerSetAddress(L"http://localhost:4444/");

    //Set device serial port settings
    zfp::ServerSetDeviceSerialPortSettings(L"COM5", 115200);

    //Open receipt
    zfp::OpenReceipt(1, L"0000", opt::OptionFiscalReceiptPrintType::frpt_Step_by_step_printing);

    //Sell article
    zfp::SellPLUwithSpecifiedVAT(L"Tomato", opt::OptionVATClass::vatc_VAT_Class_A, 1.29, NULL,
        NULL, NULL, NULL, NULL, NULL, NULL);

    //Subtotal
    zfp::Subtotal(opt::OptionPrinting::p_Yes, opt::OptionDisplay::d_Yes, NULL, NULL);

    //Payment
    zfp::Payment(opt::OptionPaymentType::pt_Payment_0, NULL);

    //Close receipt
    zfp::CloseReceipt();
};
```

void Tremol::zfp::OpenReceipt(double operNum, LPCWSTR operPass, Tremol::opt::OptionFiscalReceiptPrintType optionFiscalReceiptPrintType)  
Opens a fiscal receipt assigned to the specified operator and print type depends of FiscalReceiptPrintType parameter.

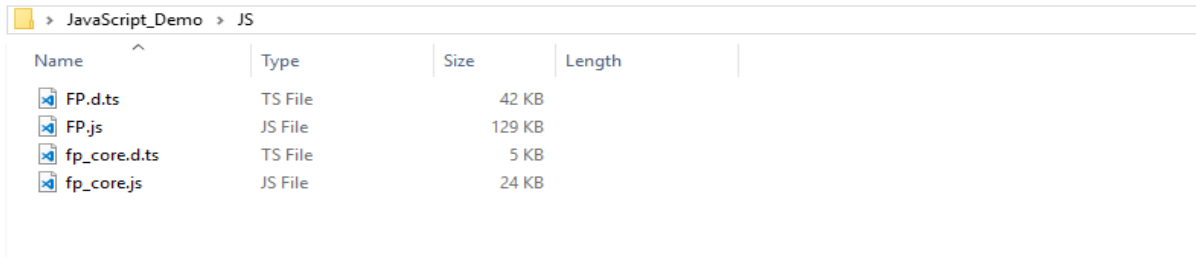
**Note:**

- **Linux is currently not supported.**
- **Exception handling details see in [4.11 Exception Handling](#)**
- **FP.cpp and FP.h files are generated by a tool. Changing the files may cause incorrect behavior and will be lost, if the code is regenerated.**
- **More detailed demonstration of Library APIs can be found in the demo application in DEMOS/CPP folder.**



## 4.8. JavaScript

Create new project, copy folder **Libs\JS** to project folder (e.g. JavaScript\_Demo):



JavaScript_Demo > JS			
Name	Type	Size	Length
FP.d.ts	TS File	42 KB	
FP.js	JS File	129 KB	
fp_core.d.ts	TS File	5 KB	
fp_core.js	JS File	24 KB	

Add the script files to main HTML between the <head> tags:

```
<head>
  <title>JavaScript_Demo</title>

  <script type="text/javascript" src="JavaScript_Demo/FP/fp_core.js"></script>
  <script type="text/javascript" src="JavaScript_Demo/FP/fp.js"></script>
</head>
```

Example JavaScript code snippet with receipt related commands:

```
function CreateReceipt() {
  try {
    //Set server address
    fp.ServerSetSettings("LocalHost", 4444);

    //Set device serial port settings
    fp.ServerSetDeviceSerialSettings("COM5", 115200, false);

    //Open receipt
    fp.OpenReceipt(1, "0000", Tremol.Enums.OptionFiscalReceiptPrintType.Step_by_step_printing);

    //Sell article
    fp.SellPLUwithSpecifiedVAT("Tomato", Tremol.Enums.OptionVATClass.VAT_Class_A, 1.29, 3.21);

    //Subtotal
    fp.Subtotal(Tremol.Enums.OptionPrinting.Yes, Tremol.Enums.OptionDisplay.Yes);

    //Payment
    fp.Payment(Tremol.Enums.OptionPaymentType.Payment_0, 4.15);

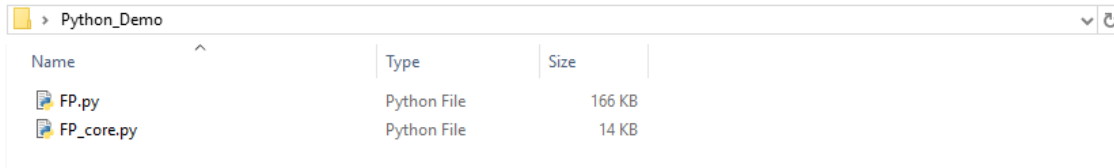
    //Close receipt
    fp.CloseReceipt();
  }
  catch (ex) {
    handleException(ex);
  }
}
```

**Note:**

- Exception handling details see in [4.11 Exception Handling](#)
- FP.js and FP.d.ts files are generated by a tool. Changing the files may cause incorrect behavior and will be lost, if the code is regenerated.
- More detailed demonstration of Library APIs can be found in the demo application in DEMOS/JS folder.

## 4.9. Python

Create new project, copy folder **Libs\PY** to project folder (e.g. Python\_Demo):



> Python_Demo			^	
Name	Type	Size		
FP.py	Python File	166 KB		
FP_core.py	Python File	14 KB		

Example Python code snippet with receipt related commands:

```
def create_receipt():
    try:
        # Set server address
        fp.serverSetSettings("LocalHost", 4444)

        # Set device serial port settings
        fp.serverSetDeviceSerialSettings("COM5", 115200)

        # Open receipt
        fp.OpenReceipt(1, "0000", Enums.OptionFiscalReceiptPrintType.Step_by_step_printing)

        # Sell article
        fp.SellPLUwithSpecifiedVAT("Tomato", Enums.OptionVATClass.VAT_Class_A, 1.29, 3.21)

        # Payment
        fp.Payment(Enums.OptionPaymentType.Payment_0, 4.15)

        # Close receipt
        fp.CloseReceipt()
    except Exception as ex:
        handle_exception(ex)
```

Note:

- Exception handling details see in [4.11 Exception Handling](#)
- FP.py and FP\_core.py files are generated by a tool. Changing the files may cause incorrect behavior and will be lost, if the code is regenerated.
- FP.py and FP\_core.py are compatible with Python 2.x and Python 3.x
- More detailed demonstration of Library APIs can be found in the demo application in DEMOS/PY folder.

## 4.10. PHP

Create new project, copy folder **Libs\PHP** to project folder (e.g. PHP\_Demo):

PHP_Demo				Search PHP_Demo
Name	Date modified	Type	Size	
FP.php	19.4.2019 г. 10:43 ч.	PHP File	129 KB	
FP_Core.php	22.4.2019 г. 12:54 ч.	PHP File	22 KB	

Example PHP code snippet with receipt related commands:

```
function createReceipt(\Tremol\FP $fp) {
    try {
        /** Set Server Address */
        $fp->ServerSetSettings("localhost", 4444);

        /** Set Device serial port settings */
        $fp->ServerSetDeviceSerialSettings("COM6", 115200, TRUE);

        /** Open fiscal receipt with Operator Number 1, Password 0000 */
        $fp->OpenReceipt(1, "0", Tremol\OptionReceiptFormat::Brief, Tremol\OptionPrintVAT::No,
            Tremol\OptionFiscalRcpPrintType::Step_by_step_printing, "ZK000001-0001-0000001");

        /** Sell item */
        $fp->SellPLUwithSpecifiedVAT("Tomato", Tremol\OptionVATClass::VAT_Class_1, 2.5, 3);

        /** Read Subtotal */
        $st = $fp->Subtotal(Tremol\OptionPrinting::No, Tremol\OptionDisplay::Yes, NULL, NULL);

        /** Payment */
        $fp->Payment(Tremol\OptionPaymentType::Payment_0, Tremol\OptionChange::With_Change, $st,
            Tremol\OptionChangeType::Change_In_Cash);

        /** Close fiscal receipt */
        $fp->CloseReceipt();
    } catch (Exception $ex) {
        handleException($ex);
    }
}
```

Note:

- Exception handling details see in [4.11 Exception Handling](#)
- FP.php and FP\_core.php files are generated by a tool. Changing the files may cause incorrect behavior and will be lost, if the code is regenerated.
- More detailed demonstration of Library APIs can be found in the demo application in DEMOS/PHP folder.

## 4.11. Exception Handling

If an error occurs, the server (driver) returns error object (in most cases, extends base exception of the corresponding programming language) and in addition contains - error type, error message and two status bytes (STE1, STE2) for detailed error description.

- **Error type** - contains error code, indicating the type of error. Some of the most important codes are:
  - **9 - ServMismatchBetweenDefinitionAndFPResult** - The current library version and the fiscal device firmware are not matching.
  - **104 - ServerDefsMismatch** - The current library version and server definitions version do not match.
  - **101 - ServerConnectionError** - Connection between client application and server is not established.
  - **30 - ServSockConnectionFailed** - Connection between server and fiscal device is not established.
  - **40 - FPException** - Error occurred in the fiscal device when executing the last command.
- **Status bytes (STE1, STE2)** - two bytes representing more detailed information, when **40 - FPException** has occurred. **STE1** - provides information about the error in the fiscal device, **STE2** - command error code.

STE1	STE2
0x30 OK	0x30 OK
0x31 Out of paper, printer failure	0x31 Invalid command
0x32 Registers overflow	0x32 Illegal command
0x33 Clock failure or incorrect date&time	0x33 Z daily report is not zero
0x34 Opened fiscal receipt	0x34 Syntax error
0x35 Payment residue account	0x35 Input registers overflow
0x36 Opened non-fiscal receipt	0x36 Zero input registers
0x37 Registered payment but receipt is not closed	0x37 Unavailable transaction for correction
0x38 Fiscal memory failure	0x38 Insufficient amount on hand
0x39 Incorrect password	0x3A No access
0x3A Missing external display	
0x3B 24hours block – missing Z report	
0x3C Overheated printer thermal head.	
0x3D Interrupt power supply in fiscal receipt	
0x3E Overflow EJ	
0x3F Insufficient conditions	

- **Error message** - short message, describing detailed information about the cause of the error.

### Note:

- **More error types information and exception handling details can be found in any of the provided demos.**

## 4.12. Development/Testing without using auto-generated library

If a library for specific language is not provided, fiscal device operations can be performed by sending **HTTP GET** request to **ZFPLabServer**. The server might be running locally or remotely.

**API** docs and test environment can be found at the following url location:

<http://localhost:4444>

The end user software sends commands in the format: <http://ip:4444/command>

### Examples:

[http://localhost:4444/OpenReceipt\(OperNum=1,OperPass=0\)](http://localhost:4444/OpenReceipt(OperNum=1,OperPass=0))

[http://localhost:4444/SellPLUWithSpecifiedVat\(NamePLU=Article1,OptionVATClass=B,Price=1.2,Quantity=2,DiscAddP=5,DiscAddV=0\)](http://localhost:4444/SellPLUWithSpecifiedVat(NamePLU=Article1,OptionVATClass=B,Price=1.2,Quantity=2,DiscAddP=5,DiscAddV=0))

[http://localhost:4444/CashPayCloseReceipt\(\)](http://localhost:4444/CashPayCloseReceipt())

[http://localhost:4444/PrintDailyReport\(OptionZeroing=Z\)](http://localhost:4444/PrintDailyReport(OptionZeroing=Z))

<http://localhost:4444/PaperFeed>

### Setting of connection to fiscal device:

- TCP:  
[http://localhost:4444/Settings\(tcp=1,ip=10.10.10.113,port=8000,password=123456\)](http://localhost:4444/Settings(tcp=1,ip=10.10.10.113,port=8000,password=123456))
- Serial port:  
[http://localhost:4444/Settings\(tcp=0,com=COM3,baud=115200\)](http://localhost:4444/Settings(tcp=0,com=COM3,baud=115200))
- Auto find fiscal device on serial port:  
[http://localhost:4444/FindDevice\(usefound=1\)](http://localhost:4444/FindDevice(usefound=1))

### Response format:

- Successful execution with no data returned:

```
<Res Code="0"></Res>
```

- Successful execution with data returned:

```
<Res Code="0">
  <Res Name="VATrateA" Value="20.00" Type="Decimal"/>
  <Res Name="VATrateB" Value="09.00" Type="Decimal"/>...
</Res>
```

- Error response. Res Code is not 0:

```
<Res Code="40">
  <Err Type="FPException" STE1="30" STE2="34" FPLibErrorCode="04">
    <Message>
      FP: OK; Command: Syntax error. FP: 06 71 30 34 37 35 0A
    </Message>
  </Err>
</Res>
```

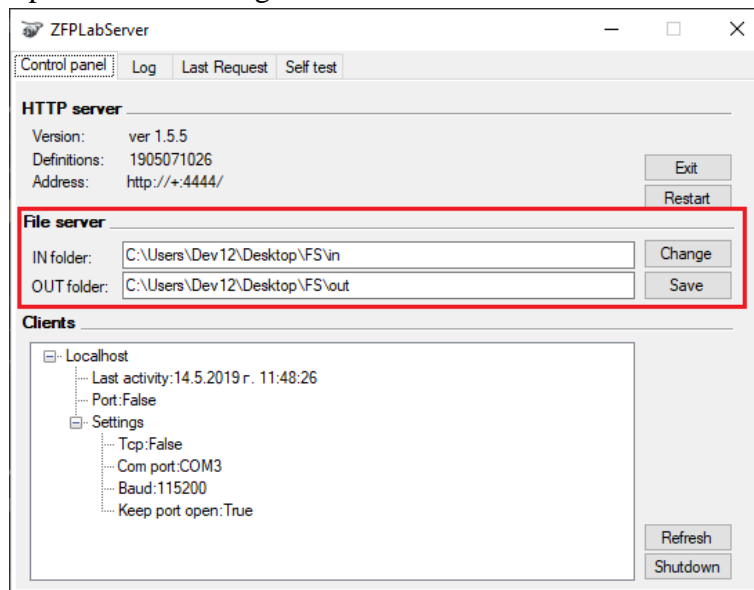
### Notes:

- All commands can be executed from a browser.
- The full list of commands and their definition may be retrieved from <http://localhost:4444>
- The same usage rules and error checks apply.
- If the command has no parameters the brackets may be omitted.
- Commands and parameters are not case sensitive.

## 5. File server

### 5.1. Configuration

After ZFPLabServer is installed (installation details can be found in [3.2](#)), the installation folder contains “FILE IN” and “FILE OUT” folders, which are used as communication input and output. Default paths can be changed from the Server UI - **Control Panel** tab:



If the server is running in **headless mode (without GUI)** or OS is Linux, a file with path settings - “FileServerSettings.xml” should be created next to the executable with format:

```
<FileServerFolders>
  <FolderIn Path="C:\FILE_IN" />
  <FolderOut Path="C:\FILE_OUT" />
</FileServerFolders>
```

“FileServerSettings.xml” configuration is default for Linux.

- **"FileServerCommands"** folder contains **all available**, supported protocol-based commands in xml format, which can be used for execution after filling in the values of command parameters. ZFPLabServer can execute the xml commands **separately or combined in one file**, allowing concise and customized workflow management.
- The function names of all languages APIs, the file commands from "FileServerCommands" folder and “zfpdef” names from **Protocol\_description\_xx\_XXXXXXX.pdf** are identical.
- Connection settings can be set using either “\_Settings.xml” or “\_FindDevice.xml”, described in detail in section [5.2.1. Connection settings](#)

## 5.2. Examples

**Demos/FileServer** folder contains file server example files, showing how commands can be combined. All examples introduced in this document were prepared in accordance with specific protocol description.

The commands can be simplified by removing non-compulsory tags or leaving them blank. Option tags with the enumerated valid values only provide information for the possible value of the enclosing tag. They can be omitted.

### 5.2.1. Connection settings

To choose connection type for fiscal device, fill in and execute "\_Settings.xml" file.

- Serial port – complete "com" and "baud" value.
- TCP/IP – complete "IP", "port" and "password" value.

When using **Serial** communication, the **TCP** connection <Arg> tags can be skipped and vice versa.

```
<Command Name="settings">
  <Args>
    <Arg Name="com" Value="COM1" />
    <Arg Name="baud" Value="115200" />

    <Arg Name="tcp" Value="0" /> | Use "tcp" with "value = 0" for Serial port connection
                                | Use "tcp" with "value = 1" for TCP/IP connection
    <Arg Name="ip" Value="10.10.10.113" />
    <Arg Name="port" Value="8000" />
    <Arg Name="password" Value="123456" />
  </Args>
</Command>
```

- “\_FindDevice.xml” can be used to automatically scan all available COM ports (virtual USB included), stops at the first detected device and returns port number and baud rate values. The newly found port can automatically be used for the subsequent connections to the Fiscal Device if the command is executed with parameter “usefound = 1”.

```
<Command Name="finddevice">
  <Arg Name="usefound" Value="1" />
</Command>
```

#### Note:

- Executing “\_Settings.xml” forces the file server to re-establish the connection to the fiscal device, it is advisable for TCP connections the file to be sent only once.
- “\_FindDevice.xml” is not for frequent usage. It can cause delay issues as a result of scanning multiple virtual COM ports.

### 5.2.2. Simple fiscal receipt

"FiscReceipt.xml" contains three different commands to issue fiscal receipt:

- Opening fiscal receipt by entering operator number and password values.
- Selling article with predefined name, VAT and price.
- Closing the receipt with cash payment.

#### Note:

- In the example below there are non-compulsory fields which can be omitted if they do not have values - quantity, discount/addition in percent (DiscAddP)

```
<Command Name="OpenReceipt">
  <Args>
    <Arg Name="OperNum" Value="2" />
    <Arg Name="OperPass" Value="0000" />
    <Arg Name="OptionReceiptFormat" Value="1"/>
    <Arg Name="OptionPrintVAT" Value="1"/>
    <Arg Name="OptionFiscalRcpPrintType" Value="0"/>
  </Args>
</Command>
<Command Name="SellPLUwithSpecifiedVAT">
  <Args>
    <Arg Name="NamePLU" Value="Office chair with leather" />
    <Arg Name="OptionVATClass" Value="A"/>
    <Arg Name="Price" Value="225.00" />
    <Arg Name="Quantity" Value="3" Compulsory="false" />
    <Arg Name="DiscAddP" Value="-10" Compulsory="false" />
  </Args>
</Command>
<Command Name="CashPayCloseReceipt">
```

Fill these to open a fiscal receipt

Fill these with the desired product information to sell it

### 5.2.3. Reading date and time

```
<Command Name="ReadDateTime">
  <Response ACK="false">
    <Res Name="DateTime" Value="" Type="DateTime" MaxLen="10" Format="dd-MM-yyyy HH:mm" />
  </Response>
</Command>
```

### 5.2.4. Daily report

"Z" for zeroing / "X" without zeroing:

```
<Command Name="PrintDailyReport">
  <Arg Name="OptionZeroing" Value="Z" />
</Command>
```



## 5.2.5. Invoice receipt with postponed printing

```
<Command Name="OpenInvoiceWithFreeCustomerData">
  <Args>
    <Arg Name="OperNum" Value="2" />
    <Arg Name="OperPass" Value="0000" />
    <Arg Name="OptionInvoicePrintType" Value="2"/>
    <Arg Name="Recipient" Value="Tremol OOD" />
    <Arg Name="Buyer" Value="Michael A." />
    <Arg Name="VATNumber" Value="104593442" />
    <Arg Name="UIC" Value="BG104593442" />
    <Arg Name="Address" Value="Veliko Turnovo, Asti 10" />
    <Arg Name="OptionUICType" Value="0"/>
  </Args>
</Command>
<Command Name="SellPLUwithSpecifiedVAT">
  <Args>
    <Arg Name="NamePLU" Value="Printing paper stack" />
    <Arg Name="OptionVATClass" Value="A"/>
    <Arg Name="Quantity" Value="1" Compulsory="false" />
    <Arg Name="Price" Value="10.00" />
  </Args>
</Command>
<Command Name="CashPayCloseReceipt" />
```

Fill these with the required information to open an invoice receipt.  
**Note:** "OptionInvoicePrintType" with Value="2" controls the printing type to be postponed.

Fill these with the desired article information to sell.

## 5.2.6. Programming Department

```
<Command Name="ProgDepartment">
  <Args>
    <Arg Name="Number" Value="" />
    <Arg Name="Name" Value="" />
    <Arg Name="OptionVATClass" Value="">
      <Options>
        <Option Name="Forbidden" Value="*" />
        <Option Name="VAT Class A" Value="A" />
        <Option Name="VAT Class B" Value="B" />
        <Option Name="VAT Class C" Value="C" />
        <Option Name="VAT Class D" Value="D" />
        <Option Name="VAT Class E" Value="E" />
      </Options>
    </Arg>
    <Arg Name="Price" Value="" Compulsory="false" />
    <Arg Name="OptionDepPrice" Value="" Compulsory="false">
      <Options>
        <Option Name="Free price disabled" Value="0" />
        <Option Name="Free price disabled for single transaction" Value="4" />
        <Option Name="Free price enabled" Value="1" />
        <Option Name="Free price enabled for single transaction" Value="5" />
        <Option Name="Limited price" Value="2" />
        <Option Name="Limited price for single transaction" Value="6" />
      </Options>
    </Arg>
    <Arg Name="AdditionalName" Value="" Compulsory="false" />
  </Args>
</Command>
```

Choose the desired VAT Class

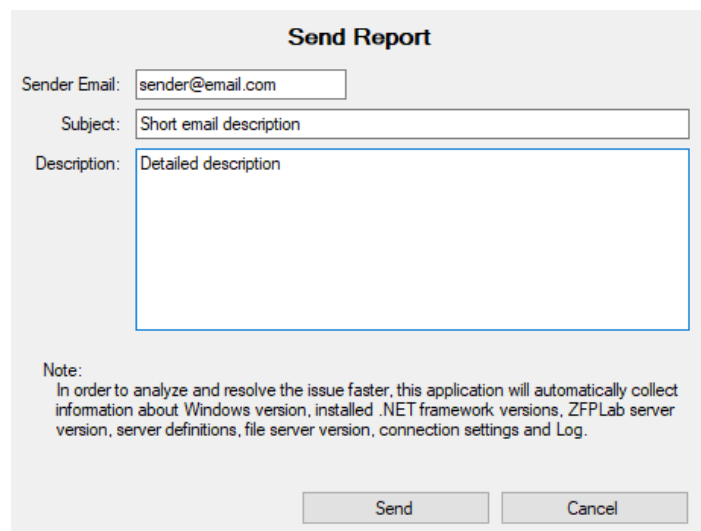
Fill the desired department details

## 6. Contacts and support

Contact us at [zfplab@tremol.bg](mailto:zfplab@tremol.bg) if any questions or issues.

In case of any unexpected error/behavior regarding the fiscal device or libraries, a report can be made with the **report tool** from folder Tools. The tool automatically collects information about Windows version, installed .Net Framework versions, connection settings and ZFPlab server log. The log contains low-level communication data, file server version, ZFPlab server and definition versions. This information will help us to analyze and resolve the issue faster.

To send a report enter **Sender Email** (which we will reply to), **Subject** and **Description** of the issue:



The image shows a 'Send Report' dialog box with a light gray background. At the top, the title 'Send Report' is centered in bold. Below the title, there are three input fields: 'Sender Email:' with the text 'sender@email.com', 'Subject:' with the text 'Short email description', and 'Description:' with the text 'Detailed description'. The 'Description' field is a larger text area. Below these fields, there is a 'Note:' section with a small font size, stating: 'In order to analyze and resolve the issue faster, this application will automatically collect information about Windows version, installed .NET framework versions, ZFPLab server version, server definitions, file server version, connection settings and Log.' At the bottom of the dialog, there are two buttons: 'Send' and 'Cancel'.

For direct contact, send an email to: [zfplab@tremol.bg](mailto:zfplab@tremol.bg) with attached **log** (from **Log** tab in ZFPlab server UI or at <http://localhost:4444/log>) and OS information.