

1. Overview of software package

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

It is a result of 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:

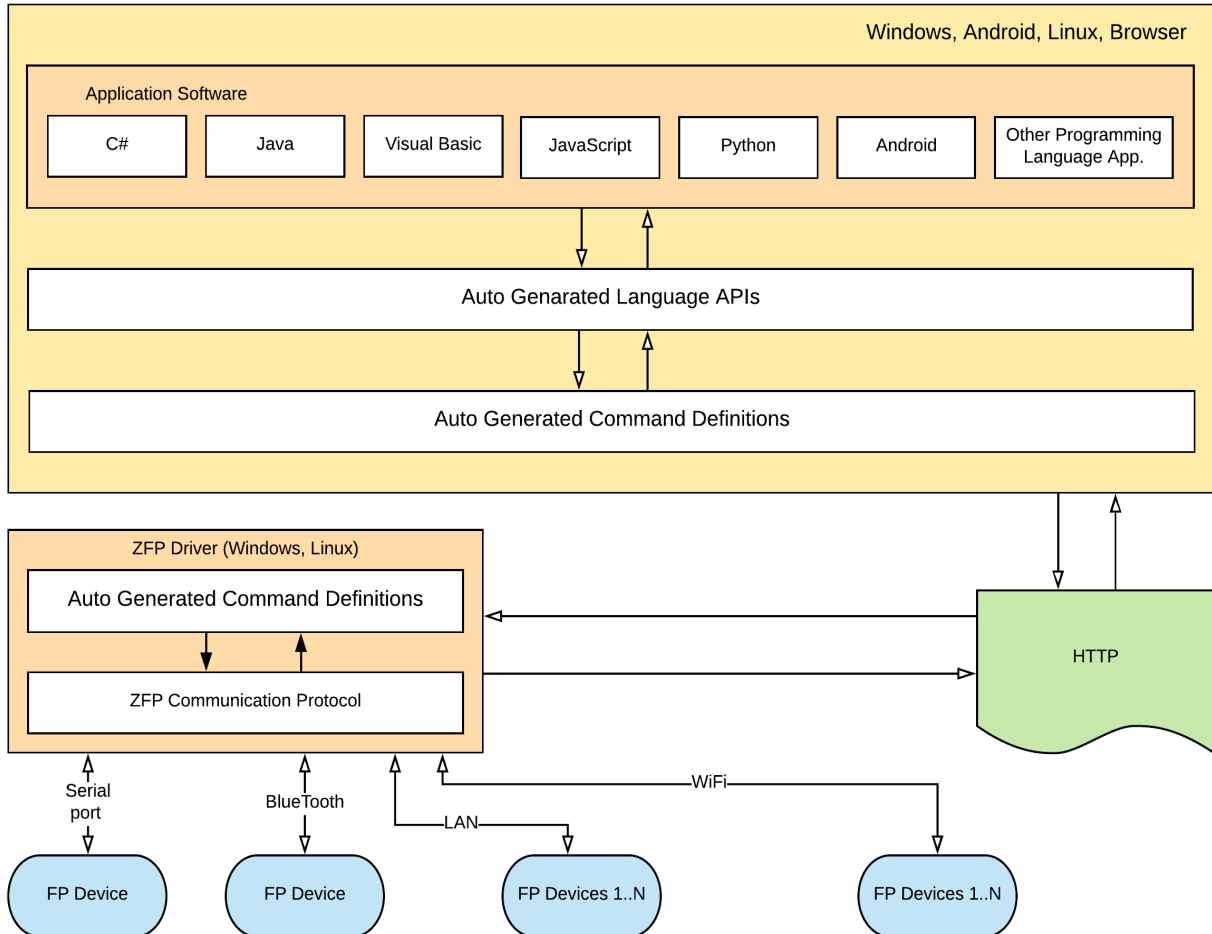
- Run on various platforms: Windows, Android, Linux, Web browser.
- Communicate with the fiscal device over Serial port, Bluetooth, LAN, WIFI.
- Control multiple fiscal devices from one workstation and/or share one fiscal device between multiple workstations.
- Use comprehensive and documented APIs for 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.

2. General software architecture

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



a. Client applications send commands to local or remote driver through the API.

- Communication is done over HTTP.
- The transmitted data is structured according to the definitions.
- APIs for sending commands are auto-generated according to the definition files.
- Command's request and response are fully documented.
- Possibility to control many fiscal devices in the local network.

b. 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.

3. Software components, installment and usage

- **Protocol_description_XX_yyMMddHHmm.pdf** is a fully-described communication protocol of *TREMOL* fiscal devices, working in fiscal printer mode.
- **Libs** 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.
- **Demo** folder contains demo applications written on the different programming languages, based on the auto-generated libraries. The apps demonstrate the usage of the most common fiscal device operations and intercepting the exceptions which can be caused by the device or the ZFPLabServer.
- **ZFPLabServerSetup** is a Windows driver, which translates the requests from Demo app or Customer app to the fiscal device, and translates back the responses from the device to the app. The translation is based on server definitions, which are also auto-generated by the protocol. After the installation – it hides in the PC system tray and starts working on TCP port 4444. Usually it won't be needed to show, but can be used to view the communication log between the apps and the devices, examine problems and set some global settings. One server can serve multiple clients and devices. The communication with the device can be done by Serial or TCP port.
- **ZFPLabServerLinux** is a Linux driver, sharing the same functionality like the Windows version. The folder contains the executable and the definitions. It may be configured for auto-start in the Linux machine.
- **ZFPLabSetup.exe** is a standalone application, which maintains the overall functionality of the protocol and provides the user access to all commands and possibility to execute and examine them.
- **File based server** and demo based on the same architecture is available upon request.

The developer needs the specific library and the server installed to start development.

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 information received is insufficient request detailed status information – the command ReadStatus.

Basic usage of the library is as follows.

- Setting of connection:
 - Server HTTP local or remote address - *ServerAddress*
 - Device TCP/Serial connection-*ServerSetDeviceTcpSettings/ServerSetDeviceSerialPortSettings*
- Sample sequence of commands for issuing a receipt:
 - Fiscal receipt opening – *OpenFiscReceipt()*
 - Sale registrations –*SalePosPLU()* / *SalePLUBelongingDep()*
 - Subtotal amount – *Subtotal()*
 - Payment – *Payment()*
 - Fiscal receipt closure - *CloseFiscReceipt()*
- Performing error checks when executing the commands.

3.1 Development 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/OpenFiscReceipt\(OperNum=1,OperPass=0\)](http://localhost:4444/OpenFiscReceipt(OperNum=1,OperPass=0))

[http://localhost:4444/SalePosPLU\(NamePLU=Article1,OptionVATClass=B,Price=1.2,Quantity=2\)](http://localhost:4444/SalePosPLU(NamePLU=Article1,OptionVATClass=B,Price=1.2,Quantity=2))

[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))

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 Name="VATrateC" Value="00.00" Type="Decimal"/>`

`<Res Name="VATrateD" Value="00.00" Type="Decimal"/>`

`<Res Name="VATrateE" Value="00.00" Type="Decimal"/>`

`<Res Name="VATrateF" Value="00.00" 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.