TfhkaNet LIBRARY

INTEGRATION MANUAL

1.0 Version - Venezuela

February, 2018

The Factory HKA, C.A.

INTEGRATION MANUAL – TfhkaNet LIBRARY

1.0 VERSION - VENEZUELA

The Factory HKA
La California Norte, Callejón Gutiérrez
Edif. Riva, PB Ofic. 2-1, Caracas - Venezuela
Teléfono (212) 237.4112 • 2398176
Departamento de Soporte e Integración
integration@thefactoryhka.com

Revisions History

Revision	Date	Affected Pages	Comments
1.0	23/02/2018	All	Initial Version

Content

Revisions History **7**

Introduction **6**

Hardware Requeriments 6

Software Requeriments 6

Installation **7**

Import and Declarations **10**

Declaration of an object of type Tfhka 10

Initialization of the object Tfhka 10

Functions of the Tfhka Class **11**

OpenFpctrl 11

CloseFpctrl 11

CheckFprinter 11

CheckDrawer 11

ReadFpStatus 11

SendCmd 12

SendFileCmd 12

UploadReportCmd 12

UploadStatusCmd 13

Properties of the Tfhka Class **13**

Object-oriented methods of the Tfhka class **13**

GetPrinterStatus type PrinterStatus 13

GetXReport type ReportData 14

GetX2Report type ReportData 15

GetX4Report type AcumuladosX 16

GetX5Report type AcumuladosX 16

GetX6Report type AcumuladosX 17

GetX7Report type AcumuladosX 17

PrintXReport type Void 17

GetZReport type ReportData and type ReportData[] 18

PrintZReport type Void 21

GetS1PrinterData type S1PrinterData 21

GetS2PrinterData type S2PrinterData 22

GetS2EPrinterData type S2PrinterData 22
GetS21PrinterData type S2PrinterData 23
GetS22PrinterData type S2PrinterData 23
GetS23PrinterData type S2PrinterData 24
GetS3PrinterData type S3PrinterData 24
GetS4PrinterData type S4PrinterData 25
GetS5PrinterData type S5PrinterData 25
GetS6PrinterData type S6PrinterData 26
GetS7PrinterData type S7PrinterData 26
GetS8EPrinterData type S8EPrinterData 26
GetS8PPrinterData type S8PPrinterData 27
GetSVPrinterData type SVPrinterData 27
PrinterException 28
Annexes 29
Annex 1: List of status codes 29
Annex 2: List of error codes 30

Introduction

The TfhkaNet library allows the integration with administrative systems developed under the .NET technology; like C#, J#, VB.NET and ASP.

In the following pages, details are given for the use of the TfhkaNet application library, from the hardware requirements, the installation guide, the API reference of the library and the additional components that may be required for the use of the library and the development of an application under MS .NET platform that targets any of the printers distributed by The Factory HKA.

Hardware Requeriments

- 1GHz Processor or higher
- At least 1GB RAM Memory
- 50MB of free space in the Hard Drive
- Physical Serial Port or USB 2.0

Software Requeriments

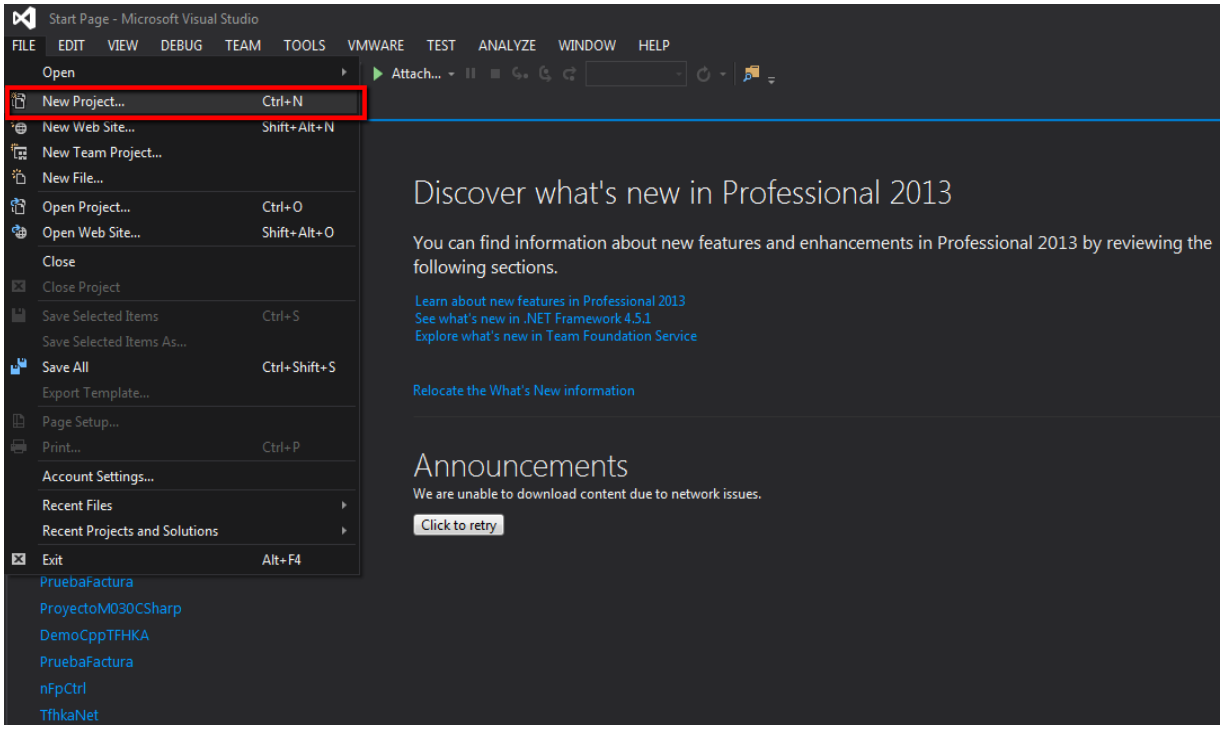
- Windows Operative System
- IDE to handle a development in .NET

Installation

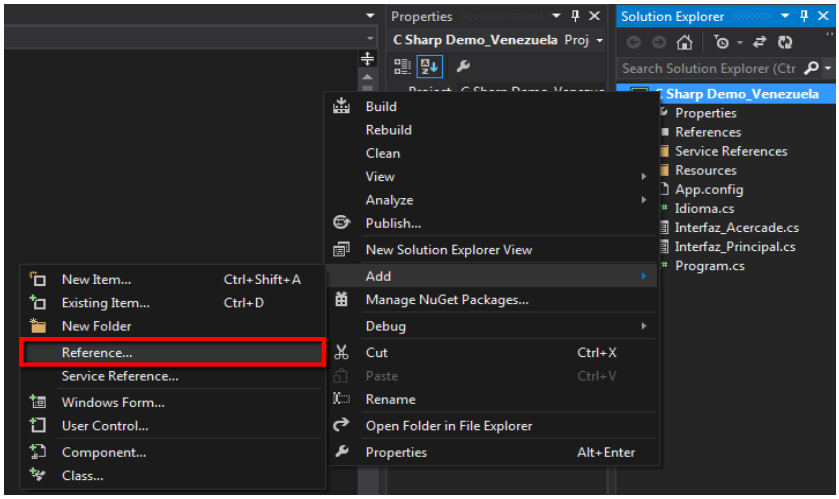
In the downloaded SDK, you will find the three basic tools needed to do the integration between your administrative system and our fiscal printer; the integration library “TfhkaNet.dll”, the dll manual and a functional open source demo where you can observe a practical example of how to properly use the library.

Before beginning to use the library in our development, the first thing that must be done is include and reference the library in our project. This can be achieved by following these steps:

- Create a new project.

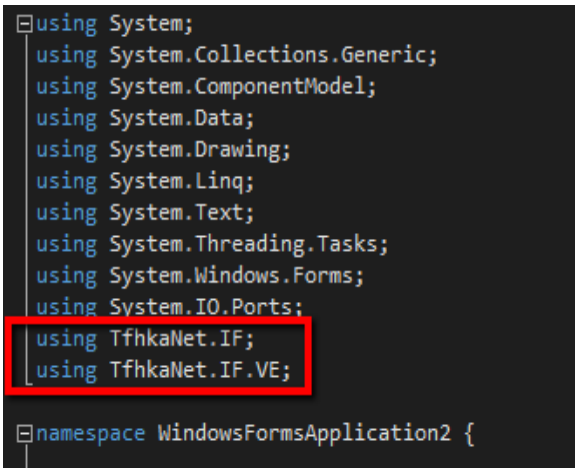


- Add the reference to the integration library, from our IDE.

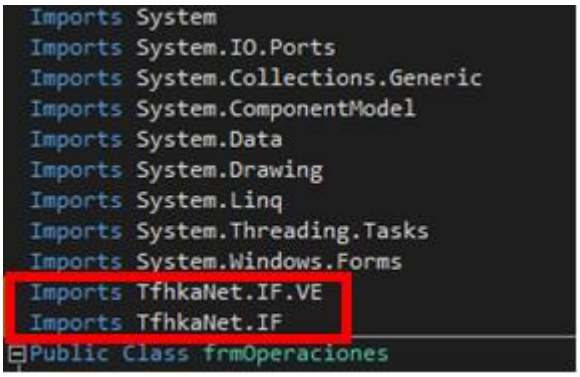


- Once the reference to the library has been added, we can start using it. To begin, we must include the methods that will be used from the library, in the following way:

In C#:



In Visual Basic:



- Finally, we need to create two variables to work with the class during the use of our development and initialize an object of type Tfhka which is the one that will manage the different methods and functions to interact with the fiscal printer.

In C#:

```
namespace WindowsFormsApplication2 {

        public partial class Interfaz_Principal : Form {
        /*SE DECLARAN LAS VARIABLES Y PARAMETROS NECESARIOS*/
        public string version_libreria = "TfhaNet 1.7.4.0";
        public string version_demo = "0.0.0.2";
        public double  suma;
        private Tfhka Impresora;
        private idioma  idioma;
        private bool Respuesta;
        private ReportData Reporte;
        private ReportData[] Reportes;
        private PrinterStatus StatusError;
        private S1PrinterData EstatusS1;
        private S2PrinterData EstatusS2;
        private S3PrinterData EstatusS3;
        private S4PrinterData EstatusS4;
        private S5PrinterData EstatusS5;
        private S6PrinterData EstatusS6;
        private SVPrinterData ModeloPais;

        public Interfaz_Principal()
        {
            InitializeComponent();
            this.Text = "Demo C#.NET - Impresoras Fiscales The Factory HKA C.A.  " + version_demo;
            españolToolStripMenuItem.Checked = true;
            inglesToolStripMenuItem.Checked = false;

            string[] ports = SerialPort.GetPortNames();
            foreach (string port in ports)
            {
                cmbPuerto.Items.Add(port);
            }
            this.idioma = new Idioma();
            this.Impresora = new Tfhka();
        }
```

In Visual Basic:

```
Public Class frmOperaciones
    'Declarar el objeto de la clase "Tfhka"
    Dim Impresora As New Tfhka()
    'Declaramos una variable de control
    Dim Respuesta As Boolean
    'Declarar el objeto de la clase "ReportData"
    Dim Reporte As ReportData
    Dim ReporteArray() As ReportData
    Dim cadena() As ReportData
    'Declarar los objeto de las clases PrinterData
    Dim ReporteS1 As S1PrinterData
    Dim ReporteS2 As S2PrinterData
    Dim ReporteS3 As S3PrinterData
    Dim ReporteS4 As S4PrinterData
    Dim ReporteS5 As S5PrinterData
    Dim ReporteS8E As S8EPrinterData
    Dim ReporteS8P As S8PPrinterData
    'Declarar el objeto de la clase "PrinterStatus"
    Dim StatusError As PrinterStatus
    'Declarar el objeto de la clase "SVPrinterData"
    Dim ModeloPais As SVPrinterData
```

After this, we're ready to begin using the methods and functions that the Tfhka class has.

Import and Declarations

Once the reference to TfhkaNet class has been added, we can begin to work with its attributes and public methods.

In C#:

```
using TfhkaNet.IF;
using TfhkaNet.IF.VE;
```

Declaration of an object of type Tfhka

<Type of Modifier> Tfhka <Name of the Object>

Example:

```
private Tfhka Impresora;
public Tfhka Impresora;
```

Initialization of the object Tfhka

The Tfhka class has only its default constructor. This allows you to initialize the fundamental properties of the interface, in this scenario you must subsequently use the corresponding method to perform the port opening, which is the one explained below.

<Name of the Object> = new Tfhka();

Example:

```
this.Impresora = new Tfhka();
```

In Visual Basic:

```
Imports TfhkaNet.IF

Imports TfhkaNet.IF.VE
```

Declaration and Initialization of an object of type Tfhka

Dim < Name of the Object > As New Tfhka()

Example:

```
Dim Impresora As New Tfhka()
```

Functions of the Tfhka Class

OpenFpctrl

Allows opening the port for communicating with the printer. This method runs in the unique constructor of the class, but can be ran again if required.

BOOLEAN OpenFpctrl(String lpPortName)

- **Parameters:**

String lpPortName: Name of the COM port to be opened.

- **Return:**

True: Port opened successfully.
False: Failure in opening the COM port.

CloseFpctrl

Allows to close the COM port that was previously opened:

VOID CloseFpctrl()

CheckFprinter

Allows to verify if the printer is connected.

BOOLEAN CheckFprinter()

- **Return:**

True: Printer connected.
False: Printer not connected.

CheckDrawer

Allows to verify if the money drawer is connected to the PC.

BOOLEAN CheckDrawer()

- **Return:**

True: Drawer connected.
False: Drawer not connected.

ReadFpStatus

Allows to read the variables of status and error of the printer. When this method is ran, a value for the string variable Estado is set. (See annex 1 and 2).

BOOLEAN ReadFpStatus()

- **Return:**

True: Method ran successfully.
False: Error in method execution.

ⓘ**IMPORTANT:** This method is obsolete and its use is not recommended, it will be eliminated in future versions of the library, use the GetPrinterStatus () method described ahead.

SendCmd

Allows to send commands to the printer, in ASCII characters format, just as described in the integration manuals of each printers and in the TFHKA general protocol and commands manual.

BOOLEAN SendCmd(String Cmd)

- **Parameters:**

String Cmd: ASCII commands that will be sent to the printer.

- **Return:**

True: Method ran successfully.
False: Error in method execution.

SendFileCmd

Allows sending a file containing commands in different lines to the printer.

INT SendFileCmd(String rutaFile)

- **Parameters:**

String rutaFile: Path where is the file to be sent to the printer.

- **Return:**

INT, Number of lines in the file that were sent successfully.

UploadReportCmd

Allows to save a printer report in a text file.

BOOLEAN UploadReportCmd(String Cmd, String file)

- **Parameters:**

String Cmd: Type of report to request. The reports that can be requested are the following (see the integration manual of the respective printer for more details): U0X, U0Z, U1Z, U1X.

String file: Path of the file where the requested report will be saved.

- **Return:**

True: Method ran successfully.
False: Error in method execution.

UploadStatusCmd

This function allows to save a status of the printer in a text file.

BOOLEAN UploadStatusCmd(String Cmd,String file)

- **Parameters:**

String cmd: Type of status to request. The status that can be requested are the following (see the integration manual of the respective printer for more details): S1, S2, S3, S4, S5, S8E, S8P.

String file: Path of the file where the requested status will be saved.

- **Return:**

True: Method ran successfully.
False: Error in method execution.

Properties of the Tfhka Class

- **string ComPort:** Name of the communication port.
- **string Estado:** Description of the current state of the the printer.
- **int SendCmdRetryAttempts:** Number of retries when a method is not correctly processed or NAK is received.
- **int SendCmdRetryInterval:** Time for retry when a method is not correctly processed or NAK is received.
- **bool SerialPortDataReady:** Indicates if the communication port is ready to receive information.
- **byte[] SerialPortInputBuffer:** Data entry buffer through the serial port.
- **int SerialPortReceiveTimeout:** Maximum time for the execution of a method**.**
- **string Status_Error:** Description of the current status and error of the the printer.
- **bool StatusPort:** Indicates whether the communication port is open or closed.

Object-oriented methods of the Tfhka class

GetPrinterStatus type PrinterStatus

Get a status report and printer error in a PrinterStatus type object that contains the code and a description for both the Status and the Current Error.

PrinterStatus GetPrinterStatus()

- **Return:**

An object of type PrinterStatus with the following attributes:

- bool ErrorValidity: Error Validity
- int PrinterErrorCode: Int value of the Error (See Annex 2)
- string PrinterErrorDescription: Error description.
- int PrinterStatusCode: Int value of the Status (See Annex 1)
- string PrinterStatusDescription: Status description.

GetXReport type ReportData

Upload an X Report to the PC using the "U0X" command, updating its data values.

ReportData GetXReport() throws PrinterException

- **Return:**

An object of type ReportData with the following attributes:

- int NumberOfLastZReport: Number of the last Z report emitted
- DateTime ZReportDate: Last Z report date
- int NumberOfLastInvoice: Number of last invoice
- DateTime LastInvoiceDate: Date and hour of last invoice
- int NumberOfLastCreditNote: Number of last credit note
- int NumberOfLastDebitNote: Number of last debit note
- int NumberOfLastNonFiscal: Number of last non-fiscal document
- double FreeSalesTax: Total amount of tax free sales
- double GeneralRate1Sale: General rate 1 sales
- double GeneralRate1Tax: General rate 1 tax sales
- double ReducedRate2Sale: Reduced rate 2 sales
- double ReducedRate2Tax: Reduced rate 2 Tax sales
- double AdditionalRate3Sale: Additional rate 3 sales
- double AdditionalRate3Tax: Additional rate 3 tax sales
- double FreeTaxDebit: Tax free debit notes
- double GeneralRateDebit: General rate 1 debit notes
- double GeneralRateTaxDebit: General rate 1 tax debit notes
- double ReducedRateDebit: General rate 2 debit notes
- double ReducedRateTaxDebit: General rate 2 tax debit notes
- double AdditionalRateDebit: General rate 3 debit notes
- double AdditionalRateTaxDebit: General rate 3 tax debit notes
- double FreeTaxDevolution: Tax free devolutions
- double GeneralRateDevolution: General rate 1 devolutions
- double GeneralRateTaxDevolution: General rate 1 tax devolutions
- double ReducedRateDevolution: General rate 2 devolutions
- double ReducedRateTaxDevolution: General rate 2 tax devolutions
- double AdditionalRateDevolution: General rate 3 devolutions

- double AdditionalRateTaxDevolution: General rate 3 tax devolutions

- **Exception**:

Throws the PrinterException exception.

GetX2Report type ReportData

Upload an X2 Report First Package to the PC using the "U1X" command updating its data values.

ReportData GetX2Report() throws PrinterException

- **Returns:**

An object of type ReportData with the following attributes:

- int NumberOfLastZReport: Number of the last Z report emitted
- DateTime ZReportDate: Last Z report date
- int NumberOfLastInvoice: Number of last invoice
- DateTime LastInvoiceDate: Date and hour of last invoice
- int NumberOfLastCreditNote: Number of last credit note
- int NumberOfLastDebitNote: Number of last debit note
- int NumberOfLastNonFiscal: Number of last non-fiscal document
- double FreeSalesTax: Total amount of tax free sales
- double GeneralRate1Sale: General rate 1 sales
- double GeneralRate1Tax: General rate 1 tax sales
- double ReducedRate2Sale: Reduced rate 2 sales
- double ReducedRate2Tax: Reduced rate 2 Tax sales
- double AdditionalRate3Sale: Additional rate 3 sales
- double AdditionalRate3Tax: Additional rate 3 tax sales
- double FreeTaxDebit: Tax free debit notes
- double GeneralRateDebit: General rate 1 debit notes
- double GeneralRateTaxDebit: General rate 1 tax debit notes
- double ReducedRateDebit: General rate 2 debit notes
- double ReducedRateTaxDebit: General rate 2 tax debit notes
- double AdditionalRateDebit: General rate 3 debit notes
- double AdditionalRateTaxDebit: General rate 3 tax debit notes
- double FreeTaxDevolution: Tax free devolutions
- double GeneralRateDevolution: General rate 1 devolutions
- double GeneralRateTaxDevolution: General rate 1 tax devolutions
- double ReducedRateDevolution: General rate 2 devolutions
- double ReducedRateTaxDevolution: General rate 2 tax devolutions
- double AdditionalRateDevolution: General rate 3 devolutions
- double AdditionalRateTaxDevolution: General rate 3 tax devolutions

- **Exception**:

Throws the exception *PrinterException*

GetX4Report type AcumuladosX

Upload an X4 Report First Package to the PC using the "U0X4" command updating its data values.

AcumuladosX GetX4Report() throws PrinterException

- **Returns:**

An object of type AcumuladosX with the following attributes:

- double FreeTax: Accumulated exempt
- double GeneralRate1: Accumulated base rate 1
- double GeneralRate2: Accumulated base rate 2
- double GeneralRate3: Accumulated base rate 3
- double GeneralRate1Tax: Accumulated tax rate 1
- double GeneralRate2Tax: Accumulated tax rate 2
- double GeneralRate3Tax: Accumulated tax rate 3

- **Exception:**

Throws the exception *PrinterException*

GetX5Report type AcumuladosX

Upload an X5 Report First Package to the PC using the "U0X5" command updating its data values.

AcumuladosX GetX5Report() throws PrinterException

- **Returns:**

An object of type AcumuladosX with the following attributes:

- double FreeTax: Accumulated exempt
- double GeneralRate1: Accumulated base rate 1
- double GeneralRate2: Accumulated base rate 2
- double GeneralRate3: Accumulated base rate 3
- double GeneralRate1Tax: Accumulated tax rate 1
- double GeneralRate2Tax: Accumulated tax rate 2
- double GeneralRate3Tax: Accumulated tax rate 3

- **Exception:**

Throws the exception *PrinterException*

GetX6Report type AcumuladosX

Upload an X6 Report First Package to the PC using the "U0X6" command updating its data values.

AcumuladosX GetX6Report() throws PrinterException

- **Returns:**

An object of type AcumuladosX with the following attributes:

- double FreeTax: Accumulated exempt
- double GeneralRate1: Accumulated base rate 1
- double GeneralRate2: Accumulated base rate 2
- double GeneralRate3: Accumulated base rate 3
- double GeneralRate1Tax: Accumulated tax rate 1
- double GeneralRate2Tax: Accumulated tax rate 2
- double GeneralRate3Tax: Accumulated tax rate 3

- **Exception:**

Throws the exception *PrinterException*

GetX7Report type AcumuladosX

Upload an X7 Report First Package to the PC using the "U0X7" command updating its data values.

AcumuladosX GetX7Report() throws PrinterException

- **Returns:**

An object of type AcumuladosX with the following attributes:

- double FreeTax: Accumulated exempt
- double GeneralRate1: Accumulated base rate 1
- double GeneralRate2: Accumulated base rate 2
- double GeneralRate3: Accumulated base rate 3
- double GeneralRate1Tax: Accumulated tax rate 1
- double GeneralRate2Tax: Accumulated tax rate 2
- double GeneralRate3Tax: Accumulated tax rate 3

- **Exception:**

Throws the exception *PrinterException*

PrintXReport type Void

Print Report X

VOID PrintXReport() throws PrinterException

- **Exception**

Throws the PrinterException exception.

GetZReport type ReportData and type ReportData[]

ReportData GetZReport() throws PrinterException

Upload a Z Report to the PC using the "U0Z" command, updating its data values.

- **Return**

An object of type ReportData with the following attributes:

- int NumberOfLastZReport: Number of the last Z report emitted
- DateTime ZReportDate: Last Z report date
- int NumberOfLastInvoice: Number of last invoice
- DateTime LastInvoiceDate: Date and hour of last invoice
- int NumberOfLastCreditNote: Number of last credit note
- int NumberOfLastDebitNote: Number of last debit note
- int NumberOfLastNonFiscal: Number of last non-fiscal document
- double FreeSalesTax: Total amount of tax free sales
- double GeneralRate1Sale: General rate 1 sales
- double GeneralRate1Tax: General rate 1 tax sales
- double ReducedRate2Sale: Reduced rate 2 sales
- double ReducedRate2Tax: Reduced rate 2 Tax sales
- double AdditionalRate3Sale: Additional rate 3 sales
- double AdditionalRate3Tax: Additional rate 3 tax sales
- double FreeTaxDebit: Tax free debit notes
- double GeneralRateDebit: General rate 1 debit notes
- double GeneralRateTaxDebit: General rate 1 tax debit notes
- double ReducedRateDebit: General rate 2 debit notes
- double ReducedRateTaxDebit: General rate 2 tax debit notes
- double AdditionalRateDebit: General rate 3 debit notes
- double AdditionalRateTaxDebit: General rate 3 tax debit notes
- double FreeTaxDevolution: Tax free devolutions
- double GeneralRateDevolution: General rate 1 devolutions
- double GeneralRateTaxDevolution: General rate 1 tax devolutions
- double ReducedRateDevolution: General rate 2 devolutions
- double ReducedRateTaxDevolution: General rate 2 tax devolutions
- double AdditionalRateDevolution: General rate 3 devolutions
- double AdditionalRateTaxDevolution: General rate 3 tax devolutions

- **Exception**

Throws the PrinterException exception.

ReportData[] GetZReport(string StartDate, string EndDate) throws PrinterException

Performs a fiscal memory reading by date range.

- **Parameters:**

- StartDate: Date of the initial Z report.
- EndDate: Date of the final Z report.

- **Return:**

A list of ReportData [] objects with the information of the Z reports included in the requested interval with the following attributes:

- int NumberOfLastZReport: Number of the last Z report emitted
- DateTime ZReportDate: Last Z report date
- int NumberOfLastInvoice: Number of last invoice
- DateTime LastInvoiceDate: Date and hour of last invoice
- int NumberOfLastCreditNote: Number of last credit note
- int NumberOfLastDebitNote: Number of last debit note
- int NumberOfLastNonFiscal: Number of last non-fiscal document
- double FreeSalesTax: Total amount of tax free sales
- double GeneralRate1Sale: General rate 1 sales
- double GeneralRate1Tax: General rate 1 tax sales
- double ReducedRate2Sale: Reduced rate 2 sales
- double ReducedRate2Tax: Reduced rate 2 Tax sales
- double AdditionalRate3Sale: Additional rate 3 sales
- double AdditionalRate3Tax: Additional rate 3 tax sales
- double FreeTaxDebit: Tax free debit notes
- double GeneralRateDebit: General rate 1 debit notes
- double GeneralRateTaxDebit: General rate 1 tax debit notes
- double ReducedRateDebit: General rate 2 debit notes
- double ReducedRateTaxDebit: General rate 2 tax debit notes
- double AdditionalRateDebit: General rate 3 debit notes
- double AdditionalRateTaxDebit: General rate 3 tax debit notes
- double FreeTaxDevolution: Tax free devolutions
- double GeneralRateDevolution: General rate 1 devolutions
- double GeneralRateTaxDevolution: General rate 1 tax devolutions
- double ReducedRateDevolution: General rate 2 devolutions
- double ReducedRateTaxDevolution: General rate 2 tax devolutions
- double AdditionalRateDevolution: General rate 3 devolutions
- double AdditionalRateTaxDevolution: General rate 3 tax devolutions

- **Exception:**

Throws the PrinterException exception.

ReportData[] GetZReport(int StartReportNumber, int EndReportNumber) throws PrinterException

Performs a fiscal memory reading by number range.

- **Parameters:**

- StartReportNumber: Date of the initial Z report.
- EndReportNumber: Date of the final Z report.

- **Return:**

A list of ReportData [] objects with the information of the Z reports included in the requested interval with the following attributes:

- int NumberOfLastZReport: Number of the last Z report emitted
- DateTime ZReportDate: Last Z report date
- int NumberOfLastInvoice: Number of last invoice
- DateTime LastInvoiceDate: Date and hour of last invoice
- int NumberOfLastCreditNote: Number of last credit note
- int NumberOfLastDebitNote: Number of last debit note
- int NumberOfLastNonFiscal: Number of last non-fiscal document
- double FreeSalesTax: Total amount of tax free sales
- double GeneralRate1Sale: General rate 1 sales
- double GeneralRate1Tax: General rate 1 tax sales
- double ReducedRate2Sale: Reduced rate 2 sales
- double ReducedRate2Tax: Reduced rate 2 Tax sales
- double AdditionalRate3Sale: Additional rate 3 sales
- double AdditionalRate3Tax: Additional rate 3 tax sales
- double FreeTaxDebit: Tax free debit notes
- double GeneralRateDebit: General rate 1 debit notes
- double GeneralRateTaxDebit: General rate 1 tax debit notes
- double ReducedRateDebit: General rate 2 debit notes
- double ReducedRateTaxDebit: General rate 2 tax debit notes
- double AdditionalRateDebit: General rate 3 debit notes
- double AdditionalRateTaxDebit: General rate 3 tax debit notes
- double FreeTaxDevolution: Tax free devolutions
- double GeneralRateDevolution: General rate 1 devolutions
- double GeneralRateTaxDevolution: General rate 1 tax devolutions
- double ReducedRateDevolution: General rate 2 devolutions
- double ReducedRateTaxDevolution: General rate 2 tax devolutions
- double AdditionalRateDevolution: General rate 3 devolutions
- double AdditionalRateTaxDevolution: General rate 3 tax devolutions

- **Exception:**

Throws the PrinterException exception.

PrintZReport type Void

VOID PrintZReport() throws PrinterException

Print the daily Z report.

- **Exception**

Throws the PrinterException exception.

VOID PrintZReport(int StartReportNumber, int EndReportNumber) throws PrinterException

Print a history of Z reports by number range

- **Parameters**

- StartReportNumber: Number of the initial Z report.
- EndReportNumber: Number of the final Z report.

- **Exception**

Throws the PrinterException exception.

VOID PrintZReport(Date StartDate, Date EndDate) throws PrinterException

Print a history of Z reports by date range

- **Parameters**

- StarDate: Date of the initial Z report.
- EndDate: Date of the final Z report.

- **Exception**

Throws the PrinterException exception.

GetS1PrinterData type S1PrinterData

Upload the S1 status to the PC (general parameters information of the printer).

S1PrinterData GetS1PrinterData() throws PrinterException

- **Return:**

An object of type S1PrinterData with the following attributes:

- int CashierNumber: Cashier Number
- double TotalDailySales: Total Amount of Daily Sales
- int LastInvoiceNumber: Last Invoice Number
- int QuantityOfInvoicesToday: Quantity Of Invoices Today
- int LastDebitNoteNumber: Last Debit Note Number
- int QuantityOfDebitNotesToday: Quantity of Debit Notes Today
- int LastCreditNoteNumber: Last Credit Note Number
- int QuantityOfCreditNotesToday: Quantity Of Credit Notes Today
- int NumberNonFiscalDocuments: Number of last Non-Fiscal Document
- int QuantityNonFiscalDocuments: Quantity of Non-Fiscal Documents
- int AuditReportsCounter: Audit Reports Counter
- int DailyClosureCounter: Daily Closure Counter (Z Report)
- string RIF: RIF
- string RegisteredMachineNumber: Registered Machine Number
- DateTime CurrentPrinterDateTime: Current Printer Date and Time

- **Exception:**

Throws the PrinterException exception.

GetS2PrinterData type S2PrinterData

Upload status S2 to the PC (general information on the amounts of the document in progress).

S2PrinterData GetS2PrinterData() throws PrinterException

- **Return:**

An object of type S2PrinterData with the following attributes:

- double AmountPayable: Amount to pay
- int Condition: Transaction condition
- string DataDummy: Fill data
- int NumberPaymentsMade: Amount of payments made
- int QuantityArticles: Number of items
- double SubTotalBases: Subtotal taxable bases
- double SubTotalTax: Tax Subtotal
- int TypeDocument: Type of document

- **Exception**

Throws the PrinterException exception.

GetS2EPrinterData type S2PrinterData

Upload to the PC the status S2E (information of the amounts for exempt of the current document).

S2PrinterData GetS2EPrinterData() throws PrinterException

- **Return**

An object of type S2PrinterData with the following attributes:

- double AmountPayable: Amount to pay
- int Condition: Transaction condition
- string DataDummy: Fill data
- int NumberPaymentsMade: Amount of payments made
- int QuantityArticles: Number of items
- double SubTotalBases: Subtotal taxable bases
- double SubTotalTax: Tax Subtotal
- int TypeDocument: Type of document

- **Exception**

Throws the PrinterException exception.

GetS21PrinterData type S2PrinterData

The S21 status is uploaded to the PC (information on the amounts for rate 1 of the document in progress).

S2PrinterData GetS21PrinterData() throws PrinterException

- **Return**

An object of type S2PrinterData with the following attributes:

- double AmountPayable: Amount to pay
- int Condition: Transaction condition
- string DataDummy: Fill data
- int NumberPaymentsMade: Amount of payments made
- int QuantityArticles: Number of items
- double SubTotalBases: Subtotal taxable bases
- double SubTotalTax: Tax Subtotal
- int TypeDocument: Type of document

- **Exception**

Throws the PrinterException exception.

GetS22PrinterData type S2PrinterData

The S22 status is uploaded to the PC (information on the amounts for rate 2 of the document in progress).

S2PrinterData GetS22PrinterData() throws PrinterException

- **Return**

An object of type S2PrinterData with the following attributes:

- double AmountPayable: Amount to pay
- int Condition: Transaction condition
- string DataDummy: Fill data
- int NumberPaymentsMade: Amount of payments made
- int QuantityArticles: Number of items
- double SubTotalBases: Subtotal taxable bases
- double SubTotalTax: Tax Subtotal
- int TypeDocument: Type of document

- **Exception**

Throws the PrinterException exception.

GetS23PrinterData type S2PrinterData

The S23 status is uploaded to the PC (information on the amounts for rate 3 of the document in progress).

S2PrinterData GetS23PrinterData() throws PrinterException

- **Return**

An object of type S2PrinterData with the following attributes:

- double AmountPayable: Amount to pay
- int Condition: Transaction condition
- string DataDummy: Fill data
- int NumberPaymentsMade: Amount of payments made
- int QuantityArticles: Number of items
- double SubTotalBases: Subtotal taxable bases
- double SubTotalTax: Tax Subtotal
- int TypeDocument: Type of document

- **Exception**

Throws the PrinterException exception.

GetS3PrinterData type S3PrinterData

Upload to the PC the status S3 (information of Flags and rates).

S3PrinterData GetS3PrinterData() throws PrinterException

- **Return**

An object of type S3PrinterData with the following attributes:

- int[ ] AllSystemFlags: All the flags
- double Tax1: Value of rate 1 (%)
- double Tax2: Value of rate 2 (%)
- double Tax3: Value of rate 3 (%)
- int TypeTax1: Type of rate 1 (Included mode = 1, Excluded mode = 2)
- int TypeTax2: Type of rate 2 (Included mode = 1, Excluded mode = 2)
- int TypeTax3: Type of rate 3 (Included mode = 1, Excluded mode = 2)

- **Exception**

Throws the PrinterException exception.

GetS4PrinterData type S4PrinterData

Upload the status S4 to the PC (information about the amounts in the means of payment).

S4PrinterData GetS4PrinterData() throws PrinterException

- **Return**

An object of type S4PrinterData with the following attributes:

- double[] AccumulatedMountsAllMeansOfPayment: List of the accumulated amounts of the means of payment of the fiscal printer, and the amount corresponding to donations.

- **Exception**

Throws the PrinterException exception.

GetS5PrinterData type S5PrinterData

Upload the S5 status to the PC (information about the audit memory).

S5PrinterData GetS5PrinterData() throws PrinterException

- **Return**

An object of type S5PrinterData with the following attributes:

- double AuditMemoryFreeCapacity: Availability in audit memory (MB)
- int AuditMemoryNumber: Audit memory number
- double AuditMemoryTotalCapacity: Total capacity of the audit memory (MB)
- int NumberRegisteredDocuments: Amount Documents in the audit report
- string RegisteredMachineNumber: Registration number of the fiscal printer
- string RIF: Fiscalization RIF of the printer

- **Exception**

Throws the PrinterException exception.

GetS6PrinterData type S6PrinterData

Upload the S6 status to the PC (SLIP mode information).

S6PrinterData GetS6PrinterData() throws PrinterException

- **Return**

An object of type S6PrinterData with the following attributes:

- string Bit_Facturacion: Indicates the presence of paper in the billing station
- string Bit_Slip: Indicates the presence of paper in the Slip / Checkbook station
- string Bit_Validacion: Indicates the presence of paper in the validation station

- **Exception**

Throws the PrinterException exception.

GetS7PrinterData type S7PrinterData

Upload the S7 status to the PC (SLIP mode information).

S7PrinterData GetS7PrinterData() throws PrinterException

- **Return**

An object of type S7PrinterData with the following attributes:

- String MICR: MICR reading of the check

- **Exception**

Throws the PrinterException exception.

GetS8EPrinterData type S8EPrinterData

Upload the S8E status to the PC (ticket header information).

S8EPrinterData GetS8EPrinterData() throws PrinterException

- **Return**

An object of type S8EPrinterData with the following attributes:

- string Header1: Header, line number 1
- string Header2: Header, line number 2
- string Header3: Header, line number 3
- string Header4: Header, line number 4
- string Header5: Header, line number 5
- string Header6: Header, line number 6
- string Header7: Header, line number 7

- string Header8: Header, line number 8

- **Exception**

Throws the PrinterException exception.

GetS8PPrinterData type S8PPrinterData

Upload the S8P status to the PC (ticket footer information).

S8PPrinterData GetS8PPrinterData() throws PrinterException

- **Return**

An object of type S8PPrinterData with the following attributes:

- string Footer1: Footer, line number 1
- string Footer2: Footer, line number 2
- string Footer3: Footer, line number 3
- string Footer4: Footer, line number 4
- string Footer5: Footer, line number 5
- string Footer6: Footer, line number 6
- string Footer7: Footer, line number 7
- string Footer8: Footer, line number 8

- **Exception**

Throws the PrinterException exception.

GetSVPrinterData type SVPrinterData

Upload the SV status to the PC (model information and printer country).

SVPrinterData GetSVPrinterData() throws PrinterException

- **Return**

An object of type SVPrinterData with the following attributes:

- PModel Model: Reading the printer model
- PCountry Country: Reading the printer country

- **Exception**

Throws the PrinterException exception.

PrinterException

Represents a type of exception thrown by some object creation methods of the previously defined structures when an error occurs in the transaction with the fiscal printer. It is composed of the following elements:

- PrinterStatus StatusError: Returns an object that contains information about the Status and the Error when the exception was generated.
- Message: Contains the description of the exception thrown.

Annexes

Annex 1: List of status codes

STATUS		
Return (Hex)	**Return (Decimal)**	**Commentary**
0	0	Unknow state.
1	1	In test and standby mode.
2	2	In test mode and issuance of tax documents.
3	3	In test mode and issuance of non-fiscal documents.
4	4	In fiscal mode and waiting.
5	5	In fiscal mode and issuance of tax documents.
6	6	In fiscal mode and issuance of non-fiscal documents.
7	7	In fiscal mode, close full charge of fiscal memory and waiting.
8	8	In fiscal mode, close full charge of fiscal memory and issuance of tax documents.
9	9	In fiscal mode, close full load of fiscal memory and issuance of non-fiscal documents.
0A	10	In fiscal mode, full load of fiscal memory and waiting.
0B	11	In fiscal mode, full load of fiscal memory and issuance of fiscal documents.
0C	12	In fiscal mode, full load of fiscal memory and issuance of non-fiscal documents.

Note: The “TfhkaNet.dll” library returns this value in decimal.

Annex 2: List of error codes

ERROR		
Return (Hex)	**Return (Decimal)**	**Commentary**
00	0	There is no error.
01	1	End in paper delivery.
02	2	Mechanical error in paper delivery.
03	3	End in paper delivery and mechanical error.
50	80	Invalid command or invalid value.
54	84	Invalid rate.
58	88	No directives assigned.
5C	92	Invalid command.
60	96	Fiscal Error.
64	100	Fiscal memory error.
6C	108	Fiscal memory full.
70	112	Full buffer (must send the reset command)
80	128	Communication error.
89	137	There is no answer.
90	144	LRC Error.
91	145	Internal error api.
99	153	Error opening the file.

Note: The “TfhkaNet.dll” library returns this value in decimal.