EpsonFpMate

Development Guide

REVISION HISTORY

A	All	preliminary	March, 1st 2005	C. Guietti
B	All	First edition	March, 15th 2005	C. Radaelli
C	All	Add some clarification sentences	May, 27th 2005	C. Radaelli
D	All	Management of display messages; Misspelling corrections; Review of the sample files	June, 9th 2005	C. Guietti
E	All	Added “Open Drawer” command Added modification of “Not allowed words” Added BarCode Printing	August, 5th 2005	C. Guietti
F	All	Added “Print from DGFE of last receipt” command	February, 6th 2006	C. Guietti
G	8	FP210 and FP285 have been replaced by FP81 and FP90II	February, 7th 2006	C. Guietti
H	All	Added: management of Fiscal closures log. management of graphic files direct EpsonFp2.ocx commands	June, 21st 2006	C. Guietti
I	All	Added: Option to avoid the MessageBox: “Paper Near to end” Language options Command to read and save E/J Command to read and save Printer Status	December, 6th 2006	C. Guietti
J	All	Updated description of methods: printSetProperty printGetProperty printRecTotal (ticket payment mode)	January, 11th 2006	C. Guietti
K	All	Added notes related to fiscal receipt Errors corrections	January, 23th 2006	C. Guietti
L	All	Added management of EFT-POS devices Added management of Fiscal Documents	May, 16th 2008	C. Guietti

Index

1 Overview

Aim of this manual is to provide information to the Sw designers who want to interface Epson fiscal printers using the EpsonFpMate utility.

EpsonFpMate is a software utility designed for the software houses who want to use Epson fiscal printer in a very easy way, all the communication protocol management are under the control of EpsonFpMate.

Epson FP printer can be easily managed by means of simple TXT file or XML one, based on the result of the print-out, EpsonFpMate return a file with the information relating the print-out of last fiscal receipt or the result of the last command.

Epson FPMate is provided with two utilities:

- EpsonFpMmateConfig: used to set the environment
- EpsonFpMate: used to manage the fiscal print

EpsonFpMate has been developed under the Windows.net technology. Using WIN98 or WIN2K operating system it is required to load on your computer the framework.

2 List of related documents

FP xxx manuale operatore
ActiveX user manual

3 Abbreviation

E/J: Electronic Journal (Dispositivo Giornale di Fondo Elettronico)

4 EpsonFpMateConfig

Allows to create and/or edit the “Settings.xml” file that is used by the application EpsonFpMate.

EfpsonFpMateConfig tool must be used after installing the package in order to configure:

- Operational setting
- Input option
- Output option
- Fiscal Printers
- EFT-POS devices

4.1 Operational settings

4.1.1 Language setting

Allows to select the language used for menus, descriptions and messages generated by EpsonFpMate software.
Note that it is not related to the language used in the EpsonFpmateConfig forms.

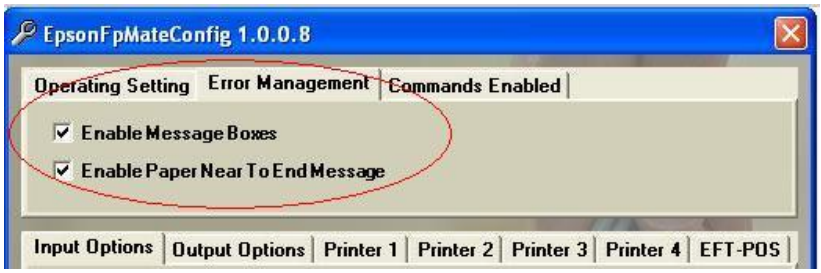
4.1.2 Operating mode

EpsonFpMate can operate in Free Running mode and Single Shot mode.

- In SingleShot mode, the application must be started by a command that have as argument the file that must be processed
- In FreeRunning mode, the application, once started looks, once a second, for files to be processed in a programmed folder.
    - In FreeRunning mode, it is required to indicate the path where the input files will located and the rule defined for the name.
    - In FreeRunning mode, the input files will be deleted after processing.

4.1.3 Error Management

In case of errors EpsonFpMate returns the error codes in the return file and stores it in the output log files.
Moreover it can be chosen if errors must be presented as message boxes on the monitor. In this case the software requires an action by the operator in order to decide how to manage the error.
If the error message boxes are enabled it is also possible to select if the diagnostic information "Paper Near to End" must be presented on the monitor as a message box.

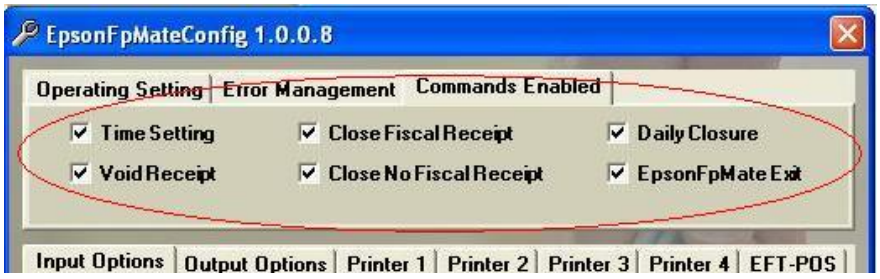


4.1.4 Commands Enabled

EpsonFpMate by default allows some commands that can be sent by the operator using appropriate buttons on the EpsonFpMate main form.
Some of them can be disabled by means of EpsonFpMateConfig.
Commands that can be enabled/disabled are:

- Date and Time setting of Fiscal Printer
- Void Receipt
- Forced Closure of open Fiscal Recipt
- Forced Closure of open Not Fiscal Receipt
- Printout of Z-report (Daily closure)
- EpsonFpMate program exit. When disabled it is not possible to close the program even using the "Alt F4" command.

Disabled commands are not visible on the EpsonFpMate form.

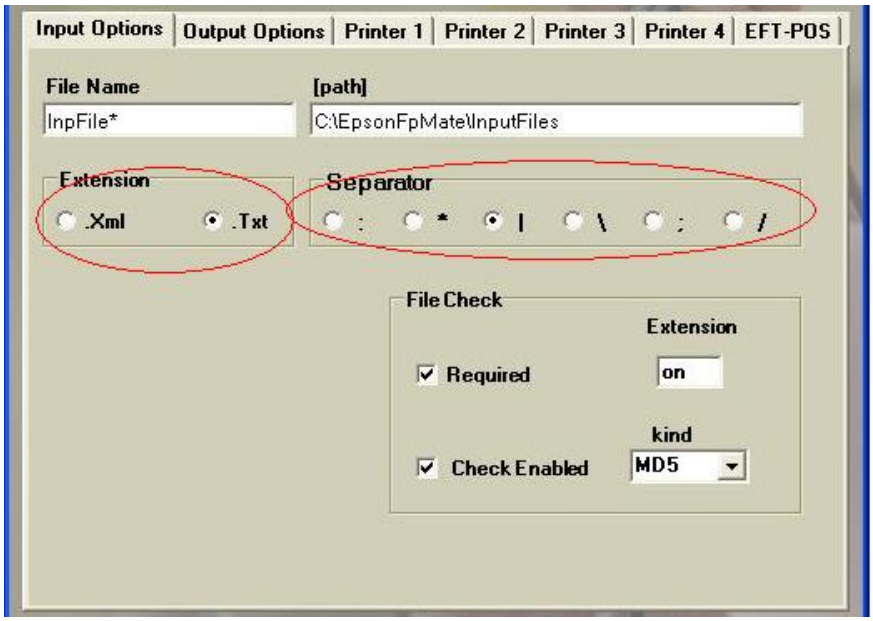


4.2 Input Options

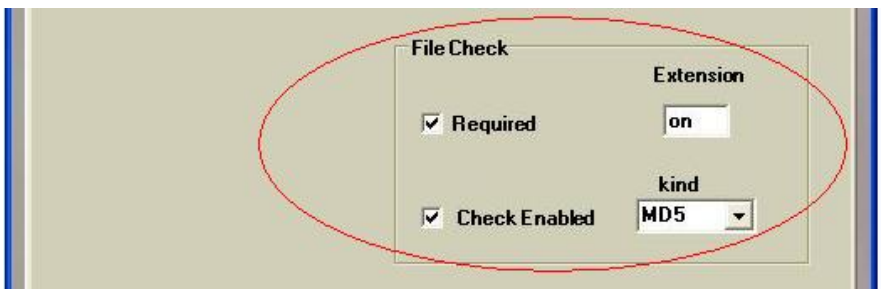
It allows to define:

- The rules for the input file name and the path of the directory where EpsonFpMate in Free Running mode has to look for it.

- The format of input/output files selecting the appropriate file extension.
  - Xml
  - Txt, if selected TXT file format, it must be indicated also the separator to be used to separate the data fields inside the records

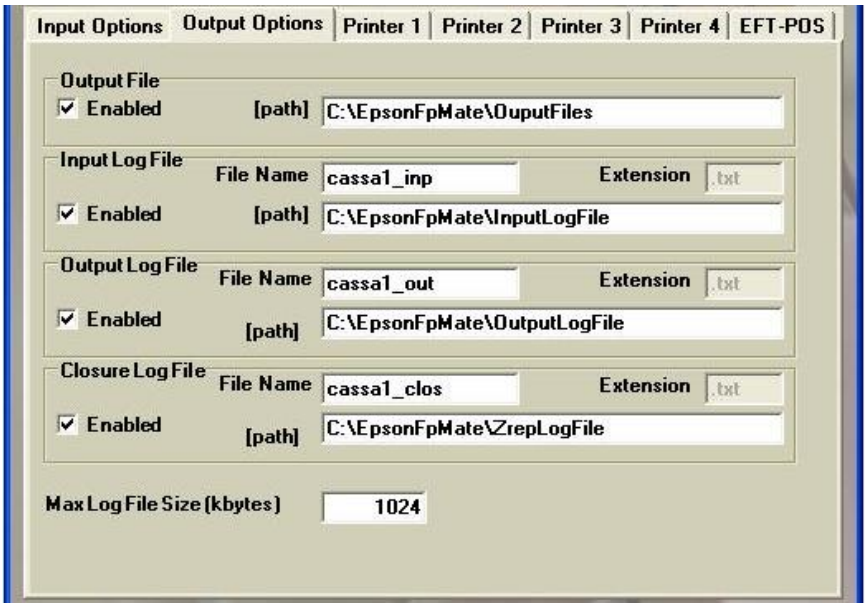


- File Check options:
  - When “Required” is selected, EpsonFpMate requires that the input file must be written in the input directory before writing a check file (same name and different extension). It indicates that the file transfer of the input file has been completed.
  - When “Check Enabled” is selected, EpsonFpMate verifies that the check file is got applying the MD5 algorithm to the input file. The input file is managed only if the check file is compliant. Otherwise no check is performed.



4.3 Output Options

By means of this folder can be selected the output files and the operating conditions that ca be set for EpsonFpMate.



Output file: if enabled, it creates an output file for each input file received. It is required to indicate the path were the file must be stored.

Input Log File: It keeps trace of the input files. Default name is “InputLogFpMate”. If enabled it is required to indicate the path were the must be stored.

Output Log File: Input Log File: It keeps trace of all the error conditions. Default name is “OutLogFpMate”. If enabled it is required to indicate the path were the must be stored.

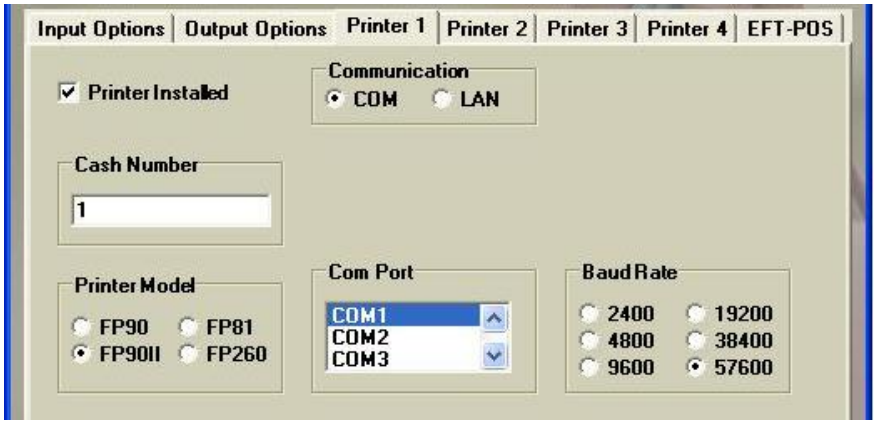
Closure Log File: Input Log File: It keeps trace of all the closures. Default name is “RepZLog”. If enabled it is required to indicate the path were the must be stored.

Max Long File Size allows to define the maximum size admitted for log files. Once reached the maximum size, the file is stored as old and a new log file is started.

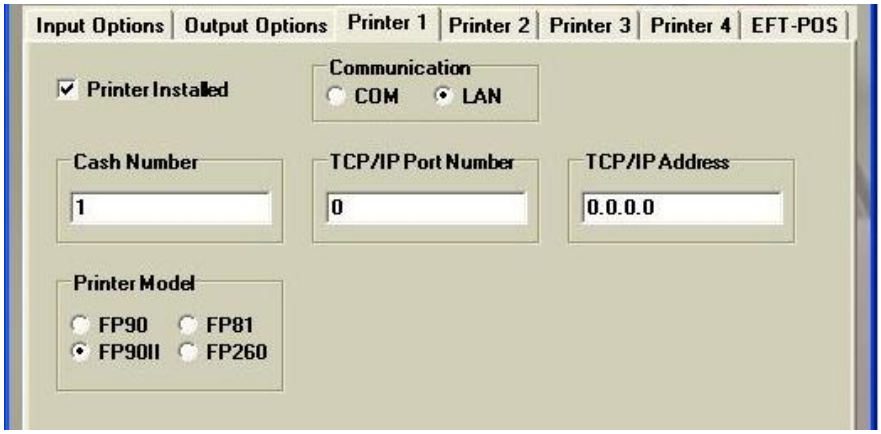
4.4 Printer Options

- It allows to manage up to four printers:
- For each printer, if installed, it must be selected:
  - The printer model:
  - Communication mode (RS232 or TCP/IP) and the related parameters
  - CashDesk Id number: range 0 - 999

In case of serial communication it must be selected the printer model and the communication parameters



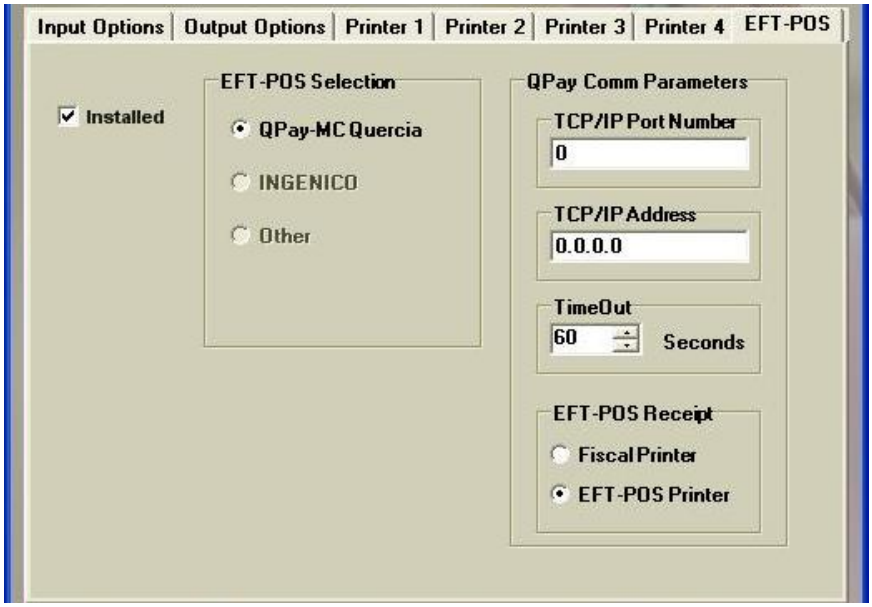
In case of LAN interface communication it must be selected the printer model and the LAN PORT with IP address.



4.5 EFT-POS Options

In case of installation of EFT-POS device it must be configured indicating:

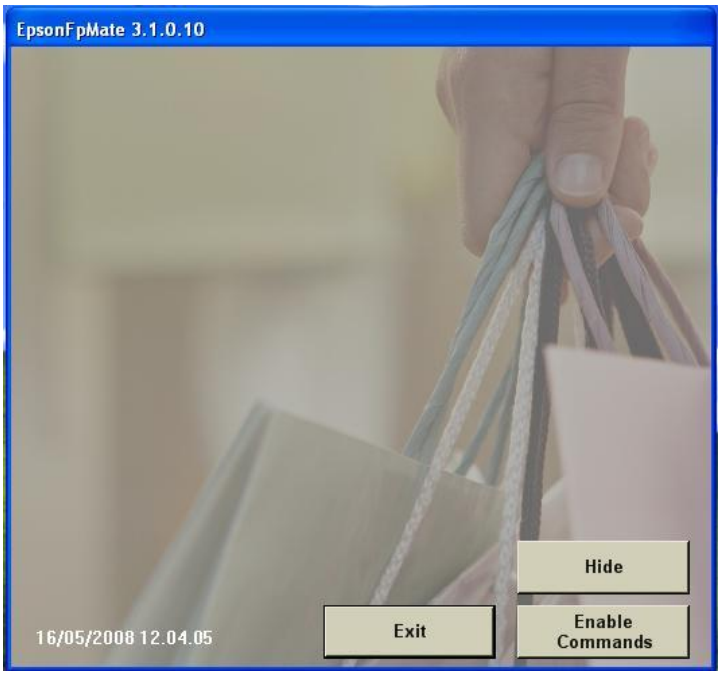
- The EFT-POS device used.
- The communication parameters (Port Number and TCP-IP address)
- The communication time-out before generating communication error
- The printer selected to produce the EFT-POS receipt



5 EpsonFpMate in Free Running mode

Once the system has been configured (by means of EpsonFpMateConfig) the application EpsonFpMate can be started.
The icon represented by a bridge remains visible in the system try.

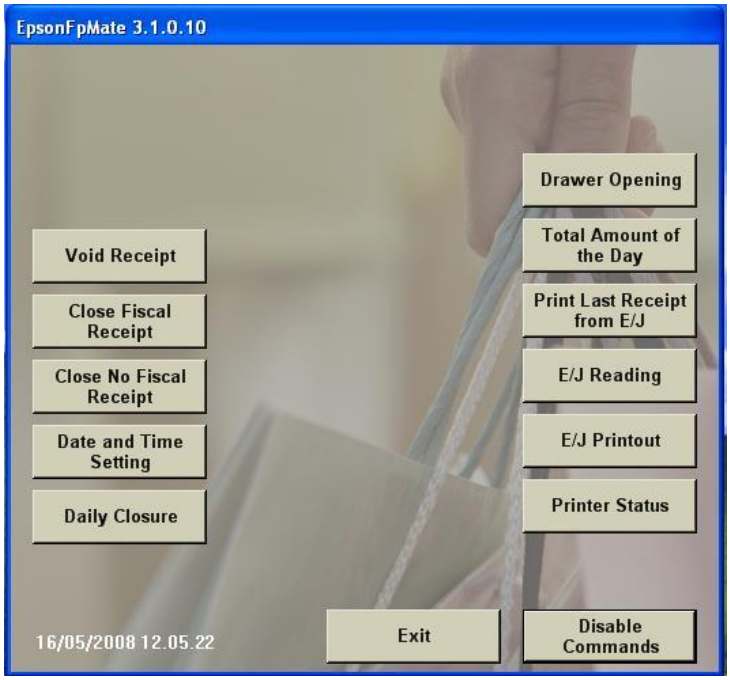
Once started the applications looks for files and, if found, it process them sequentially.
According with the settings it generates output and/or log files.
Clicking on the icon, the application window becomes visible



In the case of the Exit button has been disabled the bottom part of the form will appear like the following image.

The application can be hidden or closed using the appropriate buttons.

The application can be also used to perform local management. Local management can be enabled by means of the “Enable Commands” button.

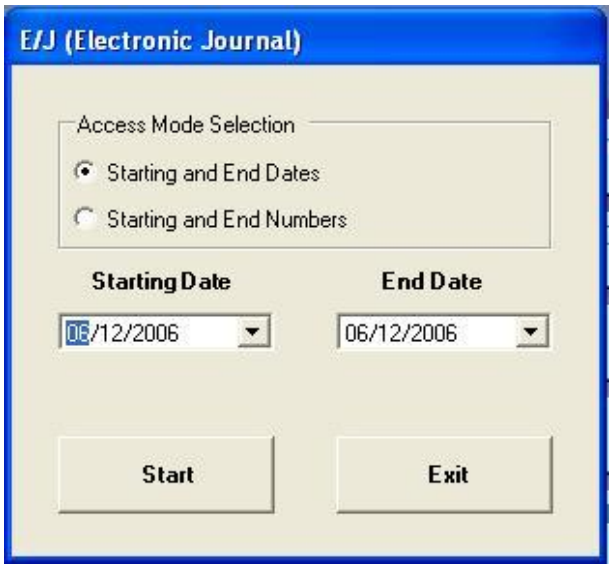


Obviously if one or more commands has been disabled, the related buttons will be hidden.

In local management mode, it allows to send to the printer commands in order to perform:

- Fiscal closures
- Date and time settings
- Printout of Electronic Journal content
- Reading of
  - Printer status
  - Electronic Journal
  - Fiscal amount
- Close a opened fiscal/Not fiscal receipt
- Void a fiscal receipt opened
- Open the Cash drawer
- Printout, from Electronic Journal, of a not fiscal receipt that reports data of last fiscal receipt issued.

By means of Electronic Journal relating option it is possible to read/print the content of E/J.


E/J (Electronic Journal)
Access Mode Selection
Starting and End Dates
Starting and End Numbers
Starting Date
End Date
06/12/2006
06/12/2006
Start
Exit

6 Input File

EpsonFpMate is able to manage two different kind of file format: TXT or XML one.
When used Txt files, setting operations require also to define a separator character used to separate the different fields inside each record

Input file for must be composed by three sections:

- Input file specification
- Printer selection
- File body

6.1 Input File specification

First record of the file.
It allows to recognize what kind of file has to be managed

Type of Input File	Keyword
Fiscal receipt	printerFiscalReceipt
Daily closure (Zreport)	printerFiscalReport
Not fiscal receipt	printerNonFiscal
Command	printerCommand
Fiscal document	printerFiscalDocument

Example:

- Txt Mode: printerFiscalReceipt
- Xml Mode: In Xml mode that keyword represents the root

INPUT FILE SPECIFICATION				
Fiscal Receipt	**Not Fiscal Receipt**	**Daily Closure**	**Command**	**Fiscal Document**
printerFiscalReceipt	printerNonFiscal	printerFiscalReport	printerCommand	printerFiscalDocument
Message to Customer display Direct Activation of EpsonFp2.ocx method Setup of EpsonFp2.ocx properties Reading of EpsonFp2.ocx properties				
printRecItem	beginNonFiscal	printXReport	queryEjContent	beginFiscalDocument
printRecRefund	printNormal	printZTotReport	queryPrinterStatus	printFiscalDocumentLine
printRecVoidItem	printBarCode	printZReport		printFiscalDocAmount
printRecItemAdjustment	printGraphicCoupon			endFiscalDocument
printRecSubtotalAdjustment	endNonFiscal			printFiscalDocument
printRecSubtotal				
printRecMessage				
printBarCode				
printGraphicCoupon				
printRecTotal				

6.2 Printer selection

Second record of the file.
It allows to select the printer that must be addressed. The record is optional, it must be used if more than one printer has to be managed.

1. Txt Mode printer(FieldSeparator)(Num)
   Es: Printer; 3
   If we want to address printer N. 3 and the field separator character is semicolon.

2. Xml Mode
   <Printer Num=”3” />

6.3 File body

The content of the receipt body depends on the kind of file has to be managed: fiscal receipt, fiscal not receipt or others.

6.3.1 Records managed regardless the kind of file.

N	Kind of Record	keyword
1	Message to customer display	displayText
2	Direct Activation of EpsonFp2.ocx method	directIO
3	Setup of EpsonFp2.ocx properties	printSetProperty
4	Reading of EpsonFp2.ocx properties	printReadProperty

6.3.1.1 Text message for customer display

It sends to the customer display the text messages indicated.

- Txt Format (supposing that the separator used is the semicolon character)
  displayText; nOpe; DisplayTextMessage

- Xml Format
  <displayText Ope="nOpe" Text="DisplayTextMessage"/>

where:
- **displayText** is the keyword for the operation
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **DisplayTextMessage** represents the text that must be presented on the displayed. Max 40 characters.

6.3.1.2 Activation of EpsonFp2.ocx method

It performs a call of the selected EpsonFp2.ocx method using the argument list presented in the command line

- Txt Format (supposing that the separator used is the semicolon character)
  directIO; Method; Data0; Data 1; Data2; . . . , DataN

- Xml Format
  <directIO OcxMethod="Method" Arg0="Data0" Arg1=" Data1" Arg2="Data2" . . . ArgN= "DataN" />

where:
- **directIO** is the keyword for the operation
- **Method** is name of the EpsonFp2.ocx method to be used (refer to the list of methods in the Development Guide)
- **DataX** represents the X^{th} parameter to be passed to the method. The sequence order of the parameters (values or Arguments) must be consistent with the sequence presented in the related document. Obviously the number, the kind, the format and the meaning of the arguments depend upon the method called.

Note: Square brackets [ ] in the EpsonFp2.ocx development guide, indicate the optional parameters. They can present, partially present or missing in the list of parameters.

6.3.1.3 Setup of EpsonFp2.ocx properties

Allows settings of one EpsonFp2.ocx property.

- Txt Format (supposing that the separator used is the semicolon character)
  printSetProperty; Property; Value

- Xml Format
  <printSetProperty OcxProperty =“Property” SetUpValue=“Value” />

where:
- **printSetProperty** is the keyword for the operation
- **Property** is Name of the one EpsonFp2.ocx property to be set.
- **Value** is the value to be assigned to the selected property.

6.3.1.4 Reading of EpsonFp2.ocx properties

It gets the current value of the selected EpsonFp2.ocx property.

- Txt Format (supposing that the separator used is the semicolon character)
  printReadProperty; Property

- Xml Format
  <printReadProperty OcxProperty=“Property” />

where:
- **printReadProperty** is the keyword for the operation
- **Property** is Name of the one EpsonFp2.ocx property to be read.

6.3.2 File body for FISCAL RECEIPT

One or more records of different kind. It is composed by a set of the following kind of record:

N	Kind of Record	keyword
1	Sale	printRecItem
2	Good Return	printRecRefund
3	Sale Cancel	printRecVoidItem
4	Discount on the Sale	printRecItemAdjustment
5	Discount on the Subtotal	printRecSubtotalAdjustment
6	Print of Subtotal	printRecSubtotal
7	Additional Row or Description	printRecMessage
8	Print of a Bar Code	printBarCode
9	Print of a Graphic Coupon	printGraphicCoupon
10	Payment Section	printRecTotal

Note: Each fiscal receipt file requires, excluding the credit note receipt described in 6.4, must be compliant with the following conditions:
1. it must include at least one sale row
2. the total amount of the receipt can not be negative
3. the payment must be completed.

In other words it means that a fiscal receipt must be open and closed.

6.3.2.1 Sale

It prints on the receipt the sale of an item. If it is the first sale, it starts the new fiscal receipt.

3. Txt Format (supposing that the separator used is the semicolon character)
printRecItem; nOpe; SaleDescription; xx,xxx; yy,yy; nDep; Just

4. Xml Format
<printRecItem Ope="nOpe" Text="SaleDescription" Qty="xx,xxx" UnitCost="yyy,yy" Dep="nDep" Just="Just"/>

where:
5. **printRecItem** is the keyword for selling operations
6. **nOpe** is Id number of the operator (range 1 ÷ 12)
7. **SaleDescrition** is a description composed by max 32 characters
8. **xx,xxx** represents the quantity (range 0,001 ÷ 9999,999)
9. **yy,yy** represents the unit price (range 0,01 ÷ 9999999,999)
10. **nDdep** represents the department number (range 1 ÷ 40)
11. **Just** it is used to select the part of description (max chars)to present on the display (20 chars). If Just = 1 it will present the first 20 characters, if just = 2 it will present the last 20 characters.

6.3.2.2 Good return from the customer

Good Return from the customer (refund required), it prints on the receipt the return of an item that must be re-found to the customer.

- Txt Format (supposing that the separator used is the semicolon character)

printRecRefund; nOpe; RefundDescription; xx,xxx; yy,yy; nDep; Just

- Xml Format
  <printRecRefound Ope="nOpe" Text="RefundDescription" Qty="xx,xxx" UnitCost="yyy,yy" Dep="nDep" Just="Just"/>

where:
- **printRecRefund** is the keyword for refund operations
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **RefundDescrition** is a description composed by max 32 characters
- **xx,xxx** represents the quantity (range 0,001 ÷ 9999,999)
- **yy,yy** represents the unit price (range 0,01 ÷ 9999999,999)
- **Ndep** represents the department number (range 1 ÷ 40)
- **Just** it is used to select the part of description (max chars)to present on the display (20 chars). If Just = 1 it will present the first 20 characters, if just = 2 it will present the last 20 characters.

6.3.2.3 Sale cancel

It voids one of the previous sale operations.

- Txt Format (supposing that the separator used is the semicolon character)
  printRecVoidItem; nOpe; VoidItemDescription; xx,xxx; yy,yy; nDep; Just

- Xml Format
  <printRecVoidItem Ope="nOpe" Text="VoidItemDescription" Qty="xx,xxx" UnitCost="yyy,yy" Dep="nDep" Just="Just"/>

where:
- **printRecVoidItem** is the keyword for void operations
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **VoidItemDescrition** is a description composed by max 32 characters
- **xx,xxx** represents the quantity (range 0,001 ÷ 9999,999)
- **yy,yy** represents the unit price (range 0,01 ÷ 9999999,999)
- **nDep** represents the department number (range 1 ÷ 40)
- **Just** it is used to select the part of description (max chars)to present on the display (20 chars). If Just = 1 it will present the first 20 characters, if just = 2 it will present the last 20 characters.

6.3.2.4 Discount on the sale

It applies a discount on the previous sale operation.

Txt Format (supposing that the separator used is the semicolon character)
printRecItemAdjustment; nOpe; DiscountDescription; NU; yyy,yy; Ndep; Just

Xml Format
<printRecItemAdjustment Ope="nOpe" Text="DiscountDescription" Type="NU" Amount="yyy,yy" Dep="nDep" Just="Just"/>

where:
- **printRecItemAdjustment** is the keyword for the operation
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **DiscountDescrition** is a description composed by max 32 characters
- **NU** is an unused numeric field that can be left = 0
- **yyy,yy** represents the discount amount (range 0,01 ÷ 9999999,999)
- **nDep** represents the department number (range 1 ÷ 40)
- **Just** it is used to select the part of description (max chars)to present on the display (20 chars). If Just = 1 it will present the first 20 characters, if just = 2 it will present the last 20 characters.

6.3.2.5 Discount on the Subtotal

It applies a discount on the subtotal of the receipt.

- Txt Format (supposing that the separator used is the semicolon character)
  printRecSubtotalAdjustment; nOpe; DiscountDescription; Prn; yyy,yy; nDep; Just

- Xml Format
  <printRecSubtotalAdjustment Ope="nOpe" Text="DiscountDescription" Type="Prn" Amount="yyy,yy" Dep="nDep" Just="Just"/>

where: include
- **printRecSubtotalAdjustment** is the keyword for the operation
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **DiscountDescrition** is a description composed by max 32 characters
- **Prn** indicates if it is required a printout of the subtotal (Prn=1) or it is not required (Prn=2)
- **yyy,yy** represents the discount amount (range 0,01 ÷ 9999999,999)
- **nDep** represents the department number (range 1 ÷ 40) (NOT NEEDED for SUBTOTAL DISCOUNT
- **Just** it is used to select the part of description (max chars)to present on the display (20 chars). If Just = 1 it will present the first 20 characters, if just = 2 it will present the last 20 characters.

6.3.2.6 Print Subtotal

This Keyword gives the possibility to print out or to show on the display the content of SUBOTATAL register.

- Txt Format (supposing that the separator used is the semicolon character)
- printRecSubtotal; nOpe; prnOption

- Xml Format
- <printRecSubtotal Ope="**nOpe**" Type="**prnOption**" **/>**

where:
- **printRecSubtotal** is the keyword for the operation
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **prnOption** indicates the printing option of the subtotal: 0=Print on printer & Display; 1=Just print on printer; 2=Just print on display; 3= get subtotal amount

6.3.2.7 Additional Row Description

It applies additional header, row or description on the receipt

- Txt Format (supposing that the separator used is the semicolon character)
  printRecMessage; nOpe; Type; Num; Font; RowText

- Xml Format
  <printRecMessage Ope="NOpe" Type="Type" Index="Num" Font="Font" Text="RowText"/>

where:
- **printRecMessage** is the keyword for the operation
- **NOpe** is Id number of the operator (range 1 ÷ 12)
- **Type** defines the kind of row that must be added; 1=Additional header; 2= Additional row (before MF logotype); 3= Additional Promo (After MF logotype); 4=Additional Description (in the body of the receipt)
- **Num** indicates the number of the additional rows (Range 1 ÷ 9 for additional header (type 1); Range 1 ÷ 99 for additional promo and descriptions (types 2 and 3); No meaning for additional row (type 4)
- **Font** indicates the font that must be used for the row (1=Normal; 2=Bold, 3=Double height; 4=Bold Double height)
- **RowText** represents the text that must be printed. Max 40 characters. For type 4, it will be accepted up to 32 characters. Further characters will be ignored

Notes: Additional headers must be placed before the first selling operation.

6.3.2.8 Print of Bar Codes

It allows to print a Bar Code at the end of the receipt after the additional header, row or description on the receipt

- Txt Format (supposing that the separator used is the semicolon character)
  printBarCode; nOpe; Pos; BcWidth; BcHeight; HriEnabling; HriFont; CodeType; RowText

- Xml Format
  <printBarCode Ope="nOpe" Pos="Position" Width="BcWidth" Height ="BcHeight" Hri="HriEnabling" Font ="HriFont" Tipo="CodeType" Text="BcText"/>

where:
- **printBarCode** is the keyword for the operation
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **Position** defines the starting position from the left margin (range 0 ÷ 511)
- **BcWidth** indicates the width of the bar-code (range 1 ÷ 8).
- **BcHeight** indicates the height of the bar-code in print dots (range 1 ÷ 255).
- **HriEnabling** enables or disables the characters string of the barcode (DISABLED, BELOW, ABOVE, TWICE)
- **HriFont** indicates the font that must be used for the Hri string (FontA, FontB, FontC)
- **CodeType** indicates the barcode system to be printed. It must be chosen among the following (UPC-A, UPC-B, EAN13, EAN8, ITF, CODABAR, CODE93, CODE128)
- **RowText** represents the barcode string to be printed

Notes:

1) The character set and the string length are defined by the standard related to each bar code system.
2) The starting position, the length of the string and the width of the barcode must be defined according to the roll paper size. If there is no room enough for the row, the barcode will not be printed.
3) According to the standard, using CODE128, it must be selected one of the following character set:
   - CODE A, To select CODE A, the first two characters of the RowText must be "{A"
   - CODE B, To select CODE B, the first two characters of the RowText must be "{B"
   - CODE C, To select CODE C, the first two characters of the RowText must be "{C"

   In case of CODE128, the first two characters are not part of the bar code itself so they are not printed. They just allow the selection of the character set

Important Note: BarCode Systems CODE93 and CODE128 are managed by FP90 printers since Fw release 3.00

6.3.2.9 Print of Graphic Coupons

It allows to print one Graphic Coupon at the end of the receipt after the additional row, description or bar code on the receipt

- Txt Format (supposing that the separator used is the semicolon character)
  printGraphicCoupon; nOpe; FileName

- Xml Format
  <printGraphicCoupon Ope="nOpe" GraphicFile="FileName" />

where:
- **printGraphicCopupon** is the keyword for the operation
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **FileName** is the name of the graphic file to be printed

6.3.2.10 Payment section

One or more records. Usually it is composed by one record, more than one record means that the payment is composed by several partial payments.

- Txt Format
  printRecTotal; nOpe; PaymentDescription; yyy,yy; Mode; Index; Just

- Xml Format
  <printRecTotal Ope="nOpe" Text="PaymentDescription" Amount="yyy,yy" Type="Mode" Index="Ncard" Just="Just"/>

where:
- **printRecTotal** is the keyword for payment operations
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **PaymentDescrition** is a description composed by max 32 characters
- **yyy,yy** represents the amount (range 0,01 ÷ 9999999,999)
- **Mode** indicates the payment mode (0=Cash; 1=Check; 2=Credit/Credit Card; 3=Ticket)
- **Index** represents in case of Credit Cards or Ticket to select which kind of credit card or ticket has been used)
- **Just** it is used to select the part of description (max chars)to present on the display (20 chars). If Just = 1 it will present the first 20 characters, if just = 2 it will present the last 20 characters.

6.3.3 Forbidden words in text messages on the fiscal Receipt.

There are words that cannot be printed on fiscal receipt as additional text or description.
The forbidden words are:

- Totale
- Sconto
- Importo
- Contante

These words can not be printed even if the characters are separated by not alphabetic characters (es: Cont-ant#e; s?c'o"n%t o,).
These word should not be present in any text message or description.
Fiscal printer must refuse any message that requires to print any of the above words on fiscal receipt.
EpsonFpMate performs a pre-processing operation on text message for fiscal receipt. In this way it avoids that the printer refuses a message returning and error code because in the text there is one or more "forbidden words".
Pre processing operation consists in the following operation:

- Check the presence of one forbidden words.
- If the word has been found, replace the last character with an "*" (Asterisk).
- Check if the word is followed by one or more of the replaced character and in the case replace all of them.

Examples:
Instead of "Totale" EpsonFpMate prints "Total*"
Instead of "Importo" EpsonFpMate prints "Import*"
Instead of "Sconto" EpsonFpMate prints "Scont*"
Instead of "Contante" EpsonFpMate prints "Contant*"
Instead of "Totale eaiou" EpsonFpMate prints "Total* *aiou"
Instead of "Sconto o o#oao" EpsonFpMate prints "Scont* * *#oao"

6.4 File body for CREDIT NOTE FISCAL RECEIPT

Starting from Release 3.00, FP90 allows to manage receipt related only to good refund activity. It means that the customer receive back the money he paid when he bought it.
The operation allowed in a Credit Note (Nota di Credito) are almost the same than in a normal fiscal receipt and listed in the following table:

N	Kind of Record	keyword
1	Good Return	printRecRefund
2	Operation Cancel	printRecVoidItem
3	Discount on the Sale	printRecItemAdjustment
4	Discount on the Subtotal	printRecSubtotalAdjustment
5	Print of Subtotal	printRecSubtotal
6	Additional Row or Description	printRecMessage
7	Print of Bar Code	printBarCode
8	Print of a Graphic Coupon	printGraphicCoupon
9	Payment Section	printRecTotal

Note: Each credit note fiscal receipt file requires:
1. The open command
2. At least one Good Return row.
3. The payment row (closure)

6.4.1 Opening a Credit Note

To open a Credit Note it is required to send as first command a “printRecMessage”, type 4 that begins with the sentence “PRATICA DI RESO ”.
It is important to use capital font and separate the words by one blank character.

6.4.2 Closure of a “Nota Di Credito” FISCAL RECEIP

A Credit Note is closed by a payment command. It is ignored the amount indicated in the command line. The amount reported on the receipt is the one computed by the FP90 itself.

Important Note: “Nota di Credito” is managed by FP90 printers since Fw release 3.00

6.5 File body for NOT FISCAL RECEIPT

EpsonFpMate is able to manage the print of NOT fiscal receipt.
The keyword used to print out a NOT fiscal receipt is “printerNonFiscal”.

Following there is a list of command to be used in the NOT FISCAL file:

Type of NOT FISCAL RECEIPT command	Keyword
Open a NOT fiscal receipt	beginNonFiscal
Print a row	printNormal
Print of Bar Code	printBarCode
Print of a Graphic Coupon	printGraphicCoupon
Close a fiscal receipt	endNonFiscal

6.5.1 Begin Non fiscal receipt

Used to start a not fiscal receipt, the format is a follow:

- Txt Format
  beginNonFiscal;nOpe

- Xml Format
  <beginNonFiscal Ope=“nOpe” />

“nOpe” means the operator field (range 1-12)

6.5.2 Print Bar Code

Printout of bar codes in not fiscal receipts uses the same command with the same rules described in the Fiscal Receipt section.

6.5.3 Print Graphic Coupon

Printout of Graphic Coupon in not fiscal receipts uses the same command with the same rules described in the Fiscal Receipt section. The only difference is that while in fiscal receipt the Graphic Coupon can be printed only once at the end of the receipt, in not fiscal receipt it can be printed everywhere also more than once. It is also allowed to print several different graphic images.

6.5.4 Print row

To printout row in a NOT fiscal receipt, the keyword is printNormal, the format is as follow:

- Txt Format
  printNormal;nOpe;Font;RowText

- Xml Formato
  <printNormal Ope="nOpe" Font="Font" Text= "RowText"/>

- "nOpe" means the operator field (range 1-12)
- "Font" means the kind of font to be used: 1=normal, 2=bold, 3=double, 4 double and bold
- "RowText" means the row to be printed

6.5.5 Close Not fiscal receipt

To close a Not fiscal receipt must be used the following keyword: endNonFiscal, the format is as follow:

- Formato Txt
  endNonFiscal;NOpe
- Formato Xml
  <endNonFiscal Ope="nOpe" />

- "nOpe" means the operator field (range 1-12)

6.6 File body for DAILY CLOSURE

By means of file it is possible to manage the daily closure with print out of the Zreport.

Type of print out	Keyword
Normal daily closure	printZReport
With additional financial report	printZTotReport
X Daily closure (W/O reset)	printXReport

Example for Normal daily closure

- Txt Format

printZReport;NOpe

- Xml Format

<printZReport Ope="nOpe" />

- "nOpe" means the operator field (range 1-12)

Example for daily closure with financial report

- Xml Format

<printZTotReport Ope="nOpe" />

- Txt Format

printZTotReport; nOpe

- "nOpe" means the operator field (range 1-12)

Example for daily closure with financial report

- Xml Format

<printXReport Ope="nOpe" />

- Txt Format

printXReport; nOpe

- "nOpe" means the operator field (range 1-12)

6.7 File body for COMMANDS

EpsonFpMate is able to manage the print of NOT fiscal receipt.
The keyword used to print out a NOT fiscal receipt is “printerCommand”.

Following there is a list of command to be used in the COMMND file:

Type of command	Keyword
Collect Electronic Journal data and save in a file	queryEjContent
Collect Status Information of the printer	queryPrinterStatus

6.7.1 Collect Electronic Journal data

Used to save in a file the Electronic Journal data. The file will be written in the output directory.

- Txt Format
  queryEjContent;NOpe;dataType;stD;stM;stY;endD;endM;endY;file

- Xml Format
  < queryEjContent Ope=“nOpe” Type=“dataType” stDay=“stD” stMonth=“stM” stYear=“stY” endDay=“endD” endMonth=“endM” endYear=“endY” fileName=”file”/>

“nOpe” means the operator field (range 1-12).
“dataType” represents the kind of data to collect (0=All).
“stD” represents the starting day of the data to collect. (1 – 31)
“stM” represents the starting month of the data to collect. (1 – 12)
“stY” represents the starting year of the data to collect. (YYYY)
“endD” represents the end day of the data to collect. (1 – 31)
“endM” represents the end month of the data to collect. (1 – 12)
“endY” represents the end year of the data to collect. (YYYY)
“file” represents the file name .

6.7.2 Collect status information of the printer

Used to save in a file all the status information of the Printer. The file will be written in the output directory.

- Txt Format
  queryPrinterStatus;nOpe;file

- Xml Format
  < queryPrinterStatus Ope=“nOpe” fileName=”file”/>

“nOpe” means the operator field (range 1-12).
“file” represents the file name .

6.8 File body for FISCAL DOCUMENTS

It is composed by the following kind of record:

N	Kind of Record	keyword
1	Invoice open	beginFiscalDocument
2	Print an invoice text row	printFiscalDocumentLine
3	Print the invoice amount	printFiscalDocAmount
4	Invoice close	endFiscalDocument

6.8.1 Invoice Open

It prints on the receipt the sale of an item. If it is the first sale, it starts the new fiscal receipt.

- Txt Format (supposing that the separator used is the semicolon character)
  beginFiscalDocument; nOpe; invAmount

- Xml Format
  <printRecItem Ope="nOpe" Amount=" invAmount" />

where:
- **beginFiscalDocument** is the keyword for the command
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **invAmount** is the total amount of the invoice

6.8.2 Print Invoice Text Row

To print a text row inside a fiscal invoice, the keyword to be used is printFiscalDocumentLine.

- Txt Format
  printFiscalDocumentLine;nOpe;Font;RowText

- Xml Formato
  < printFiscalDocumentLine Ope="nOpe" Font="Font" Text= "RowText"/>

- **printFiscalDocumentLine** is the keyword for the command
- **nOpe** means the operator field (range 1-12)
- **Font** means the kind of font to be used: 1=normal, 2=bold, 3=double, 4 double and bold
- **RowText** means the row to be printed

6.8.3 Print Invoice Amount

It prints a row on the fiscal invoice with the total amount of the invoice sent in the invoice open message.

- Txt Format (supposing that the separator used is the semicolon character)
  printFiscalDocAmount; nOpe
- Xml Format
  < printFiscalDocAmount Ope="nOpe" />

where:
- **printFiscalDocAmount** is the keyword for the command
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **invAmount** is the total amount of the invoice

6.8.4 Invoice Close

It closes the fiscal invoice.

- Txt Format (supposing that the separator used is the semicolon character)
  endFiscalDocument; nOpe
- Xml Format
  < endFiscalDocument Ope="nOpe" />

where:
- **endFiscalDocument** is the keyword for the command
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **invAmount** is the total amount of the invoice

6.8.5 Management of Fiscal Invoices

Each "Fiscal Document" file must contain:
1) The open Invoice command (beginFiscalDocument) with the total amount of the invoice
2) The print Amount command (printFiscalDocAmount)
3) The close Invoice command (endFiscalDocument).

Text rows (printFiscalDocumentLine) can be included in the file without restriction regarding their number. The only requirement is that they must be included after the open Invoice and before the close Invoice commands.

Important Note: Managed only by Fiscal Printer FP81 and FP90II starting from Fw release 2.01

Examples:

```
printerFiscalDocument
Printer|1
beginFiscalDocument|1|1234,56
printFiscalDocumentLine|1|1|Invoice Text row 1 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocumentLine|1|1|Invoice Text row 2 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocumentLine|1|1|Invoice Text row 3 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocumentLine|1|1|Invoice Text row 4 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocumentLine|1|1|Invoice Text row 5 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocAmount|1
printFiscalDocumentLine|1|1|Invoice Text row 6 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocumentLine|1|1|Invoice Text row 7 ABCDEFGHIJKLMNOPQRSTU
endFiscalDocument|1
```

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerFiscalDocument>
  <Printer Num="1" />
  <beginFiscalDocument Ope="1" Amount="1234,56" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 1 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 2 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 3 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 4 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 5 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocAmount Ope="1" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 6 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 7 ABCDEFGHIJKLMNOPQRSTU" />
  <endFiscalDocument Ope="1" />
    </printerFiscalDocument>
```

After the close invoice command, the fiscal printer prints two copies of the invoice. EpsonFpMate returns a file that reports the result of the operation. If the result is OK, it reports also date, time and the invoice amount.

6.9 Alternative File body for FISCAL DOCUMENTS

It is composed by only one kind of record:

N	Kind of Record	keyword
1	Print invoice/receipt document	printFiscalDocument

It allows the following operations
- Print of a fiscal invoice following a fiscal receipt. It can be printed by the fiscal printer itself or by a "slip printer" connected to the fiscal printer.
- Print of a "double copy fiscal receipt". It can only be printed by means of a slip printer connected to the fiscal printer.

- Txt Format (supposing that the separator used is the semicolon character)
  printFiscalDocument; nOpe; docType; docNumber
- Xml Format
  < printFiscalDocument Ope="nOpe" Document="docType" Number="docNumber"/>

where:
- **printFiscalDocument** is the keyword for the command
- **nOpe** is Id number of the operator (range 1 ÷ 12)
- **docType** is the kind of document that must be printed (Invoice, Receipt)
- **docNumber** is the number of the document (range 1 ÷ 99999)

Important Notes:
1) Previous set-up of the fiscal printer is required to indicate the printer used for fiscal documents (Fiscal printer itself or external slip printer).
2) "Double copy fiscal receipts" can by printed only by the external slip printer. If it is missing, this kind of document cannot be printed.
3) The invoice can be printed only after the management of a "printerFiscalReceipt" file. The invoice will report all the data present in the previous receipt.
4) Top get a "Double copy fiscal receipt" printed by the slip printer it is required to send the printFiscalDocument command with the indication "Receipt" and then (immediately after) the "printerFiscalReceipt" file.
5) Invoice number must be **greater** than the last invoice number printed.
6) Number of "Double copy fiscal receipt" must be **greater** than the one printed on the last "Double copy fiscal receipt".
7) If a file contains a "printFiscalDocument" record, it cannot include the record types described in §6.8.
8) The file returned EpsonFpMtate does not report the amount. The value is set = 0.
9) Amount can be read before printing the invoice or after printing the "Double copy fiscal receipt".

Examples:

```
printerFiscalDocument
Printer|1
printFiscalDocument|1|Invoice|20

printerFiscalDocument
Printer|1
printFiscalDocument|1|Receipt|20
```

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerFiscalDocument>
  <Printer Num="1" />
  <printFiscalDocument Ope="1" Document="Invoice" Number="30" />
    </printerFiscalDocument>

<?xml version="1.0" encoding="utf-16" ?>
- <printerFiscalDocument>
  <Printer Num="1" />
  <printFiscalDocument Ope="1" Document="Receipt" Number="30" />
    </printerFiscalDocument>
```

7 Output File for Fiscal Receipt

When selected the option "**CreateOutputFile"** the application generates an output file.

In case of Xml files the name of the output file is achieved putting the characters "Out" before the name of the file.

In case of Txt files the name of the output file is achieved changing the extension with the extension "out".

Output file reports:

- Date and time;
- Printer number
- Id Cash Number
- Result of the operation (ok or ko)
- In case of Result = ok
  - Receipt Number
  - Total amount of the receipt
- In case of Result = ko
  - Error Code
  - Short description of the error

8 Examples of TXT file

8.1 Example of fiscal receipt

Following a simple file in TXT format used to print out fiscal receipt:

```
printerFiscalReceipt
Printer|1
printRecMessage|1|1|1|1|Prima Riga Aggiuntiva Tipo 1
printRecMessage|1|1|2|1|Seconda Riga Aggiuntiva Tipo 1
printRecMessage|1|4|1|1|Prima Riga Aggiuntiva Tipo 4
printRecItem|1|Vendita Reparto Iva 10%|1|100,0|15|1
printRecMessage|1|4|1|1|Seconda Riga Aggiuntiva Tipo 4
printRecItem|1|Vendita Reparto Iva 20%|1,234|100,0|15|1
printRecVoidItem|1|Vendita Reparto Iva 20%|1,234|100,0|15|1
printRecItem|1|Vendita Reparto Iva 4%|2,5|100,17|3|2
printRecMessage|1|4|1|1|Terza Riga Aggiuntiva Tipo 4
printRecRefund|2|Restituzione Imballo|10,000|5,0|15|1
printRecItem|3|Vendita Reparto Esente Iva|12,13|216,17|3|2
printRecItemAdjustment|3|Sconto su singolo prodotto|0|123,45|1|2
printRecMessage|1|2|1|1|Prima Riga Aggiuntiva Tipo 2
printRecMessage|1|2|2|1|Seconda Riga Aggiuntiva Tipo 2
printRecMessage|1|3|1|1|Prima Riga Aggiuntiva Tipo 3
printRecMessage|1|3|2|1|Seconda Riga Aggiuntiva Tipo 3
printRecItem|3|Vendita Reparto esente IVA|12,13|216,17|3|2
printRecSubtotalAdjustment|3|Sconto sul subtotal|1|300,12|3|2
printRecSubtotal|3|1
printBarCode|1|10|2|66|TWICE|FontA|CODE39|01234567ABCDEF
printRecTotal|4|Pagamento con carta di credito|6000,0|0|0|2
displayText|1|Customer Display    Printed Fisc Receipt
```

8.2 Example of Credit Note fiscal receipt

Following a simple file in TXT format used to print out fiscal receipt for money refund:

```
printerFiscalReceipt
Printer|1
printRecMessage|1|4|1|1|PRATICA DI RESO BETA 1
printRecRefund|2|Refund1 (good return)|10,000|5,0|1|1
printRecRefund|2|Refund2 (good return)|13,000|6,0|1|1
printRecRefund|2|Refund3 (good return)|16,000|7,0|1|1
printRecItemAdjustment|3|Discount applied to the product|0|1,23|1|2
printRecMessage|1|2|1|1|First Additional Row Type 2
printRecMessage|1|2|2|1|Second Additional Row Type 2
printRecMessage|1|3|1|1|First Additional Row Type 3
printRecMessage|1|3|2|1|Second Additional Row Type 3
printRecSubtotal|3|1
printBarCode|1|10|2|66|BELOW|FontA|CODE93|0123456789ABCDEFG
printRecTotal|4|Payment cash|0,0|0|0|2
displayText|1|Arrivederci e grazie
```

8.3 Example of Z report command file

```
printerFiscalReport
Printer|1
printZReport|1
displayText|1|Messaggio su    display cliente
```

8.4 Example of Not fiscal receipt with text and Bar Codes

```
printerNonFiscal
beginNonFiscal|1|
printNormal|1|1|Non Fiscal Receipt Row N. 1 Font 1
printNormal|1|2|
printNormal|1|2|BarCode System CODE39
printBarCode|1|10|2|66|TWICE|FontA|CODE39|01234567ABCD
printNormal|1|2|
printNormal|1|2|Non Fiscal Receipt Row N. 2 Font 2
printNormal|1|3|Non Fiscal Receipt Row N. 3 Font 3
printNormal|1|2|
printNormal|1|2|BarCode System CODE93
printBarCode|1|10|2|66|TWICE|FontA|CODE93|01234567ABCDEF
printNormal|1|2|
printNormal|1|4|Non Fiscal Receipt Row N. 4 Font 4
printNormal|1|1|Non Fiscal Receipt Row N. 5 Font 1
printNormal|1|2|Non Fiscal Receipt Row N. 6 Font 2
printNormal|1|2|
printNormal|1|2|BarCode System CODE128 CODE A
printBarCode|1|10|2|66|TWICE|FontA|CODE128|{A01234567ABCDEF
printNormal|1|2|
printNormal|1|3|Non Fiscal Receipt Row N. 7 Font 3
printNormal|1|2|
printNormal|1|2|BarCode System CODE128 CODE B
printBarCode|1|10|2|66|TWICE|FontA|CODE128|{B01234567ABCDEF
printNormal|1|2|
printNormal|1|4|Non Fiscal Receipt Row N. 8 Font 4
printNormal|1|2|
printNormal|1|2|BarCode System CODE128 CODE C
printBarCode|1|10|2|66|TWICE|FontA|CODE128|{C01234567ABCDEF
printNormal|1|2|
displayText|4|Customer Display    Printed Not Fisc Doc
endNonFiscal|1|
```

8.5 Example of Direct IO command file

```
printerCommand
directIO|opRepartoLight|1|1|Telefono cellulare|100,0|15|1
directIO|opAddRowDesc|1|4|1|1|IMEI:0001213540
directIO|opRepartoLight|1|2|Descrizone Prodotto|12,34|1|1
printReadProperty|LastOPCode
printReadProperty|LastOP
directIO|opSubTot|1
directIO|opPaymentLight|1|Pagamento contanti|300,00|0|0|2
```

8.6 Example of Fiscal Receipt With Graphic Coupon

```
printerFiscalReceipt
printRecItem|3|Telefono cellulare|1|100,0|15|1
printRecMessage|1|4|1|1|IMEI:0001213540
printGraphicCoupon|1|A90C.TMB
printRecSubtotal|3|1
printRecTotal|1|Pagamento contanti|300,00|0|0|2
```

8.7 Example of Non Fiscal Receipt With Graphic Coupon

```
printerNonFiscal
beginNonFiscal|1|
printGraphicCoupon|1|A90C.TMB
endNonFiscal|1|
```

8.8 Example of Fiscal Document (Invoice)

```
printerFiscalDocument
Printer|1
beginFiscalDocument|1|1234,56
printFiscalDocumentLine|1|1|Invoice Text row 1 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocumentLine|1|1|Invoice Text row 2 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocumentLine|1|1|Invoice Text row 3 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocumentLine|1|1|Invoice Text row 4 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocumentLine|1|1|Invoice Text row 5 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocAmount|1
printFiscalDocumentLine|1|1|Invoice Text row 6 ABCDEFGHIJKLMNOPQRSTU
printFiscalDocumentLine|1|1|Invoice Text row 7 ABCDEFGHIJKLMNOPQRSTU
endFiscalDocument|1
```

8.9 Example of Invoice Request Following a Fiscal Receipt

```
printerFiscalDocument
Printer|1
printFiscalDocument|1|Invoice|20
```

8.10 Example of Request for Double Copy Fiscal Receipt (Slip Printer)

printerFiscalDocument
Printer|1
printFiscalDocument|1|Receipt|20

8.11 Example of a Electronic Journal query file command

printerCommand
Printer|1
queryEjContent|1|0|1|10|2005|30|11|2006|DgfeSaved.txt

8.12 Example of a Printer Status query file command

printerCommand
Printer|1
queryPrinterStatus|1|Printer1_Status.txt

8.13 Example of a file command to send a display text message

printerCommand
Printer|1
displayText|1|00 ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789A

8.14 Example of Exit application file command

applicationExit

9 Examples of XML file

9.1 Example of fiscal receipt

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerFiscalReceipt>
  <Printer Num="1" />
  <displayText Ope="1" Text="="Messagge on       customer display" />
  <printRecMessage Ope="1" Text="First Additional Row Type 1" Type="1" Index="1" Font="1" />
  <printRecMessage Ope="1" Text="Second Additional Row Type 1" Type="1" Index="2" Font="1" />
  <beginFiscalReceipt Ope="1" />
  <printRecMessage Ope="1" Text="First Additional Row Type 4" Type="4" Index="1" Font="1" />
  <printRecItem Ope="1" Text="Selling Item 1 VAT 10%" Qty="1" UnitCost="100,0" Dep="15" Just="1" />
  <printRecMessage Ope="1" Text="Second Additional Row Type 4" Type="4" Index="1" Font="1" />
  <printRecItem Ope="2" Text="Selling Item 2 VAT 20%" Qty="1,234" UnitCost="100,0" Dep="15" Just="1" />
  <printRecVoidItem Ope="2" Text="Void selling Item2" Qty="1,234" UnitCost="100,0" Dep="15" Just="1" />
  <printRecItem Ope="4" Text="Selling Item 3 VAT 20%" Qty="2,5" UnitCost="100,17" Dep="3" Just="2" />
  <printRecMessage Ope="1" Text="Third Additional Row Type 4" Type="4" Index="1" Font="1" />
  <printRecRefund Ope="2" Text="Refound (good return)" Qty="10,000" UnitCost="5,0" Dep="15" Just="1" />
  <printRecItem Ope="3" Text="Selling Item 4 VAT 20%" Qty="12,13" UnitCost="216,17" Dep="3" Just="2" />
  <printRecItemAdjustment Ope="3" Text="Discount applied to the product" Type="1" Amount="123,45" Dep="3"
     Just="2" />
  <printRecMessage Ope="1" Text="First Additional Row Type 2" Type="2" Index="1" Font="1" />
  <printRecMessage Ope="1" Text="Second Additional Row Type 2" Type="2" Index="2" Font="1" />
  <printRecMessage Ope="1" Text="First Additional Row Type 3" Type="3" Index="1" Font="1" />
  <printRecMessage Ope="1" Text="Second Additional Row Type 3" Type="3" Index="2" Font="1" />
  <printRecItem Ope="3" Text="Selling Item 5 20%" Qty="12,13" UnitCost="216,17" Dep="3" Just="2" />
  <printRecSubtotalAdjustment Ope="3" Text="Discount applied to the subtotal" Type="1" Amount="300,12" Dep="3"
     Just="2" />
  <printRecSubtotal Ope="3" Type="1" />
  <printBarCode Ope="1" Position="10" Width="2" Height="66" Hri="TWICE" Font="FontA" Tipo="CODE39"
     Text="0123456789" />
  <printRecTotal Ope="4" Text="Payment cash" Amount="6000,0" Type="0" Index="0" Just="2" />
  <displayText Ope="1" Text="Messagge on       customer display" />
    </printerFiscalReceipt>
```

9.2 Example of Credit Note fiscal receipt

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerFiscalReceipt>
  <Printer Num="1" />
  <printRecMessage Ope="1" Text="PRATICA DI RESO BETA 1" Type="4" Index="1" Font="1" />
  <printRecRefund Ope="2" Text="Refund1 (good return)" Qty="10,000" UnitCost="5,0" Dep="1" Just="1" />
  <printRecRefund Ope="2" Text="Refund2 (good return)" Qty="13,000" UnitCost="6,0" Dep="2" Just="1" />
  <printRecRefund Ope="2" Text="Refund3 (good return)" Qty="16,000" UnitCost="7,0" Dep="3" Just="1" />
  <printRecItemAdjustment Ope="2" Text="Discount applied to the product" Type="1" Amount="1,23" Dep="3"
     Just="2" />
  <printRecMessage Ope="1" Text="First Additional Row Type 2" Type="2" Index="1" Font="1" />
  <printRecMessage Ope="1" Text="Second Additional Row Type 2" Type="2" Index="2" Font="1" />
  <printRecMessage Ope="1" Text="First Additional Row Type 3" Type="3" Index="1" Font="1" />
  <printRecMessage Ope="1" Text="Second Additional Row Type 3" Type="3" Index="2" Font="1" />
  <printRecSubtotal Ope="3" Type="1" />
  <printBarCode Ope="1" Pos="10" Width="2" Height="66" Hri="BELOW" Font="FontA" Tipo="CODE93"
     Text="0123456789ABCDEFG" />
  <printRecTotal Ope="4" Text="Payment cash" Amount="0,0" Type="0" Index="0" Just="2" />
  <displayText Ope="4" Text="Arrivederci e grazie" />
    </printerFiscalReceipt>
```

9.3 Example of Zreport file command

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerFiscalReport>
  <Printer Num="1" />
  <printZReport Ope="1" />
  <displayText Ope="1" Text=" Messagge on       customer display " />
  </printerFiscalReport>
```

9.4 Example of not fiscal receipt with Bar Codes

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerNonFiscal>
  <Printer Num="1" />
  <displayText Ope="1" Text="Messaggio su   display cliente" />
  <beginNonFiscal Ope="1" />
  <printNormal Ope="1" Font="1" Text="Non Fiscal Receipt Row N. 1 Font 1"
  <printNormal Ope="1" Font="2" Text="Non Fiscal Receipt Row N. 2 Font 2"
  <printNormal Ope="1" Font="3" Text="Non Fiscal Receipt Row N. 3 Font 3"
  <printBarCode Ope="1" Position="10" Width="2" Height="66" Hri="ABOVE" Font="FontA" Tipo="CODE39"
    Text="0123456789" />
  <printNormal Ope="1" Font="4" Text="Non Fiscal Receipt Row N. 4 Font 4"
  <displayText Ope="1" Text="Messaggio su   display cliente" />
  <printNormal Ope="1" Font="1" Text="Non Fiscal Receipt Row N. 5 Font 1"
  <printNormal Ope="1" Font="2" Text="Non Fiscal Receipt Row N. 6 Font 2"
  <printNormal Ope="1" Font="3" Text="Non Fiscal Receipt Row N. 7 Font 3"
   <printBarCode Ope="1" Position="10" Width="2" Height="66" Hri="BELOW" Font="FontB" Tipo="CODE128"
    Text="{A0123456789" />
<printNormal Ope="1" Font="4" Text="Non Fiscal Receipt Row N. 8 Font 4"
  <endNonFiscal Ope="1" />
  <displayText Ope="1" Text=" Messagge on       customer display " />
  </printerNonFiscal>
```

9.5 Example of Direct IO command file

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerCommand>
  <Printer Num="1" />
  <directIO OcxMethod="opRepartoLight" Arg0="1" Arg1="1" Arg2="Telefono cellulare" Arg3="100,0" Arg4="15"
    Arg5="1" />
  <directIO OcxMethod="opAddRowDesc" Arg0="1" Arg1="4" Arg2="1" Arg3="1" Arg4="IMEI:0001213540" />
  <directIO OcxMethod="opRepartoLight" Arg0="1" Arg1="1" Arg2="Descrizone Prodotto" Arg3="12,34" Arg4="1"
    Arg5="1" />
  <printReadProperty OcxProperty="LastOPCode" />
  <printReadProperty OcxProperty="LastOP" />
  <directIO OcxMethod="opSubTot" Arg0="0" />
  <directIO OcxMethod="opPaymentLight" Arg0="1" Arg1="Pagamento contanti" Arg2="300,00" Arg3="0" Arg4="0"
    Arg5="2" />
  </printerCommand>
```

9.6 Example of Fiscal Receipt with Graphic Coupon

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerFiscalReceipt>
  <Printer Num="1" />
  <printRecMessage Ope="1" Text="First Additional Row Type 1" Type="1" Index="1" Font="1" />
  <printRecItem Ope="1" Text="Selling Item 1 VAT 10%" Qty="1" UnitCost="100,0" Dep="15" Just="1" />
  <printRecMessage Ope="1" Text="Second Additional Row Type 4" Type="4" Index="1" Font="1" />
  <printRecItem Ope="1" Text="Selling Item 2 VAT 20%" Qty="1,234" UnitCost="100,0" Dep="15" Just="1" />
  <printRecMessage Ope="1" Text="First Additional Row Type 2" Type="2" Index="1" Font="1" />
  <printRecMessage Ope="1" Text="First Additional Row Type 3" Type="3" Index="1" Font="1" />
  <printRecItem Ope="3" Text="Selling Item 5 20%" Qty="12,13" UnitCost="216,17" Dep="3" Just="2" />
  <printRecSubtotalAdjustment Ope="1" Text="Discount applied to the subtotal" Type="1" Amount="300,12" Dep="3"
     Just="2" />
  <printRecSubtotal Ope="3" Type="1" />
  <printBarCode Ope="1" Pos="10" Width="2" Height="66" Hri="TWICE" Font="FontA" Tipo="CODE39"
     Text="01234567ABCDEF" />
  <printGraphicCoupon Ope="1" GraphicFile="A90C.TMB" />
  <printRecTotal Ope="4" Text="Payment cash" Amount="6000,0" Type="0" Index="0" Just="2" />
  <displayText Ope="4" Text="Customer Display Printed Fisc Receipt" />
    </printerFiscalReceipt>
```

9.7 Example of Non Fiscal Receipt with Graphic Image and Bar Codes

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerNonFiscal>
  <Printer Num="1" />
  <beginNonFiscal Ope="1" />
  <printNormal Ope="1" Font="1" Text="Non Fiscal Receipt Row N. 1 Font 1" />
  <printNormal Ope="1" Font="1" Text="" />
  <printNormal Ope="1" Font="1" Text="BarCode System CODE39" />
  <printBarCode Ope="1" Pos="10" Width="2" Height="66" Hri="ABOVE" Font="FontA" Tipo="CODE39"
     Text="0123456789ABCD" />
  <printNormal Ope="1" Font="1" Text="" />
  <printNormal Ope="1" Font="2" Text="Non Fiscal Receipt Row N. 2 Font 2" />
  <printNormal Ope="1" Font="3" Text="Non Fiscal Receipt Row N. 3 Font 3" />
  <printNormal Ope="1" Font="1" Text="" />
  <printNormal Ope="1" Font="1" Text="BarCode System CODE93" />
  <printBarCode Ope="1" Pos="10" Width="2" Height="66" Hri="BELOW" Font="FontA" Tipo="CODE93"
     Text="0123456789ABCD" />
  <printNormal Ope="1" Font="1" Text="" />
  <printNormal Ope="1" Font="4" Text="Non Fiscal Receipt Row N. 4 Font 4" />
  <printNormal Ope="1" Font="1" Text="Non Fiscal Receipt Row N. 5 Font 1" />
  <printNormal Ope="1" Font="2" Text="Non Fiscal Receipt Row N. 6 Font 2" />
  <printNormal Ope="1" Font="1" Text="" />
  <printNormal Ope="1" Font="1" Text="BarCode System CODE128 CODE A" />
  <printBarCode Ope="1" Pos="10" Width="2" Height="66" Hri="DISABLED" Font="FontA" Tipo="CODE128"
     Text="{A0123456789ABCD" />
  <printNormal Ope="1" Font="1" Text="" />
  <printNormal Ope="1" Font="3" Text="Non Fiscal Receipt Row N. 7 Font 3" />
  <printNormal Ope="1" Font="1" Text="" />
  <printNormal Ope="1" Font="1" Text="BarCode System CODE128 CODE B" />
  <printBarCode Ope="1" Pos="10" Width="2" Height="66" Hri="BELOW" Font="FontA" Tipo="CODE128"
     Text="{B0123456789ABCD" />
  <printNormal Ope="1" Font="1" Text="" />
  <printGraphicCoupon Ope="1" GraphicFile="A90C.TMB" />
  <printNormal Ope="1" Font="4" Text="Non Fiscal Receipt Row N. 8 Font 4" />
  <displayText Ope="4" Text="Customer Display Printed Not Fisc Doc" />
  <printNormal Ope="1" Font="1" Text="" />
  <printNormal Ope="1" Font="1" Text="BarCode System CODE128 CODE C" />
  <printBarCode Ope="1" Pos="10" Width="2" Height="66" Hri="TWICE" Font="FontA" Tipo="CODE128"
     Text="{C0123456789ABCD" />
  <displayText Ope="4" Text="Customer Display Printed Not Fisc Doc" />
  <endNonFiscal Ope="1" />
    </printerNonFiscal>
```

9.8 Example of Fiscal Document (Invoice)

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerFiscalDocument>
  <Printer Num="1" />
  <beginFiscalDocument Ope="1" Amount="1234,56" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 1 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 2 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 3 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 4 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 5 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocAmount Ope="1" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 6 ABCDEFGHIJKLMNOPQRSTU" />
  <printFiscalDocumentLine Ope="1" Font="1" Text="Invoice Text row 7 ABCDEFGHIJKLMNOPQRSTU" />
  <endFiscalDocument Ope="1" />
    </printerFiscalDocument>
```

9.9 Example of Invoice Request Following a Fiscal Receipt

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerFiscalDocument>
  <Printer Num="1" />
  <printFiscalDocument Ope="1" Document="Invoice" Number="30" />
    </printerFiscalDocument>
```

9.10 Example of Request for Double Copy Fiscal Receipt (Slip Printer)

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerFiscalDocument>
  <Printer Num="1" />
  <printFiscalDocument Ope="1" Document="Receipt" Number="30" />
    </printerFiscalDocument>
```

9.11 Example of queryEjContent file command

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerCommand>
  <Printer Num="1" />
  <queryEjContent Ope="1" dataType="0" stDay="1" stMonth="10" stYear="2005" endDay="30" endMonth="11"
     endYear="2006" fileName="DgfeSaved.txt" />
    </printerCommand>
```

9.12 Example of Printer Status query file command

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerCommand>
  <Printer Num="1" />
  <queryPrinterStatus Ope="1" fileName="Printer1_Status.txt" />
    </printerCommand>
```

9.13 Example of a file command to send a display text message

```
<?xml version="1.0" encoding="utf-16" ?>
- <printerCommand>
  <Printer Num="1" />
  <displayText Ope="1" Text="00 ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789A" />
  </printerCommand>
```

9.14 Example of Exit application file command

```
<?xml version="1.0" encoding="utf-16" ?>
<applicationExit />
```