CHAPTER 15

Fiscal Printer

This Chapter defines the Fiscal Printer device category.

Summary

Properties (UML attributes)

Common	*Type*	*Mutability*	*Version*	*May Use After*
AutoDisable:	*boolean*	{ read-write }	1.3	Not Supported
CapCompareFirmwareVersion:	*boolean*	{ read-only }	1.9	open
CapPowerReporting:	*int32*	{ read-only }	1.3	open
CapStatisticsReporting:	*boolean*	{ read-only }	1.8	open
CapUpdateFirmware:	*boolean*	{ read-only }	1.9	open
CapUpdateStatistics:	*boolean*	{ read-only }	1.8	open
CheckHealthText:	*string*	{ read-only }	1.3	open
Claimed:	*boolean*	{ read-only }	1.3	open
DataCount:	*int32*	{ read-only }	1.3	Not Supported
DataEventEnabled:	*boolean*	{ read-write }	1.3	Not Supported
DeviceEnabled:	*boolean*	{ read-write }	1.3	open & claim
FreezeEvents:	*boolean*	{ read-write }	1.3	open
OutputID:	*int32*	{ read-only }	1.3	open
PowerNotify:	*int32*	{ read-write }	1.3	open
PowerState:	*int32*	{ read-only }	1.3	open
State:	*int32*	{ read-only }	1.3	--
DeviceControlDescription:	*string*	{ read-only }	1.3	--
DeviceControlVersion:	*int32*	{ read-only }	1.3	--
DeviceServiceDescription:	*string*	{ read-only }	1.3	open
DeviceServiceVersion:	*int32*	{ read-only }	1.3	open
PhysicalDeviceDescription:	*string*	{ read-only }	1.3	open
PhysicalDeviceName:	*string*	{ read-only }	1.3	open

Properties (Continued)

Specific	*Type*	*Mutability*	*Version*	*May Use After*
CapAdditionalHeader:	*boolean*	{ read-only }	1.6	open
CapAdditionalLines:	*boolean*	{ read-only }	1.3	open
CapAdditionalTrailer:	*boolean*	{ read-only }	1.6	open
CapAmountAdjustment:	*boolean*	{ read-only }	1.3	open
CapAmountNotPaid:	**boolean**	**{ read-only }**	**1.3**	**Deprecated v1.11**
CapChangeDue:	*boolean*	{ read-only }	1.6	open
CapCheckTotal:	*boolean*	{ read-only }	1.3	open
CapCoverSensor: (1)	*boolean*	{ read-only }	1.3	open
CapDoubleWidth:	*boolean*	{ read-only }	1.3	open
CapDuplicateReceipt:	*boolean*	{ read-only }	1.3	open
CapEmptyReceiptIsVoidable:	*boolean*	{ read-only }	1.6	open
CapFiscalReceiptStation:	*boolean*	{ read-only }	1.6	open
CapFiscalReceiptType:	*boolean*	{ read-only }	1.6	open
CapFixedOutput:	*boolean*	{ read-only }	1.3	open
CapHasVatTable:	*boolean*	{ read-only }	1.3	open
CapIndependentHeader:	*boolean*	{ read-only }	1.3	open
CapItemList:	*boolean*	{ read-only }	1.3	open
CapJrnEmptySensor: (1)	*boolean*	{ read-only }	1.3	open
CapJrnNearEndSensor: (1)	*boolean*	{ read-only }	1.3	open
CapJrnPresent: (1)	*boolean*	{ read-only }	1.3	open
CapMultiContractor:	*boolean*	{ read-only }	1.6	open
CapNonFiscalMode:	*boolean*	{ read-only }	1.3	open
CapOnlyVoidLastItem:	*boolean*	{ read-only }	1.6	open
CapOrderAdjustmentFirst:	*boolean*	{ read-only }	1.3	open
CapPackageAdjustment:	*boolean*	{ read-only }	1.6	open
CapPercentAdjustment:	*boolean*	{ read-only }	1.3	open
CapPositiveAdjustment:	*boolean*	{ read-only }	1.3	open
CapPositiveSubtotalAdjustment	*boolean*	{ read-only }	1.11	open
CapPostPreLine:	*boolean*	{ read-only }	1.6	open
CapPowerLossReport:	*boolean*	{ read-only }	1.3	open
CapPredefinedPaymentLines:	*boolean*	{ read-only }	1.3	open
CapReceiptNotPaid:	*boolean*	{ read-only }	1.3	open
CapRecEmptySensor: (1)	*boolean*	{ read-only }	1.3	open
CapRecNearEndSensor: (1)	*boolean*	{ read-only }	1.3	open
CapRecPresent: (1)	*boolean*	{ read-only }	1.3	open

Properties (Continued)

Specific (continued)	*Type*	*Mutability*	*Version*	*May Use After*
CapRemainingFiscalMemory:	*boolean*	{ read-only }	1.3	open
CapReservedWord:	*boolean*	{ read-only }	1.3	open
CapSetCurrency:	*boolean*	{ read-only }	1.6	open
CapSetHeader:	*boolean*	{ read-only }	1.3	open
CapSetPOSID:	*boolean*	{ read-only }	1.3	open
CapSetStoreFiscalID:	*boolean*	{ read-only }	1.3	open
CapSetTrailer:	*boolean*	{ read-only }	1.3	open
CapSetVatTable:	*boolean*	{ read-only }	1.3	open
CapSlpEmptySensor: (1)	*boolean*	{ read-only }	1.3	open
CapSlpFiscalDocument:	*boolean*	{ read-only }	1.3	open
CapSlpFullSlip: (1)	*boolean*	{ read-only }	1.3	open
CapSlpNearEndSensor: (1)	*boolean*	{ read-only }	1.3	open
CapSlpPresent: (1)	*boolean*	{ read-only }	1.3	open
CapSlpValidation:	*boolean*	{ read-only }	1.3	open
CapSubAmountAdjustment:	*boolean*	{ read-only }	1.3	open
CapSubPercentAdjustment:	*boolean*	{ read-only }	1.3	open
CapSubtotal:	*boolean*	{ read-only }	1.3	open
CapTotalizerType:	*boolean*	{ read-only }	1.6	open
CapTrainingMode:	*boolean*	{ read-only }	1.3	open
CapValidateJournal:	*boolean*	{ read-only }	1.3	open
CapXReport:	*boolean*	{ read-only }	1.3	open
ActualCurrency:	*int32*	{ read-only }	1.6	open, claim, & enable
AdditionalHeader:	*string*	{ read-write }	1.6	open, claim, & enable
AdditionalTrailer:	*string*	{ read-write }	1.6	open, claim, & enable
AmountDecimalPlaces:	*int32*	{ read-only }	1.3	open, claim, & enable
AsyncMode:	*boolean*	{ read-write }	1.3	open
ChangeDue:	*string*	{ read-write }	1.6	open
CheckTotal:	*boolean*	{ read-write }	1.3	open
ContractorId:	*int32*	{ read-write }	1.6	open, claim, & enable
CountryCode:	*int32*	{ read-only }	1.3	open, claim, & enable
CoverOpen: (1)	*boolean*	{ read-only }	1.3	open, claim, & enable
DateType:	*int32*	{ read-write }	1.6	open, claim, & enable
DayOpened:	*boolean*	{ read-only }	1.3	open, claim, & enable
DescriptionLength:	*int32*	{ read-only }	1.3	open
DuplicateReceipt:	*boolean*	{ read-write }	1.3	open
ErrorLevel:	*int32*	{ read-only }	1.3	open

Properties (Continued)

Specific (continued)	*Type*	*Mutability*	*Version*	*May Use After*
ErrorOutID:	*int32*	{ read-only }	1.3	open, claim, & enable
ErrorState:	*int32*	{ read-only }	1.3	open
ErrorStation:	*int32*	{ read-only }	1.3	open
ErrorString:	*string*	{ read-only }	1.3	open
FiscalReceiptStation:	*int32*	{ read-write }	1.6	open, claim, & enable
FiscalReceiptType:	*int32*	{ read-write }	1.6	open, claim, & enable
FlagWhenIdle: (1)	*boolean*	{ read-write }	1.3	open
JrnEmpty:	*boolean*	{ read-only }	1.3	open, claim, & enable
JrnNearEnd:	*boolean*	{ read-only }	1.3	open, claim, & enable
MessageLength:	*int32*	{ read-only }	1.3	open
MessageType:	*int32*	{ read-write }	1.6	open
NumHeaderLines:	*int32*	{ read-only }	1.3	open
NumTrailerLines:	*int32*	{ read-only }	1.3	open
NumVatRates:	*int32*	{ read-only }	1.3	open
PostLine:	*string*	{ read-write }	1.6	open, claim, & enable
PredefinedPaymentLines:	*string*	{ read-only }	1.3	open
PreLine:	*string*	{ read-write }	1.6	open, claim, & enable
PrinterState:	*int32*	{ read-only }	1.3	open, claim, & enable
QuantityDecimalPlaces:	*int32*	{ read-only }	1.3	open, claim, & enable
QuantityLength:	*int32*	{ read-only }	1.3	open, claim, & enable
RecEmpty: (1)	*boolean*	{ read-only }	1.3	open, claim, & enable
RecNearEnd: (1)	*boolean*	{ read-only }	1.3	open, claim, & enable
RemainingFiscalMemory:	*int32*	{ read-only }	1.3	open, claim, & enable
ReservedWord:	*string*	{ read-only }	1.3	open
SlpEmpty: (1)	*boolean*	{ read-only }	1.3	open, claim, & enable
SlpNearEnd: (1)	*boolean*	{ read-only }	1.3	open, claim, & enable
SlipSelection:	*int32*	{ read-write }	1.3	open, claim, & enable
TotalizerType:	*int32*	{ read-write }	1.6	open, claim, & enable
TrainingModeActive:	*boolean*	{ read-only }	1.3	open, claim, & enable

Methods (UML operations)

Common

Name	*Version*
open (logicalDeviceName: *string* **):** **void { raises-exception }**	1.3
close ( ): **void { raises-exception, use after open }**	1.3
claim (timeout: *int32* **):** **void { raises-exception, use after open }**	1.3
release ( ): **void { raises-exception, use after open, claim }**	1.3
checkHealth (level: *int32* **):** **void { raises-exception, use after open, claim, enable }**	1.3
clearInput ( ): **void { }**	***Not supported***
clearInputProperties ( ): **void { }**	***Not supported***
clearOutput ( ): **void { raises-exception, use after open, claim }**	1.3
directIO (command: *int32,* **inout data:** *int32,* **inout obj:** *object* **):** **void { raises-exception, use after open }**	1.3
compareFirmwareVersion (firmwareFileName: *string,* **out result:** *int32* **):** **void { raises-exception, use after open, claim, enable }**	1.9
resetStatistics (statisticsBuffer: *string* **):** **void { raises-exception, use after open, claim, enable }**	1.8
retrieveStatistics (inout statisticsBuffer: *string* **):** **void { raises-exception, use after open, claim, enable }**	1.8
updateFirmware (firmwareFileName: *string* **):** **void { raises-exception, use after open, claim, enable }**	1.9
updateStatistics (statisticsBuffer: *string* **):** **void { raises-exception, use after open, claim, enable }**	1.8

Specific - Presetting Fiscal

setCurrency (newCurrency: *int32* **):** **void { raises-exception, use after open, claim, enable }**	1.6
setDate (date: *string* **):** **void { raises-exception, use after open, claim, enable }**	1.3
setHeaderLine (lineNumber: *int32*, **text:** *string*, **doubleWidth:** *boolean* **):** **void { raises-exception, use after open, claim, enable }**	1.3
setPOSID (POSID: *string*, **cashierID:** *string* **):** **void { raises-exception, use after open, claim, enable }**	1.3
setStoreFiscalID (ID: *string* **):** **void { raises-exception, use after open, claim, enable }**	1.3
setTrailerLine (lineNumber: *int32*, **text:** *string*, **doubleWidth:** *boolean* **):** **void { raises-exception, use after open, claim, enable }**	1.3
setVatTable ( ): **void { raises-exception, use after open, claim, enable }**	1.3

Specific - Fiscal Receipt

setVatValue (vatID: *int32*, **vatValue:** *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

beginFiscalReceipt (printHeader: *boolean* **):** 1.3
void { raises-exception, use after open, claim, enable }

endFiscalReceipt (printHeader: *boolean* **):** 1.3
void { raises-exception, use after open, claim, enable }

printDuplicateReceipt ( ): 1.3
void { raises-exception, use after open, claim, enable }

printRecCash (amount: *currency* **):** 1.6
void { raises-exception, use after open, claim, enable }

printRecItem (description: *string*, **price:** *currency*, **quantity:** *int32*, 1.3
vatInfo: *int32*, **unitPrice:** *currency*, **unitName:** *string* **):**
void { raises-exception, use after open, claim, enable }

printRecItemVoid (description: *string*, **price:** *currency*, **quantity:** *int32*, 1.11
vatInfo: *int32*, **unitPrice:** *currency*, **unitName:** *string* **):**
void { raises-exception, use after open, claim, enable }

printRecItemAdjustment (adjustmentType: *int32*, **description:** *string*, 1.3
amount: *currency*, **vatInfo:** *int32* **):**
void { raises-exception, use after open, claim, enable }

printRecItemAdjustmentVoid (adjustmentType: *int32*, **description:** *string*, 1.11
amount: *currency*, **vatInfo:** *int32* **):**
void { raises-exception, use after open, claim, enable }

printRecItemFuel (description: *string*, **price:** *currency*, **quantity:** *int32*, 1.6
vatInfo: *int32*, **unitPrice:** *currency*, **unitName:** *string*,
specialTax: *currency*, **specialTaxName:** *string* **):**
void { raises-exception, use after open, claim, enable }

printRecItemFuelVoid (description: *string*, **price:** *currency*, **vatInfo:** *int32*, 1.6
specialTax: *currency* **):**
void { raises-exception, use after open, claim, enable }

printRecItemRefund (description: *string*, **amount:** *currency*, **quantity:** *int32*, **1.12**
vatInfo: *int32*, **unitAmount:** *currency*, **unitName:** *string* **):**
void { raises-exception, use after open, claim, enable }

printRecItemRefundVoid (description: *string*, **amount:** *currency*, **quantity:** **1.12**
int32, **vatInfo:** *int32*, **unitAmount:** *currency*, **unitName:** *string* **):**
void { raises-exception, use after open, claim, enable }

printRecMessage (message: *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

printRecNotPaid (description: *string*, **amount:** *currency* **):** 1.3
void { raises-exception, use after open, claim, enable }

printRecPackageAdjustment (adjustmentType: *int32*, **description:** *string*, 1.6
vatAdjustment: *string* **):**
void { raises-exception, use after open, claim, enable }

printRecPackageAdjustVoid (adjustmentType: *int32*, 1.6
vatAdjustment: *string* **):**
void { raises-exception, use after open, claim, enable }

printRecRefund (description: *string*, **amount:** *currency*, **vatInfo:** *int32* **):** 1.3
void { raises-exception, use after open, claim, enable }

printRecRefundVoid (description: *string*, **amount:** *currency*, **vatInfo:** *int32* **):** 1.6
void { raises-exception, use after open, claim, enable }

printRecSubtotal (amount: *currency* **):** 1.3
void { raises-exception, use after open, claim, enable }

printRecSubtotalAdjustment (adjustmentType: *int32*, **description:** *string*, **amount:** *currency* **):** 1.3
void { raises-exception, use after open, claim, enable }

printRecSubtotalAdjustVoid (adjustmentType: *int32*, **amount:** *currency* **):** 1.6
void { raises-exception, use after open, claim, enable }

printRecTaxID (taxId: *string* **):** 1.6
void { raises-exception, use after open, claim, enable }

printRecTotal (total: *currency*, **payment:** *currency*, **description:** *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

printRecVoid (description: *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

printRecVoidItem (description: ***string***, **amount:** ***currency***, **quantity:** ***int32***, **adjustmentType:** ***int32***, **adjustment:** ***currency***, **vatInfo:** ***int32*** **):** **1.3 Deprecated v1.11**
void { raises-exception, use after open, claim, enable }

Specific - Fiscal Document

beginFiscalDocument (documentAmount: *int32* **):** 1.3
void { raises-exception, use after open, claim, enable }

endFiscalDocument ( ): 1.3
void { raises-exception, use after open, claim, enable }

printFiscalDocumentLine (documentLine: *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

Specific - Item Lists

beginItemList (vatID: *int32* **):** 1.3
void { raises-exception, use after open, claim, enable }

endItemList ( ): 1.3
void { raises-exception, use after open, claim, enable }

verifyItem (itemName: *string*, **vatID:** *int32* **):** 1.3
void { raises-exception, use after open, claim, enable }

Specific - Fiscal Reports

printPeriodicTotalsReport (date1: *string*, **date2:** *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

printPowerLossReport ( ): 1.3
void { raises-exception, use after open, claim, enable }

printReport (reportType: *int32*, **startNum:** *string*, **endNum:** *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

printXReport ( ): 1.3
void { raises-exception, use after open, claim, enable }

printZReport ( ): 1.3
void { raises-exception, use after open, claim, enable }

Specific - Slip Insertion

beginInsertion (timeout: *int32* **):** 1.3
void { raises-exception, use after open, claim, enable } (1)

beginRemoval (timeout: *int32* **):** 1.3
void { raises-exception, use after open, claim, enable } (1)

endInsertion ( ): 1.3
void { raises-exception, use after open, claim, enable } (1)

endRemoval ( ): 1.3
void { raises-exception, use after open, claim, enable } (1)

Specific - Non-Fiscal

beginFixedOutput (station: *int32*, **documentType:** *int32* **):** 1.3
void { raises-exception, use after open, claim, enable }

beginNonFiscal ( ): 1.3
void { raises-exception, use after open, claim, enable }

beginTraining ( ): 1.3
void { raises-exception, use after open, claim, enable }

endFixedOutput ( ): 1.3
void { raises-exception, use after open, claim, enable }

endNonFiscal ( ): 1.3
void { raises-exception, use after open, claim, enable }

endTraining ( ): 1.3
void { raises-exception, use after open, claim, enable }

printFixedOutput (documentType: *int32*, **lineNumber:** *int32*, **data:** *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

printNormal (station: *int32*, **data:** *string* **):** 1.3
void { raises-exception, use after open, claim, enable } (1)

Specific - Data Requests

getData (dataItem: *int32*, **inout optArgs:** *int32*, **inout data:** *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

getDate (inout date: *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

getTotalizer (vatID: *int32*, **optArgs:** *int32*, **inout data:** *string* **):** 1.3
void { raises-exception, use after open, claim, enable }

getVatEntry (vatID: *int32*, **optArgs:** *int32*, **inout vatRate:** *int32* **):** 1.3
void { raises-exception, use after open, claim, enable }

Specific - Error Corrections

clearError ( ): 1.3
void { raises-exception, use after open, claim, enable }

resetPrinter ( ): 1.3
void { raises-exception, use after open, claim, enable }

Events (UML interfaces)

Name	*Type*	*Mutability*	*Version*
upos::events::DataEvent		***Not Supported***	
upos::events::DirectIOEvent			1.3
EventNumber:	*int32*	{ read-only }	
Data:	*int32*	{ read-write }	
Obj:	*object*	{ read-write }	
upos::events::ErrorEvent			1.3
ErrorCode:	*int32*	{ read-only }	
ErrorCodeExtended:	*int32*	{ read-only }	
ErrorLocus:	*int32*	{ read-only }	
ErrorResponse	*int32*	{ read-write }	
upos::events::OutputCompleteEvent			1.3
OutputID:	*int32*	{ read-only }	
upos::events::StatusUpdateEvent			1.3
Status:	*int32*	{ read-only }	

Note:

(1) Properties and methods marked with **(1)** are adapted from the POS Printer device.

General Information

The Fiscal Printer programmatic name is “FiscalPrinter”.

The Fiscal Printer Control does not attempt to encapsulate a generic graphics printer. Rather, for performance and ease of use considerations, the interfaces are defined to directly control the normal printer functions.

Since fiscal rules differ between countries, this interface tries to generalize the common requirements at the maximum extent specifications. This interface is based upon the fiscal requirements of the following countries, but it may fit the needs of other countries as well:

- Brazil
- Bulgaria
- Greece
- Hungary
- Italy
- Poland
- Romania
- Russia
- Turkey
- Czech Republic
- Ukraine
- Sweden

The Fiscal Printer model defines three stations with the following general uses:

- **Journal** Used for simple text to log transaction and activity information. Kept by the store for audit and other purposes.
- **Receipt** Used to print transaction information. It is mandatory to give a printed fiscal receipt to the customer. Also often used for store reports. Contains either a knife to cut the paper between transactions, or a tear bar to manually cut the paper.
- **Slip** Used to print information on a form. Usually given to the customer.

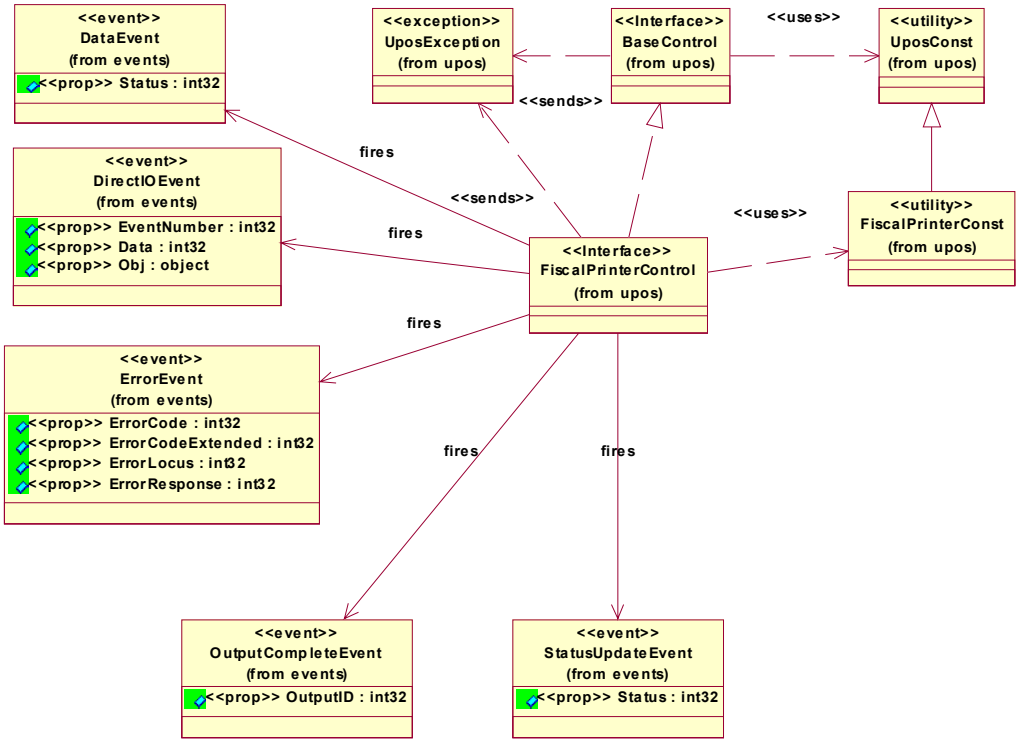
  The **Slip** station is also used to print “validation” information on a form. The form type is typically a check or credit card slip.
  It may also be used to print complete transaction information instead of printing it on the receipt station.

Sometimes, limited forms-handling capability is integrated with the receipt or journal station to permit validation printing. Often this limits the number of print lines, due to the station’s forms-handling throat depth. The Fiscal Printer Control nevertheless addresses this printer functionality as a slip station.

Configuration and initialization of the fiscal memory of the Fiscal Printer are not covered in this specification. These low-level operations must be performed by authorized technical assistance personnel.

Fiscal Printer Class Diagram

The following diagram shows the relationships between the Fiscal Printer classes.



General Requirements

Fiscal Printers do not simply print text similar to standard printers. They are used to monitor and memorize all fiscal information about a sale transaction. A Fiscal Printer has to accumulate totals, discounts, number of canceled receipts, taxes, etc. and has to store this information in different totalizers, counters and the fiscal memory. In order to perform these functions, it is not sufficient to send unformatted strings of text to the Fiscal Printer; there is a need to separate each individual field in a receipt line item, thus differentiating between descriptions, prices and discounts. Moreover, it is necessary to define different printing commands for each different sale functionality (such as refund, item or void).

Fiscal rules are different among countries. This interface tries to generalize these requirements by summarizing the common requirements. Fiscal law requires that:

- Fiscal receipts must be printed and given to the customer.
- Fiscal Printers must be equipped with memory to store daily totals. Each receipt line item must increment totals registers and, in most countries (Greece, Poland, Brazil, Hungary, Romania, Bulgaria, Russia and Turkey) tax registers as well.
- Discounts, canceled items and canceled receipts must increment their associated registers on the Fiscal Printer.
- Fiscal Printer must include a clock to store date and time information relative to each single receipt.
- Each fiscal receipt line item is normally printed both on the receipt and on the journal (Italy, Greece, Poland), but as an extension it can also be printed on the slip and journal.
- After a power failure (or a power off) the Fiscal Printer must be in the same state as it was before this event occurred. This implies that care must be taken in managing the Fiscal Printer status and that power failure events must be managed by the application. In some countries, a power failure must be logged and a report must be printed.

Fiscal Printer Modes

According to fiscal rules, it is possible for a Fiscal Printer to also offer functionality beyond the required fiscal printing mode. These additional modes are optional and may or may not be present on any particular Fiscal Printer.

There are three possible Fiscal Printer modes:

- **Fiscal:** This is the only required mode for a Fiscal Printer. In this mode the application has access to all the methods needed to manage a sale transaction and to print a fiscal receipt. It is assumed that any lines printed to the receipt station while in fiscal mode are also printed on the journal station.
- **Training:** In this mode, the Fiscal Printer is used for training purposes (such as cashier training). In this mode, the Fiscal Printer will accept fiscal commands but the Fiscal Printer will indicate on each receipt or document that the transaction is not an actual fiscal transaction. The Fiscal Printer will not update any of its internal fiscal registers while in training mode. Such printed receipts are usually marked as “training” receipts by Fiscal Printers. **CapTrainingMode** will be true if the Fiscal Printer supports training mode, otherwise it is false.
- **Non-Fiscal:** In this mode the Fiscal Printer can be used to print simple text on the receipt station (echoed on the journal station) or the slip station. The Fiscal Printer will print some additional lines along with the application requested output to indicate that this output is not of a fiscal nature. Such printed receipts are usually marked as “non-fiscal” receipts by Fiscal Printers. **CapNonFiscalMode** will be true if the Fiscal Printer supports non-fiscal printing, otherwise it is false.

Model *Updated in Release 1.12*

The Fiscal Printer follows the output model for devices, with some enhancements:

- Most methods are always performed synchronously. Synchronous methods will throw a UposException if asynchronous output is outstanding.
- The following methods are performed either synchronously or asynchronously, depending on the value of the **AsyncMode** property:

  printFiscalDocumentLine
  printFixedOutput
  printNormal
  printRecCash
  printRecItem
  printRecItemVoid
  printRecItemAdjustment
  printRecItemAdjustmentVoid
  printRecItemFuel
  printRecItemFuelVoid
  printRecItemRefund
  printRecItemRefundVoid
  printRecMessage
  printRecNotPaid
  printRecPackageAdjustment
  printRecPackageAdjustVoid
  printRecRefund
  printRecRefundVoid
  printRecSubtotal
  printRecSubtotalAdjustment
  printRecSubtotalAdjustVoid
  printRecTaxID
  printRecTotal
  printRecVoid

  When **AsyncMode** is false, then these methods print synchronously.

  When **AsyncMode** is true, then these methods operate as follows:

  - The Device buffers the request in program memory, for delivery to the Physical Device as soon as the Physical Device can receive and process it, sets the **OutputID** property to an identifier for this request, and returns as soon as possible. When the device completes the request successfully, the **OutputCompleteEvent** is enqueued. A parameter of this event contains the **OutputID** of the completed request.

  Asynchronous Fiscal Printer methods will not throw a UposException due to a printing problem, such as out of paper or Fiscal Printer fault. These errors will only be reported by an **ErrorEvent**. A UposException is thrown only if the Fiscal Printer is not claimed and enabled, a parameter is invalid, or the request cannot be enqueued. The first two error cases are due to an application error, while the last is a serious system resource exception.

- If an error occurs while performing an asynchronous request, an **ErrorEvent** is enqueued. The **ErrorStation** property is set to the station or stations that were printing when the error occurred. The **ErrorLevel**, **ErrorString** and **ErrorState** and **ErrorOutID** properties are also set.

The event handler may call synchronous print methods (but not asynchronous methods), then can either retry the outstanding output or clear it.

- Asynchronous output is performed on a first-in first-out basis.
- All buffered output data, including all asynchronous output, may be deleted by calling **clearOutput**. **OutputCompleteEvent**s will not be delivered for cleared output. This method also stops any output that may be in progress (when possible).
- The property **FlagWhenIdle** may be set to cause a **StatusUpdateEvent** to be enqueued when all outstanding outputs have finished, whether successfully or because they were cleared.

Error Model *Updated in Release 1.13*

The Fiscal Printer error reporting model is as follows:

- Most of the Fiscal Printer error conditions are reported by setting the UposException's (or ErrorEvent's) *ErrorCode* to E_EXTENDED and then setting *ErrorCodeExtended* to one of the following:

  EFPTR_COVER_OPEN
  The Fiscal Printer cover is open.

  EFPTR_JRN_EMPTY
  The journal station has run out of paper.

  EFPTR_REC_EMPTY
  The receipt station has run out of paper.

  EFPTR_SLP_EMPTY
  The slip station has run out of paper.

  EFPTR_SLP_FORM
  A form is still present in the document station even though it should have been removed by the last action.

  EFPTR_MISSING_DEVICES
  Some of the other devices that according to the local fiscal legislation are to be connected are missing. In some countries in order to use a Fiscal Printer a full set of peripheral devices are to be connected to the POS (such as cash drawer and customer display). In case one of these devices is not present, sales are not allowed.

  EFPTR_WRONG_STATE
  The requested method could not be executed in the Fiscal Printer's current state.

  EFPTR_TECHNICAL_ASSISTANCE
  The Fiscal Printer has encountered a severe error condition. Calling for Fiscal Printer technical assistance is required.

  EFPTR_CLOCK_ERROR
  The Fiscal Printer's internal clock has failed.

EFPTR_FISCAL_MEMORY_FULL
The Fiscal Printer's fiscal memory has been exhausted.

EFPTR_FISCAL_MEMORY_DISCONNECTED
The Fiscal Printer's fiscal memory has been disconnected.

EFPTR_FISCAL_TOTALS_ERROR
The Grand Total in working memory does not match the one in the EPROM.

EFPTR_BAD_ITEM_QUANTITY
The quantity parameter is invalid.

EFPTR_BAD_ITEM_AMOUNT
The amount parameter is invalid.

EFPTR_BAD_ITEM_DESCRIPTION
The description parameter is either too long, contains illegal characters or contains a reserved word.

EFPTR_RECEIPT_TOTAL_OVERFLOW
The receipt total has overflowed.

EFPTR_BAD_VAT
The vat parameter is invalid.

EFPTR_BAD_PRICE
The price parameter is invalid.

EFPTR_BAD_DATE
The date parameter is invalid.

EFPTR_NEGATIVE_TOTAL
The Fiscal Printer's computed total or subtotal is less than zero.

EFPTR_WORD_NOT_ALLOWED
The description contains the reserved word.

EFPTR_BAD_LENGTH
The length of the string to be printed as post or pre line is too long.

EFPTR_MISSING_SET_CURRENCY
The Fiscal Printer is expecting the activation of a new currency.

EFPTR_DAY_END_REQUIRED
The completion of the fiscal day is required.

Other Fiscal Printer errors are reported by setting the exception's (or **ErrorEvent**'s) *ErrorCode* to E_FAILURE or another error status. These failures are typically due to a Fiscal Printer fault or jam, or to a more serious error.

Release 1.8 Additional Model Clarifications

While the Fiscal Printer is enabled, the printer state is monitored, and changes are reported to the application. Most Fiscal Printer statuses are reported by both firing a **StatusUpdateEvent** and by updating a printer property. Statuses, as defined in the later properties and events sections, are:

Prior to Release 1.8

StatusUpdateEvent	Property
FPTR_SUE_COVER_OPEN	**CoverOpen** = true
FPTR_SUE_COVER_OK	**CoverOpen** = false
FPTR_SUE_JRN_EMPTY	**JrnEmpty** = true
FPTR_SUE_JRN_NEAREMPTY	**JrnNearEnd** = true
FPTR_SUE_JRN_PAPEROK	**JrnEmpty** = **JrnNearEnd** = false
FPTR_SUE_REC_EMPTY	**RecEmpty** = true
FPTR_SUE_REC_NEAREMPTY	**RecNearEnd** = true
FPTR_SUE_REC_PAPEROK	**RecEmpty** = **RecNearEnd** = false
FPTR_SUE_SLP_EMPTY	**SlpEmpty** = true
FPTR_SUE_SLP_NEAREMPTY	**SlpNearEnd** = true
FPTR_SUE_SLP_PAPEROK	**SlpEmpty** = **SlpNearEnd** = false
Release 1.8 and later	
FPTR_SUE_JRN_COVER_OPEN	**CoverOpen** = true
FPTR_SUE_JRN_COVER_OK	**CoverOpen** = false if all covers closed; **CoverOpen** = true if any other cover is open
FPTR_SUE_REC_COVER_OPEN	**CoverOpen** = true
FPTR_SUE_REC_COVER_OK	**CoverOpen** = false if all covers closed; **CoverOpen** = true if any other cover is open
FPTR_SUE_SLP_COVER_OPEN	**CoverOpen** = true
FPTR_SUE_SLP_COVER_OK	**CoverOpen** = false if all covers closed; **CoverOpen** = true if any other cover is open

Release 1.8 – Clarification

The Fiscal Printer's slip station statuses must be reported independently from the slip insertion and removal methods – **beginInsertion / endInsertion** and **beginRemoval / endRemoval**. This is important because some applications base logic decisions upon Fiscal Printer state changes. That is, the application will only perform slip insertion after knowing that a slip has been placed at the entrance to the slip station. An example: After the Total key is pressed, the application enters tendering mode. It begins to monitor peripherals and the keyboard to determine the type of tender to perform. If a credit or debit card is swiped at an MSR, then its **DataEvent** causes the application to begin credit/debit tender. But if a form is placed at the slip station, then its **StatusUpdateEvent** or **SlpEmpty** property change causes the application to begin a check MICR read.

When a form is placed at the entrance to the slip station, the Fiscal Printer must fire a PTR_SUE_SLP_PAPEROK **StatusUpdateEvent** and set the **SlpEmpty** and **SlpNearEnd** properties to false. The application may then call the **beginInsertion** and **endInsertion** methods with reasonable confidence that they will succeed. Note that it must not be assumed that the form is ready for printing

after the PTR_SUE_SLP_PAPEROK is received. Only after successful **beginInsertion** and **endInsertion** calls is the form ready for printing.

When a form is removed from the slip station, the Fiscal Printer must fire a PTR_SUE_SLP_EMPTY **StatusUpdateEvent** and set the **SlpEmpty** property to true. If the **beginInsertion** and **endInsertion** method sequence has not been called, then removing the form from the slip station entrance will cause this to occur. If this method sequence has successfully completed, then the event and property change will typically occur after a **beginRemoval** and **endRemoval** method sequence. But they would also occur if the slip prints beyond the end of the form or if the form is forcibly removed.

Exception: The design of some Fiscal Printers makes it impossible for a service to determine the presence of a form until the printer "jaws" are opened, which occurs when **beginInsertion** is called. This exception is largely limited to cases where the **CapSlpFullslip** property is false, indicating a "validation" type of slip station. Validation stations typically use the same Fiscal Printer mechanism as the receipt and/or journal stations. In these cases, the slip status events must be fired as soon as possible, given the constraints of the device.

Fiscal Printer States *Updated in Release 1.8*

As previously described, a Fiscal Printer is characterized by different printing modes. Moreover, the set of commands that can be executed at a particular moment depends upon the current state of the Fiscal Printer.

The current state of the Fiscal Printer is kept in the **PrinterState** property.

The Fiscal Printer has the following states:

- **Monitor:**
  This is a neutral state. From this state, it is possible to move to most of the other Fiscal Printer states. After a successful call to the **claim** method and successful setting of the **DeviceEnabled** property to true the Fiscal Printer should be in this state unless there is a Fiscal Printer error.
- **Fiscal Receipt:**
  The Fiscal Printer is processing a fiscal receipt. All **printRec…** methods except **printRecNotPaid** and **printRecTaxID** are available for use while in this state. This state is entered from the **Monitor** state using the **beginFiscalReceipt** method.
- **Fiscal Receipt Total:**
  The Fiscal Printer has already accepted at least one payment method, but the receipt's total amount has not yet been tendered. This state is entered from the **Fiscal Receipt** state by use of the **printRecTotal** method. The Fiscal Printer remains in this state while the total remains unpaid. This state can be left by using the **printRecTotal**, **printRecNotPaid** or **printRecVoid** methods.
- **Fiscal Receipt Ending:**
  The Fiscal Printer has completed the receipt up to the **Total** line. In this state, it may be possible to print tax information using the **printRecTaxID** method if this is supported by the Fiscal Printer. This state is entered from the **Fiscal Receipt** state via the **printRecVoid** method or from the **Fiscal Receipt Total** state using either the **printRecTotal**, **printRecNotPaid**, or **printRecVoid** methods. This state is exited using the **endFiscalReceipt** method at which time the Fiscal Printer returns to the **Monitor** state.
- **Fiscal Document:**
  The Fiscal Printer is processing a fiscal document. The Fiscal Printer will accept the **printFiscalDocumentLine** method while in this state.
  This state is entered from the **Monitor** state using the **beginFiscalDocument** method. This state is exited using the **endFiscalDocument** method at which time the Fiscal Printer returns to the **Monitor** state.
- **Monitor** and **TrainingModeActive** are true**:**
  The Fiscal Printer is being used for training purposes. All fiscal receipt and document commands are available. This state is entered from the **Monitor** state using the **beginTraining** method. This state is exited using the **endTraining** method at which time the Fiscal Printer returns to the **Monitor** state.
- **Fiscal Receipt** and **TrainingModeActive** are true**:**
  The Fiscal Printer is being used for training purposes and a receipt is currently opened. To each line of the receipt, special text will be added in order to differentiate it from a fiscal receipt.
- **Fiscal Total** and **TrainingModeActive** are true**:**
  The Fiscal Printer is in training mode and receipt total is being handled.
- **Fiscal ReceiptEnding** and **TrainingModeActive** are true**:**
  The Fiscal Printer is being used for training is in the receipt ending phase.

- **NonFiscal:**
  The Fiscal Printer is printing non-fiscal output on either the receipt (echoed on the journal) or the slip. In this state the Fiscal Printer will accept the **printNormal** method. The Fiscal Printer prints a message that indicates that this is non-fiscal output with all application text. This state is entered from the **Monitor** state using the **beginNonFiscal** method. This state is exited using the **endNonFiscal** method at which time the Fiscal Printer returns to the **Monitor** state.
- **Fixed:**
  The Fiscal Printer is being used to print fixed, non-fiscal output to one of the Fiscal Printer's stations. In this state the Fiscal Printer will accept the **printFixedOutput** method. This state is entered from the **Monitor** state using the **beginFixedOutput** method. This state is exited using the **endFixedOutput** method at which time the Fiscal Printer returns to the **Monitor** state.
- **ItemList:**
  The Fiscal Printer is currently printing a line item report. In this state the Fiscal Printer will accept the **verifyItem** method. This state is entered from the **Monitor** state using the **beginItemList** method. This state is exited using the **endItemList** method at which time the Fiscal Printer returns to the **Monitor** state.
- **Report:**
  The Fiscal Printer is currently printing one of the supported types of reports. This state is entered from the **Monitor** state using one of the **printReport**, **printPeriodicTotalsReport**, **printPowerLossReport**, **printXReport** or **printZReport** methods. When the report print completes, the Fiscal Printer automatically returns to **Monitor** state.
- **FiscalSystemBlocked:**
  The Fiscal Printer is no longer operational due to one of the following reasons:
  - The Fiscal Printer has been disconnected or has lost power.
  - The Fiscal Printer's fiscal memory has been exhausted.
  - The Fiscal Printer's internal data has become inconsistent.

  In this state the Fiscal Printer will only accept methods to print reports and retrieve data. The Fiscal Printer cannot exit this state without the assistance of an authorized technician.

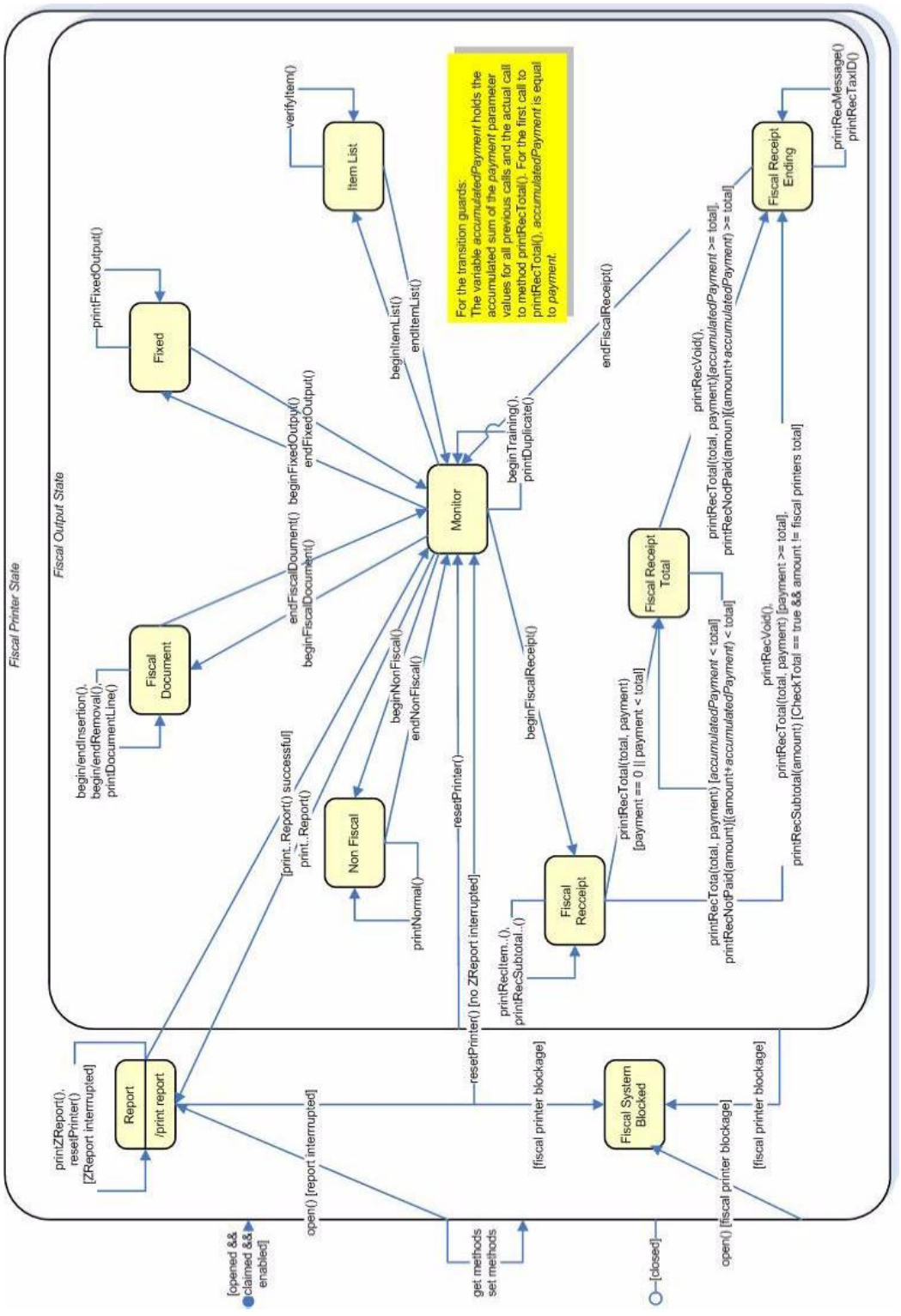
When the application sets the property **DeviceEnabled** to true it also monitors its current state. In a standard situation, the **PrinterState** property is set to FPTR_PS_MONITOR after a successfully setting **DeviceEnabled** to true. This indicates that there was no interrupted operation remaining in the Fiscal Printer.

If the Fiscal Printer is not in the FPTR_PS_MONITOR state, the state reflects the Fiscal Printer's interrupted operation and the **PowerState** property is set to PS_OFF. In this situation, it is necessary to force the Fiscal Printer to a normal state by calling the **resetPrinter** method.

This means that a power failure occurred or the last application that accessed the device left it in a not clear state.

Notice that even in this case the method returns successfully after setting **DeviceEnabled** to true. It is required that the application checks the **PowerState** property and checks for a received **StatusUpdateEvent** with the value SUE_POWER_OFF in the *Status* property after successfully setting the **DeviceEnabled** property.

Fiscal Printer State Diagram *Added in Release 1.12*



Document Printing

Using a Fiscal Printer's slip station it may be possible (depending upon the Fiscal Printer's capabilities and on special fiscal rules) to print the following kinds of documents:

- **Fiscal Documents:**
  In order to print fiscal documents an amount value must be sent to the Fiscal Printer and recorded by it. **CapSlpFiscalDocument** will be true if the Fiscal Printer supports printing fiscal documents. If fiscal documents are supported they may be either full length (if **CapSlpFullSlip** is true) or validation (if **CapSlpValidation** is true). The actual selection is made using the **SlipSelection** property but only one totalizer is assigned to all the fiscal documents.
  A fiscal document is started using the **beginFiscalDocument** method and terminated by using the **endFiscalDocument** method. A line is printed using the **printFiscalDocumentLine** method.
- **Non-Fiscal Full Length Documents:**
  Full-length slip documents may be printed if **CapSlpFullSlip** is true and **SlipSelection** is set to FPTR_SS_FULL_LENGTH.
  This document is started using the **beginNonFiscal** method and terminated by using the **endNonFiscal** method. A line is printed using the **printNormal** method.
- **Non-Fiscal Validation Documents:**
  Validation documents may be printed if **CapSlpValidation** is true and **SlipSelection** is set to FPTR_SS_VALIDATION.
  This document is started using the **beginNonFiscal** method and terminated by using the **endNonFiscal** method. A line is printed using the **printNormal** method.
- **Fixed Text Documents:**
  Fixed text documents may be printed if **CapFixedOutput** is true. If fixed text documents are supported they may be either full length (if **CapSlpFullSlip** is true) or validation (if **CapSlpValidation** is true). The actual selection is made using the **SlipSelection** property.

Ordering of Fiscal Receipt Print Requests

Updated in Release 1.13

A fiscal receipt is started using the **beginFiscalReceipt** method.

Each fiscal receipt consists of a mandatory receipt header and a mandatory receipt trailer, normally with the country specific logotype. If **CapFiscalReceiptType** is true the type of a fiscal receipt may be specified by the **FiscalReceiptType** property.

The following receipt types are defined:

- **Retail Sales Receipt:**
  The daily totalizers are updated, the **printRec...** methods must be used.
- **Simplified Invoice Receipt:**
  The daily totalizers are updated, a special title is printed, the **printRec...** methods can be used, except the **printRecRefund**, **printRecRefundVoid** **printRecItemRefund**, and **printRecItemRefundVoid** methods.
- **Service Sales Receipt:**
  The daily totalizers are updated, but a special header line is printed to identify this type of receipt. The **printRec...** methods must be used.
- **Generic Receipt:**
  Free text can be printed using **printNormal** method, no totalizer is updated. A special header line is printed to identify this type of receipt.
- **Cash-In Receipt:**
  This type of receipt helps to reconcile the cash amount. The cash-in amount is incremented by the amount given as an argument to the **printRecCash** method. Free text can be printed using **printNormal** method, the receipt can be cancelled.
- **Cash-Out Receipt:**
  This type of receipt helps to reconcile the cash amount. The cash-in amount is decremented by the amount given as an argument to the **printRecCash** method. Free text can be printed using **printNormal** method, the receipt can be cancelled.

If **CapIndependentHeader** is true, then it is up to the application to decide if the fiscal receipt header lines are to be printed at this time or not. Otherwise, the header lines are printed immediately prior to the first line item inside a fiscal receipt. Printing the header lines at this time will decrease the amount of time required to process the first fiscal receipt print method, but it may result in more receipt voids as well. The **beginFiscalReceipt** method may only be called if the Fiscal Printer is currently in the Monitor state and this call will change the Fiscal Printer's current state to Fiscal Receipt.

Before selling the first line item, it is possible to exit from the Fiscal Receipt state by calling the **endFiscalReceipt** method. If header lines have already been printed, this method will cause also receipt voiding.

Once when a Retail Sales Receipt is selected and the first line item has been printed, the Fiscal Printer remains in the Fiscal Receipt state and the following fiscal print methods are available:

printRecItem
printRecItemVoid
printRecItemAdjustment
printRecItemAdjustmentVoid
printRecItemFuel
printRecItemFuelVoid
printRecItemRefund
printRecItemRefundVoid
printRecMessage
printRecPackageAdjustment
printRecPackageAdjustVoid
printRecRefund
printRecRefundVoid
printRecSubtotal
printRecSubtotalAdjustment
printRecSubtotalAdjustVoid
printRecTotal
printRecVoid

The **printRecItem**, **printRecItemVoid**, **printRecItemAdjustment**, **printRecItemAdjustmentVoid**, **printRecItemFuel**, **printRecItemFuelVoid**, **printRecItemRefund**, **printRecItemRefundVoid**, **printRecPackageAdjustment**, **printRecPackageAdjustVoid**, **printRecRefund**, **printRecRefundVoid**, **printRecSubtotal**, **printRecSubtotalAdjustment**, **printRecMessage** (only available if **CapAdditionalLines** is true), and **printRecSubtotalAdjustVoid** will leave the Fiscal Printer in the Fiscal Receipt state. The **printRecTotal** methods will change the Fiscal Printer's state to either Fiscal Receipt Total or Fiscal Receipt Ending, depending upon whether the entire receipt total has been met. The **printRecVoid** method will change the Fiscal Printer's state to Fiscal Receipt Ending.

While in the Fiscal Receipt Total state the following fiscal print methods are available:

printRecMessage
printRecNotPaid
printRecTotal
printRecVoid

The **printRecMessage** (only available if **CapAdditionalLines** is true) method will leave the Fiscal Printer in the Fiscal Receipt Total state. The **printRecNotPaid** (only available if **CapReceiptNotPaid** is true) and **printRecTotal** methods will either leave the Fiscal Printer in the Fiscal Receipt Total state or change the Fiscal Printer's state to Fiscal Receipt Ending, depending upon whether the entire receipt total has been met. The **printRecVoid** method will change the Fiscal Printer's state to Fiscal Receipt Ending.

While in the Fiscal Receipt Ending state the following fiscal methods are available:

printRecMessage
printRecTaxID
endFiscalReceipt

The **printRecMessage** (only available if **CapAdditionalLines** is true) and

printRecTaxID methods will leave the Fiscal Printer in the Fiscal Receipt Ending state. The **endFiscalReceipt** will cause receipt closing and will then change the Fiscal Printer's state to Monitor.

At no time can the Fiscal Printer's total for the receipt be negative. If this occurs the Fiscal Printer will generate an **ErrorEvent** or throw an exception.

Fiscal Receipt Layouts *Updated in Release 1.8*

The following is an example of a typical fiscal receipt layout:

- **Header Lines:**
  Header lines contain all of the information about the store, such as telephone number, address and name of the store. All of these lines are fixed and are defined before selling the first item (using the **setHeaderLine** method). If **CapMultiContractor** property is true, two sets of header lines can be defined, assigned to the value of the **ContractorId** property. These lines may either be printed when the **beginFiscalReceipt** method is called or when the first fiscal receipt method is called.
- **Additional Header Lines:**
  Header lines defined by the **AdditionalHeader** property to be printed after the fixed header lines when the **beginFiscalReceipt** method is called.
- **Transaction Lines:**
  All of the lines of a fiscal transaction, such as line items, discounts and surcharges. Optionally they may be assigned to a specific contractor.
- **Total Line:**
  The line containing the transaction total, tender amounts and possibly change due.
- **Message Lines:**
  These are lines printed using the **printRecMessage** method.
- **Trailer Lines:**
  These are fixed promotional messages stored on the Fiscal Printer (using the **setTrailerLine** method). They are automatically printed when the **endFiscalReceipt** method is called. In fact, depending upon fiscal legislation and upon the Fiscal Printer vendor, the relative position of the trailer and the fiscal logotype lines can vary.
- **Fiscal Lines:**
  These are lines containing information to be inserted in the receipt due to fiscal legislations such as the fiscal logotype, date, time and serial number. They are also printed automatically when the **endFiscalReceipt** method is called.
- **Additional Trailer Lines:**
  These are receipt specific information defined in the **AdditionalTrailer** property to be printed after the Fiscal Lines on the receipt before cutting it, when the **endFiscalReceipt** method is called.

Example of a Fiscal Receipt

Fiscal receipt		*Definition of the line*	*UPOS methods and properties*
name of the store		***fixed header lines***	***beginFiscalReceipt*** data stored with ***setHeaderLine*** and ***setFIscalID***
address			
ZIP code and place			
fiscal identification of the store		***tax number line***	
Good Morning		***add. header line***	***AdditionalHeader*** *property*
Milk	1.000 A	***transaction line***	***printRecItem***
Special offer		***pre item line***	***PreLine*** *property*
Beer	4.000 B	***transaction line***	***printRecItem***
Discount Beer	-500 B	***transaction line***	***printRecItemAdjustment***
Bread	3.500 A	***transaction line***	***printRecItem***
Storno Bread	-3.500 A	***transaction line***	***printRecItemVoid***
Apples	2.000 A	***transaction line***	***printRecItem***
SUBTOTAL	6.500	***subtotal line***	***printRecSubtotal***
Lamp	12.000 C	***transaction line***	***printRecItem***
VAT category A	3.000	***VAT summary***	***printRecTotal*** ***(… , 10000, "Check")***
VAT 7.50%	225		
VAT category B	3.500		
VAT 12.00%	420		
VAT category C	12.000		
VAT 10.00%	1.200		
sum of VAT	1.845		
TOTALE	18.500	***total line***	
Check	10.000	***payment line***	
Cash	10.000	***payment line***	***printRecTotal*** ***(… , 10000, "Cash")***
Return	- 1.500	***change line***	
Advertising messages a.s.o.		***message line***	***printRecMessage***
THANK YOU FOR BUYING AT SABERTINI		***trailer line***	***endFiscalReceipt***
		trailer line	data stored with ***setTrailerLine*** and at initialisation time of the fiscal printer
24/05/99 14:25 No 225		***logo line***	
MF B5 012345678		***logo line***	
Good Bye		***additional trailer lines***	***AdditionalTrailer*** *property*
CONGRATULATION Mrs. Smith!			
You have won: 150 points of fidelity			

Totalizers and Fiscal Memory

The Fiscal Printer is able to select the fiscal relevant data and to accumulate and store them in following types of totalizers:

- **Receipt Totalizers:**
  The different kind of amounts of the current receipt are accumulated in receipt totalizers.
- **Day Totalizers:**
  At the end of a fiscal receipt, when calling the **endFiscalReceipt** method, the receipt totalizers are added to the day totalizers where the totals of a fiscal period (day) are summarized. The contents of the current day totalizers are printed when calling the **printXReport** method. At the end of a fiscal day or period totalizers are printed when calling the **printZReport** method.
- **Document Totalizers:**
  The different kind of amounts of the current document are accumulated in document totalizers.
- **Grand Totalizers:**
  Some of the totalizers are stored in the fiscal memory at the end of a fiscal period when calling the **printZReport** method. These are the grand totalizers. The application may print the contents of the fiscal memory by calling **printReport** method.

The application may fetch the different totalizers using the **getData** method or the **getTotalizer** method, whereas the type of totalizer can be specified by setting the **TotalizerType** property and the assignment to a contractor by setting the **ContractorId** property.

Counters

The Fiscal Printer is able to count some features of fiscal receipt and documents. The application may fetch the different counters using the **getData** method.

VAT Tables

Some Fiscal Printers support storing VAT (Value Added Tax) tables in the Fiscal Printer's memory. Some of these Fiscal Printers will allow the application to set and modify any of the table entries. Others allow only adding new table entries but do not allow existing entries to be modified. Some Fiscal Printers allow the VAT table to bet set only once.

If the Fiscal Printer supports VAT tables, **CapHasVatTable** is true. If the Fiscal Printer allows the VAT table entries to be set or modified **CapSetVatTable** is true. The maximum number of different vat rate entries in the VAT table is given by the **NumVatRates** property. VAT tables are set through a two step process. First the application uses the **setVatValue** method to set each table entry to be sent to the Fiscal Printer.

Next, the **setVatTable** method is called to send the entire VAT table to the Fiscal Printer at one time.

Receipt Duplication

In some countries, fiscal legislation can allow printing more than one copy of the same receipt. **CapDuplicateReceipt** will be true if the Fiscal Printer is capable of printing duplicate receipts. Then, setting **DuplicateReceipt** true causes the

buffering of all receipt printing commands. **DuplicateReceipt** is set false after receipt closing. In order to print the receipt again the **printDuplicateReceipt** method has to be called.

Currency Amounts, Percentage Amounts, VAT Rates, and Quantity Amounts

- Currency amounts (and also prices) are passed as values with the data type long. This is a 64 bit signed integer value that implicitly assumes four digits as the fractional part. For example, an actual value of 12345 represents 1.2345. So, the range supported is from
  -922,337,203,685,477.5808
  to
  +922,337,203,685,477.5807
  The fractional part used in the calculation unit of a Fiscal Printer may differ from the long data type. The number of digits in the fractional part is stored in the **AmountDecimalPlaces** property and determined by the Fiscal Printer. The application has to take care that calculations in the application use the same fractional part for amounts.
- If **CapHasVatTable** is true, VAT rates are passed using the indexes that were sent to the **setVatValue** method.
- If **CapHasVatTable** is false, VAT rates are passed as amounts with the data type *int32*. The number of digits in the fractional part is implicitly assumed to be four.
- Percentage amounts are used in methods which allow also surcharge and/or discount amounts. If the amounts are specified to be a percentage value the value is also passed in a parameter of type long.
- The percentage value has (as given by the long data type) four digits in the fractional part. It is the percentage (0.0001% to 99.9999%) multiplied by 10000.
- Quantity amounts are passed as values with the data type *int32*. The number of digits in the fractional part is stored in the **QuantityDecimalPlaces** property and determined by the Fiscal Printer.

Currency Change

If **CapSetCurrency** is true the Fiscal Printer is able to change the currency, the application may set a new currency (e.g., EURO) using the **setCurrency** method.

Device Sharing

The Fiscal Printer is an exclusive-use device, as follows:

- The application must claim the device before enabling it.
- The application must claim and enable the device before accessing many Fiscal Printer-specific properties.
- The application must claim and enable the device before calling methods that manipulate the device.

See the “Summary” table for precise usage prerequisites.

Properties (UML attributes)

ActualCurrency Property *Updated in Release 1.12*

Syntax **ActualCurrency:** ***int32*** **{ read-only, access after open-claim-enable }**

Remarks Holds a value identifying which actual currency is used by the Fiscal Printer.

This property is only valid if **CapSetCurrency** is true.

Values are:

Value	Meaning
FPTR_AC_BRC	The actual currency is Brazilian cruceiro.
FPTR_AC_BGL	The actual currency is Bulgarian lev.
FPTR_AC_EUR	The actual currency is EURO.
FPTR_AC_GRD	The actual currency is Greek drachma.
FPTR_AC_HUF	The actual currency is Hungarian forint.
FPTR_AC_ITL	The actual currency is Italian lira.
FPTR_AC_PLZ	The actual currency is Polish zloty.
FPTR_AC_ROL	The actual currency is Romanian leu.
FPTR_AC_RUR	The actual currency is Russian rouble.
FPTR_AC_TRL	The actual currency is Turkish lira.
FPTR_AC_CZK	The actual currency is Czechian Koruna.
FPTR_AC_UAH	The actual currency is Ukrainian Hryvnia.
FPTR_AC_SEK	The actual currency is Swedish Krona.
FPTR_AC_OTHER	The actual currency is unknown. (May be used for a country that recently fiscalized.)

This property is initialized and kept current while the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **setCurrency** Method, **CapSetCurrency** Property.

AdditionalHeader Property *Added in Release 1.6*

Syntax **AdditionalHeader:** ***string*** **{ read-write, access after open-claim-enable }**

Remarks Specifies a user specific text which will be printed on the receipt after the fixed header lines when calling the **beginFiscalReceipt** method.

This property is only valid if **CapAdditionalHeader** is true.

This property is initialized to an empty string and kept current while the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support printing text after the fixed header lines.

See Also **beginFiscalReceipt** Method, **CapAdditionalHeader** Property.

AdditionalTrailer Property *Added in Release 1.6*

Syntax **AdditionalTrailer:** ***string*** **{ read-write, access after open-claim-enable }**

Remarks Specifies a user specific text which will be printed on the receipt after the fiscal trailer lines when calling the **endFiscalReceipt** method.

This property is only valid if **CapAdditionalTrailer** is true.

This property is initialized to an empty string and kept current while the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support printing text after the fiscal trailer lines.

See Also **endFiscalReceipt** Method, **CapAdditionalTrailer** Property.

AmountDecimalPlaces Property

Syntax **AmountDecimalPlaces:** ***int32*** **{ read-only, access after open-claim-enable }**

Remarks Holds the number of decimal digits that the fiscal device uses for calculations.

This property is initialized when the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

AsyncMode Property

Syntax **AsyncMode:** ***boolean*** **{ read-write, access after open }**

Remarks If true, then some print methods such as **printRecItemAdjustment**, **printRecItem**, **printNormal**, etc. will be performed asynchronously. If false, they will be performed synchronously.

This property is initialized to false by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **"Model"** on page Intro-14 for the output model description.

CapAdditionalHeader Property *Added in Release 1.6*

Syntax **CapAdditionalHeader:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer is able to print application specific text defined in the **AdditionalHeader** property after printing the fixed header lines.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapAdditionalLines Property *Updated in Release 1.13*

Syntax **CapAdditionalLines:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer supports the printing of application defined lines on a fiscal receipt.

If true, then after all totals lines are printed it is possible to print application-defined strings, such as the ones used for fidelity cards.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapAdditionalTrailer Property *Added in Release 1.6*

Syntax **CapAdditionalTrailer:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer is able to print application specific text defined in the **AdditionalTrailer** property after printing the fiscal trailer lines.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapAmountAdjustment Property

Syntax **CapAmountAdjustment:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer handles fixed amount discounts or fixed amount surcharges on items.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapAmountNotPaid Property *Deprecated in Release 1.11*

Syntax **CapAmountNotPaid:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer allows the recording of not paid amounts.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapChangeDue Property *Added in Release 1.6*

Syntax **CapChangeDue:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the text to be printed as the cash return description when using **printRecTotal** method can be defined in the **ChangeDue** property.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapCheckTotal Property *Updated in Release 1.11*

Syntax **CapCheckTotal:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then automatic comparison of the Fiscal Printer's total and the application's total can be enabled and disabled. If false, then the automatic comparison cannot be enabled or disabled, meaning that the property **CheckTotal** can not be changed and is read-only.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **CheckTotal** Property.

CapCoverSensor Property

Syntax **CapCoverSensor:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer has a "cover open" sensor.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapDoubleWidth Property

Syntax **CapDoubleWidth:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer can print double width characters.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapDuplicateReceipt Property

Syntax **CapDuplicateReceipt: *boolean* { read-only, access after open }**

Remarks If true, then the Fiscal Printer allows printing more than one copy of the same fiscal receipt.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapEmptyReceiptIsVoidable Property *Added in Release 1.6*

Syntax **CapEmptyReceiptIsVoidable: *boolean* { read-only, access after open }**

Remarks If true, then it is allowed to void an opened receipt without any items.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapFiscalReceiptStation Property *Added in Release 1.6*

Syntax **CapFiscalReceiptStation: *boolean* { read-only, access after open }**

Remarks If true, then the Fiscal Printer supports printing transactions on the station defined by the **FiscalReceiptStation** property.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapFiscalReceiptType Property *Added in Release 1.6*

Syntax **CapFiscalReceiptType: *boolean* { read-only, access after open }**

Remarks If true, then the Fiscal Printer supports printing different types of fiscal receipts defined by the **FiscalReceiptType** property.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapFixedOutput Property

Syntax **CapFixedOutput:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer supports fixed format text printing through the **beginFixedOutput**, **printFixedOutput** and **endFixedOutput** methods.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapHasVatTable Property

Syntax **CapHasVatTable:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer has a tax table.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapIndependentHeader Property

Syntax **CapIndependentHeader:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer supports printing the fiscal receipt header lines before the first fiscal receipt command is processed.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapItemList Property

Syntax **CapItemList:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer can print a report of items of a specified VAT class.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapJrnEmptySensor Property

Syntax **CapJrnEmptySensor: *boolean* { read-only, access after open }**

Remarks If true, then the journal has an out-of-paper sensor.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapJrnNearEndSensor Property

Syntax **CapJrnNearEndSensor: *boolean* { read-only, access after open }**

Remarks If true, then the journal has a low paper sensor.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapJrnPresent Property

Syntax **CapJrnPresent: *boolean* { read-only, access after open }**

Remarks If true, then the journal print station is present.

Unlike POS printers, on Fiscal Printers the application is not able to directly access the journal. The Fiscal Printer itself prints on the journal if present.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapMultiContractor Property *Added in Release 1.6*

Syntax **CapMultiContractor: *boolean* { read-only, access after open }**

Remarks If true, then the Fiscal Printer supports more than one contractor assigned to the fiscal receipt and items.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapNonFiscalMode Property

Syntax **CapNonFiscalMode:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer allows printing in non-fiscal mode.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapOnlyVoidLastItem Property *Added in Release 1.6*

Syntax **CapOnlyVoidLastItem:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then only the last printed item can be voided.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapOrderAdjustmentFirst Property

Syntax **CapOrderAdjustmentFirst:** ***boolean*** **{ read-only, access after open }**

Remarks If false, the application has to call **printRecItem** first and then call **printRecItemAdjustment** to give a discount or a surcharge for a single article.

If true, then the application has to call **printRecItemAdjustment** first and then call **printRecItem**.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapPackageAdjustment Property *Added in Release 1.6*

Syntax **CapPackageAdjustment:** ***boolean*** **{ read-only, access after open }**

Remarks If true, an adjustment may be given to a package of booked items.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapPercentAdjustment Property

Syntax **CapPercentAdjustment: *boolean* { read-only, access after open }**

Remarks If true, then the Fiscal Printer handles percentage discounts or percentage surcharges on items.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapPositiveAdjustment Property

Syntax **CapPositiveAdjustment: *boolean* { read-only, access after open }**

Remarks If true, then it is possible to apply surcharges via the **printRecItemAdjustment** method.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapPositiveSubtotalAdjustment Property *Added in Release 1.11*

Syntax **CapPositiveSubtotalAdjustment: *boolean* { read-only, access after open }**

Remarks If true, then it is possible to apply surcharges via the **printRecSubtoalAdjustment** method.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapPostPreLine Property *Added in Release 1.6*

Syntax **CapPostPreLine: *boolean* { read-only, access after open }**

Remarks If true, then the Fiscal Printer supports printing additional lines defined by the **PostLine** and/or the **PreLine** properties when calling some **printRec...** methods.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapPowerLossReport Property

Syntax **CapPowerLossReport: *boolean* { read-only, access after open }**

Remarks If true, then the Fiscal Printer can print a power loss report using the **printPowerLossReport** method.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapPredefinedPaymentLines Property

Syntax **CapPredefinedPaymentLines:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the Fiscal Printer can store and print predefined payment descriptions.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapReceiptNotPaid Property

Syntax **CapReceiptNotPaid:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer supports using the **printRecNotPaid** method to specify a part of the receipt total that is not paid.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapRecEmptySensor Property

Syntax **CapRecEmptySensor:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the receipt has an out-of-paper sensor.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapRecNearEndSensor Property

Syntax **CapRecNearEndSensor:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the receipt has a low paper sensor.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapRecPresent Property

Syntax **CapRecPresent:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the receipt print station is present.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapRemainingFiscalMemory Property

Syntax **CapRemainingFiscalMemory:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer supports using the **RemainingFiscalMemory** property to show the amount of Fiscal Memory remaining. If false, the Fiscal Printer does not support reporting the Fiscal Memory status of the Fiscal Printer.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapReservedWord Property

Syntax **CapReservedWord:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer prints a reserved word (for example, "TOTALE") before printing the total amount.

If true, the reserved word is stored in the **ReservedWord** property. This reserved word may not be printed using any fiscal print method.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSetCurrency Property *Added in Release 1.6*

Syntax **CapSetCurrency:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer is able to change the currency to a new one by calling the **setCurrency** method.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSetHeader Property

Syntax **CapSetHeader:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then it is possible to use the **setHeaderLine** method to initialize the contents of a particular line of the receipt header.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSetPOSID Property

Syntax **CapSetPOSID:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then it is possible to use the **setPOSID** method to initialize the values of POSID and CashierID. These values are printed on each fiscal receipt.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSetStoreFiscalID Property

Syntax **CapSetStoreFiscalID:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then it is possible to use the **setStoreFiscalID** method to set up the Fiscal ID number which will be printed on each fiscal receipt.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSetTrailer Property

Syntax **CapSetTrailer:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then it is possible to use the **setTrailerLine** method to initialize the contents of a particular line of the receipt trailer.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSetVatTable Property

Syntax **CapSetVatTable: *boolean* { read-only, access after open }**

Remarks If true, then it is possible to use the **setVatValue** and **setVatTable** methods to modify the contents of the Fiscal Printer's VAT table. Some Fiscal Printers may not allow existing VAT table entries to be modified. Only new entries may be set on these Fiscal Printers.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSlpEmptySensor Property

Syntax **CapSlpEmptySensor: *boolean* { read-only, access after open }**

Remarks If true, then the slip has a "slip in" sensor.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSlpFiscalDocument Property

Syntax **CapSlpFiscalDocument: *boolean* { read-only, access after open }**

Remarks If true, then the Fiscal Printer allows fiscal printing to the slip station.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSlpFullSlip Property

Syntax **CapSlpFullSlip: *boolean* { read-only, access after open }**

Remarks If true, then the Fiscal Printer supports printing full length forms on the slip station.

It is possible to choose between full slip and validation documents by setting the **SlipSelection** property.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSlpNearEndSensor Property

Syntax **CapSlpNearEndSensor:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the slip has a "slip near end" sensor.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSlpPresent Property

Syntax **CapSlpPresent:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer has a slip station.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSlpValidation Property

Syntax **CapSlpValidation:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer supports printing validation information on the slip station.

It is possible to choose between full slip and validation documents by setting the **SlipSelection** property. In some countries, when printing non fiscal validations using the slip station a limited number of lines could be printed.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSubAmountAdjustment Property

Syntax **CapSubAmountAdjustment:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer handles fixed amount discounts on the subtotal.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSubPercentAdjustment Property

Syntax **CapSubPercentAdjustment:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer handles percentage discounts on the subtotal.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapSubtotal Property

Syntax **CapSubtotal:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then it is possible to use the **printRecSubtotal** method to print the current subtotal.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapTotalizerType Property *Added in Release 1.6*

Syntax **CapTotalizerType:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer supports reading different types of totalizers by calling the **getTotalizer** method.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapTrainingMode Property

Syntax **CapTrainingMode:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then the Fiscal Printer supports a training mode.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapValidateJournal Property

Syntax **CapValidateJournal:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then it is possible to use the **printNormal** method to print a validation string on the journal station.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CapXReport Property

Syntax **CapXReport:** ***boolean*** **{ read-only, access after open }**

Remarks If true, then it is possible to use the **printXReport** method to print an X report.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

ChangeDue Property *Added in Release 1.6*

Syntax **ChangeDue:** ***string*** **{ read-write, access after open }**

Remarks This property holds the text to be printed as a description for the cash return when using the **printRecTotal** method.

This property is only valid if **CapChangeDue** is true.

This property is initialized to an empty string by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Setting this property is not valid for this service (see **CapChangeDue** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_BAD_LENGTH: The length of the string to be printed is too long.

See Also **printRecTotal** Method, **CapChangeDue** Property.

CheckTotal Property *Updated in Release 1.11*

Syntax **CheckTotal:** ***boolean*** **{ read-write, access after open }**

Remarks If true, automatic comparison between the Fiscal Printer's total and the application's total is enabled. If false, automatic comparison is disabled. This property can be changed if **CapCheckTotal** is true. Otherwise, it is read-only.

This property is initialized to true by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Setting this property is not valid for this Service (see **CapCheckTotal**).

See Also **CapCheckTotal** Property.

ContractorId Property *Added in Release 1.6*

Syntax **ContractorId:** ***int32*** **{ read-write, access after open-claim-enable }**

Remarks The identification of the contractor to whom the receipt and/or some items of the receipt are assigned.

It is used to define different header lines to be printed on the fiscal receipt, in order to assign any item to a specific contractor and to modify the counters and totalizers to be read using **getData** and **getTotalizer** methods.

Values are:

Value	Meaning
FPTR_CID_FIRST	First contractor is defined.
FPTR_CID_SECOND	Second contractor is defined.
FPTR_CID_SINGLE	Single contractor.

This property is initialized to FPTR_CID_SINGLE and kept current while the device is enabled, which is the functionality supported prior to Release 1.6.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Setting this property is not valid for this service (see **CapMultiContractor** property).

See Also **beginFiscalReceipt** Method, **getData** Method, **getTotalizer** Method, **printRec...** Methods, **CapMultiContractor** Property.

CountryCode Property *Updated in Release 1.12*

Syntax **CountryCode: *int32* { read-only, access after open }**

Remarks Holds a value identifying which countries are supported by the Fiscal Printer. It can contain any of the following values logically ORed together:

Value	Meaning
FPTR_CC_BRAZIL	The Fiscal Printer supports Brazil's fiscal rules.
FPTR_CC_GREECE	The Fiscal Printer supports Greece's fiscal rules.
FPTR_CC_HUNGARY	The Fiscal Printer supports Hungary's fiscal rules.
FPTR_CC_ITALY	The Fiscal Printer supports Italy's fiscal rules.
FPTR_CC_POLAND	The Fiscal Printer supports Poland's fiscal rules.
FPTR_CC_TURKEY	The Fiscal Printer supports Turkey's fiscal rules.
FPTR_CC_RUSSIA	The Fiscal Printer supports Russia's fiscal rules.
FPTR_CC_BULGARIA	The Fiscal Printer supports Bulgaria's fiscal rules.
FPTR_CC_ROMANIA	The Fiscal Printer supports Romania's fiscal rules.
FPTR_CC_CZECH_REPUBLIC	The Fiscal Printer supports the Czech Republic's fiscal rules.
FPTR_CC_UKRAINE	The Fiscal Printer supports Ukraine's fiscal rules.
FPTR_CC_SWEDEN	The Fiscal Printer supports Sweden's fiscal rules. 
FPTR_CC_OTHER	This is an unknown or new fiscal country.

This property is initialized when the device is first enabled following the **open** method. (In releases prior to 1.5, this description stated that initialization took place by the **open** method. In Release 1.5, it was updated for consistency with other devices.)

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

CoverOpen Property

Syntax **CoverOpen: *boolean* { read-only, access after open-claim-enable }**

Remarks If true, then the Fiscal Printer's cover is open.

If **CapCoverSensor** is false, then the Fiscal Printer does not have a cover open sensor and this property is always false.

This property is initialized and kept current while the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

DateType Property *Updated in Release 1.11*

Syntax **DateType:** ***int32*** **{ read-write, access after open-claim-enable }**

Remarks Specifies the type of date to be requested when calling the **getDate** method.

Values are:

Value	Meaning
FPTR_DT_CONF	Date of configuration.
FPTR_DT_EOD	Date of last end of day.
FPTR_DT_RESET	Date of last reset.
FPTR_DT_RTC	Real time clock of the Fiscal Printer.
FPTR_DT_VAT	Date of last VAT change.
FPTR_DT_START 	The date and time that the fiscal day started or of the first fiscal receipt or first fiscal document.

Starting with Release 1.11 support is added for countries (e.g., Greece, Russia, Italy) where it is required by law to make a Z report and therefore end the fiscal day within a 24 hour period. If the 24 hour period after the first fiscal ticket or after the fiscal day opening is exceeded, then no new fiscal ticket can be started and printing of a Z report is required. Setting **DateType** to FPTR_DT_START and calling **getDate** provides the information necessary to detect this situation.

This property is initialized to FPTR_DT_RTC and kept current while the device is enabled, which is the functionality supported prior to Release 1.6.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support the specified type.

See Also **getDate** Method.

DayOpened Property *Updated in Release 1.6*

Syntax **DayOpened:** ***boolean*** **{ read-only, access after open-claim-enable }**

Remarks If true, then the fiscal day has been started on the Fiscal Printer by a first call to the **beginFiscalReceipt** or **beginFiscalDocument** method at a fiscal period (day).

The Fiscal Day of the Fiscal Printer can be either opened or not opened. The **DayOpened** property reflects whether or not the Fiscal Printer considers its Fiscal Day to be opened or not.

Some methods may only be called while the Fiscal Day is not yet opened (**DayOpened** is false). Methods that can be called after the Fiscal Day is opened change from country to country. Usually all the configuration methods are to be called only before the Fiscal Day is opened.

This property changes to false after calling **printZReport**.

Depending on fiscal legislation, the following methods may be allowed only if the Fiscal Printer is in the Monitor State and has not yet begun its Fiscal Day:

setCurrency
setDate
setHeaderLine
setPOSID
setStoreFiscalID
setTrailerLine
setVatTable
setVatValue

This property is initialized and kept current while the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

DescriptionLength Property *Updated in Release 1.6*

Syntax **DescriptionLength:** ***int32*** **{ read-only, access after open }**

Remarks Holds the maximum number of characters that may be passed as a description parameter.

The exact maximum number for a description parameter of a specific method can be obtained by calling **getData** method.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **getData** Method.

DuplicateReceipt Property

Syntax **DuplicateReceipt:** ***boolean*** **{ read-write, access after open }**

Remarks If true, all the printing commands inside a fiscal receipt will be buffered and they can be printed again via the **printDuplicateReceipt** method.

This property is only valid if **CapDuplicateReceipt** is true.

This property is initialized to false by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

ErrorLevel Property

Syntax **ErrorLevel:** ***int32*** **{ read-only, access after open }**

Remarks Holds the severity of the error condition.

This property has one of the following values:

Value	Meaning
FPTR_EL_NONE	No error condition is present.
FPTR_EL_RECOVERABLE	A recoverable error has occurred. (Example: Out of paper.)
FPTR_EL_FATAL	A non-recoverable error has occurred. (Example: Internal printer failure.)
FPTR_EL_BLOCKED	A severe hardware failure which can be resolved only by authorized technicians. (Example: Fiscal memory failure.). This error cannot be recovered.

This property is set just before delivering an **ErrorEvent**. When the error is cleared, then the property is changed to FPTR_EL_NONE.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

ErrorOutID Property *Updated in Release 1.6*

Syntax **ErrorOutID:** ***int32*** **{ read-only, access after open }**

Remarks Holds the identifier of the output in the queue which caused an **ErrorEvent**, when using asynchronous printing.

This property is initialized when the device is first enabled following the **open** method. (In releases prior to 1.5, this description stated that initialization took place by the **open** method. In Release 1.5, it was updated for consistency with other devices.)

This property is set just before an **ErrorEvent** is delivered.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

ErrorState Property

Syntax **ErrorState:** ***int32*** **{ read-only, access after open }**

Remarks Holds the current state of the Fiscal Printer when an **ErrorEvent** is delivered for an asynchronous output.

This property is set just before an **ErrorEvent** is delivered.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **PrinterState** Property.

ErrorStation Property

Syntax **ErrorStation:** ***int32*** **{ read-only, access after open }**

Remarks Holds the station or stations that were printing when an error was detected.

This property will be set to one of the following values: FPTR_S_JOURNAL, FPTR_S_RECEIPT, FPTR_S_SLIP, FPTR_S_JOURNAL_RECEIPT, FPTR_S_JOURNAL_SLIP, FPTR_S_RECEIPT_SLIP.

This property is only valid if the ErrorLevel is not equal to PTR_EL_NONE. It is set just before delivering an **ErrorEvent**.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

ErrorString Property

Syntax **ErrorString:** ***string*** **{ read-only, access after open }**

Remarks Holds a vendor-supplied description of the current error.

This property is set just before delivering an **ErrorEvent**. If no description is available, the property is set to an empty string. When the error is cleared, then the property is changed to an empty string.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

FiscalReceiptStation Property *Added in Release 1.6*

Syntax **FiscalReceiptStation:** ***int32*** **{ read-write, access after open-claim-enable }**

Remarks Selects the station where the transaction of the fiscal receipt started with **beginFiscalReceipt** method will be printed. Setting this property is only allowed in the Monitor State.

Values are:

Value	Meaning
FPTR_RS_RECEIPT	The following transactions will be printed on the receipt station.
FPTR_RS_SLIP	The following transactions will be printed on the slip station.

This property is only valid if **CapFiscalReceiptStation** is true.

This property is initialized to FPTR_RS_RECEIPT and kept current while the device is enabled, which is the functionality supported prior to Release 1.6.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support the specified station.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Monitor State.

See Also **beginFiscalReceipt** Method, **CapFiscalReceiptStation** Property.

FiscalReceiptType Property *Updated in Release 1.11*

Syntax **FiscalReceiptType:** ***int32*** **{ read-write, access after open-claim-enable }**

Remarks Selects the type of the fiscal receipt. Setting this property is only allowed in the Monitor State.

Values are:

Value	Meaning
FPTR_RT_CASH_IN	Cash-in receipt
FPTR_RT_CASH_OUT	Cash-out receipt
FPTR_RT_GENERIC	Generic receipt
FPTR_RT_SALES	Retail sales receipt
FPTR_RT_SERVICE	Service sales receipt
FPTR_RT_SIMPLE_INVOICE	Simplified invoice receipt
FPTR_RT_REFUND	Refund sales receipt

This property is only valid if **CapFiscalReceiptType** is true.

Starting with Release 1.11, due to the need for negative receipts (e.g., in Italy), such as refund receipts, the receipt type FPTR_RT_REFUND is added.

This property is initialized to FPTR_RT_SALES and kept current while the device is enabled, which is the functionality supported prior to Release 1.6.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support the specified receipt type.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Monitor State.

See Also **beginFiscalReceipt** Method, **CapFiscalReceiptType** Property.

FlagWhenIdle Property

Syntax **FlagWhenIdle:** ***boolean*** **{ read-write, access after open }**

Remarks If true, a **StatusUpdateEvent** will be enqueued when the device is in the idle state.

This property is automatically reset to false when the status event is delivered.

The main use of idle status event that is controlled by this property is to give the application control when all outstanding asynchronous outputs have been processed. The event will be enqueued if the outputs were completed successfully or if they were cleared by the **clearOutput** method or by an **ErrorEvent** handler.

If the **State** is already set to S_IDLE when this property is set to true, then a **StatusUpdateEvent** is enqueued immediately. The application can therefore depend upon the event, with no race condition between the starting of its last asynchronous output and the setting of this flag.

This property is initialized to false by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

JrnEmpty Property

Syntax **JrnEmpty:** ***boolean*** **{ read-only, access after open-claim-enable }**

Remarks If true, the journal is out of paper. If false, journal paper is present.

If **CapJrnEmptySensor** is false, then the value of this property is always false.

This property is initialized and kept current while the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **JrnNearEnd** Property.

JrnNearEnd Property

Syntax **JrnNearEnd:** ***boolean*** **{ read-only, access after open-claim-enable }**

Remarks If true, the journal paper is low. If false, journal paper is not low.

If **CapJrnNearEndSensor** is false, then the value of this property is always false.

This property is initialized and kept current while the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **JrnEmpty** Property.

MessageLength Property

Syntax **MessageLength:** ***int32*** **{ read-only, access after open }**

Remarks Holds the maximum number of characters that may be passed as a message line in the method **printRecMessage**. The value may change in different modes of the Fiscal Printer. For example in the mode "Fiscal Receipt" the number of characters may be bigger than in the mode "Fiscal Receipt Total."

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

MessageType Property *Added in Release 1.6*

Syntax **MessageType:** ***int32*** **{ read-write, access after open-claim-enable }**

Remarks Selects the kind of message to be printed when using the **printRecMessage** method. Values are:

Value
FPTR_MT_ADVANCE
FPTR_MT_ADVANCE_PAID
FPTR_MT_AMOUNT_TO_BE_PAID
FPTR_MT_AMOUNT_TO_BE_PAID_BACK
FPTR_MT_CARD
FPTR_MT_CARD_NUMBER
FPTR_MT_CARD_TYPE
FPTR_MT_CASH
FPTR_MT_CASHIER
FPTR_MT_CASH_REGISTER_NUMBER
FPTR_MT_CHANGE
FPTR_MT_CHEQUE
FPTR_MT_CLIENT_NUMBER
FPTR_MT_CLIENT_SIGNATURE
FPTR_MT_COUNTER_STATE
FPTR_MT_CREDIT_CARD
FPTR_MT_CURRENCY
FPTR_MT_CURRENCY_VALUE
FPTR_MT_DEPOSIT
FPTR_MT_DEPOSIT_RETURNED
FPTR_MT_DOT_LINE
FPTR_MT_DRIVER_NUMB
FPTR_MT_EMPTY_LINE
FPTR_MT_FREE_TEXT

FPTR_MT_FREE_TEXT_WITH_DAY_LIMIT
FPTR_MT_GIVEN_DISCOUNT
FPTR_MT_LOCAL_CREDIT
FPTR_MT_MILEAGE_KM
FPTR_MT_NOTE
FPTR_MT_PAID
FPTR_MT_PAY_IN
FPTR_MT_POINT_GRANTED
FPTR_MT_POINTS_BONUS
FPTR_MT_POINTS_RECEIPT
FPTR_MT_POINTS_TOTAL
FPTR_MT_PROFITED
FPTR_MT_RATE
FPTR_MT_REGISTER_NUMB
FPTR_MT_SHIFT_NUMBER
FPTR_MT_STATE_OF_AN_ACCOUNT
FPTR_MT_SUBSCRIPTION
FPTR_MT_TABLE
FPTR_MT_THANK_YOU_FOR_LOYALTY
FPTR_MT_TRANSACTION_NUMB
FPTR_MT_VALID_TO
FPTR_MT_VOUCHER
FPTR_MT_VOUCHER_PAID
FPTR_MT_VOUCHER_VALUE
FPTR_MT_WITH_DISCOUNT
FPTR_MT_WITHOUT_UPLIFT

This property is initialized to FPTR_MT_FREE_TEXT by the **open** method, which is the functionality supported prior to Release 1.6.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support this value.

See Also **printRecMessage** Method.

NumHeaderLines Property

Syntax **NumHeaderLines:** ***int32*** **{ read-only, access after open }**

Remarks Holds the maximum number of header lines that can be printed for each fiscal receipt. Header lines usually contain information such as store address, store name, store Fiscal ID. Each header line is set using the **setHeaderLine** method and remains set even after the Fiscal Printer is switched off. Header lines are automatically printed when a fiscal receipt is initiated using the **beginFiscalReceipt** method or when the first line item inside a receipt is sold.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

NumTrailerLines Property

Syntax **NumTrailerLines:** ***int32*** **{ read-only, access after open }**

Remarks Holds the maximum number of trailer lines that can be printed for each fiscal receipt. Trailer lines are usually promotional messages. Each trailer line is set using the **setTrailerLine** method and remains set even after the Fiscal Printer is switched off. Trailer lines are automatically printed either after the last **printRecTotal** or when a fiscal receipt is closed using the **endFiscalReceipt** method.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

NumVatRates Property

Syntax **NumVatRates:** ***int32*** **{ read-only, access after open }**

Remarks Holds the maximum number of vat rates that can be entered into the Fiscal Printer's Vat table.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

PostLine Property *Added in Release 1.6*

Syntax **PostLine:** ***string*** **{ read-write, access after open-claim-enable }**

Remarks An application specific text to be printed on the fiscal receipt after a line item invoked by some **printRec...** methods. The property can be written in the Fiscal Receipt State. The length of the text is reduced to a country specific value

This property is only valid if **CapPostPreLine** is true.

This property is initialized to an empty string and will be reset to an empty string after being used.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support printing post item lines or the text contains invalid characters.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_BAD_LENGTH: The length of the string is too long.

See Also **printRecSubtotal** Method, **printRecTotal** Method, **CapPostPreLine** Property.

PredefinedPaymentLines Property

Syntax **PredefinedPaymentLines:** ***string*** **{ read-only, access after open }**

Remarks Holds the list of all possible words to be used as indexes of the predefined payment lines (for example, "a, b, c, d, z"). Those indexes are used in the **printRecTotal** method for the *description* parameter.

If **CapPredefinedPaymentLines** is true, only predefined payment lines are allowed.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

PreLine Property *Added in Release 1.6*

Syntax **PreLine:** ***string*** **{ read-write, access after open-claim-enable }**

Remarks An application specific text to be printed on the fiscal receipt before a line item invoked by some **printRec...** methods. The property can be written in the Fiscal Receipt State. The length of the text is reduced to a country specific value

This property is only valid if **CapPostPreLine** is true.

This property is initialized to an empty string and will be reset to an empty string after being used.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support printing pre item lines or the text contains invalid characters.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_BAD_LENGTH: The length of the string is too long.

See Also **printRecItem** Method, **printRecItemAdjustment** Method, **printRecItemRefund** Method, **printRecRefund** Method, **printRecSubtotalAdjustment** Method, **CapPostPreLine** Property.

PrinterState Property *Updated in Release 1.13*

Syntax **PrinterState:** ***int32*** **{ read-only, access after open }**

Remarks Holds the Fiscal Printer's current operational state. This property controls which methods are currently legal.

Values are:

Value	Meaning
FPTR_PS_MONITOR	If **TrainingModeActive** is false: The Fiscal Printer is currently not in a specific operational mode. In this state the Fiscal Printer will accept any of the **begin…** methods as well as the **set…** methods. If **TrainingModeActive** is true: The Fiscal Printer is currently being used for training purposes. In this state the Fiscal Printer will accept any of the **printRec…** methods or the **endTraining** method.
FPTR_PS_FISCAL_RECEIPT	If **TrainingModeActive** is false: The Fiscal Printer is currently processing a fiscal receipt. In this state the Fiscal Printer will accept any of the **printRec…** methods. If **TrainingModeActive** is true: The Fiscal Printer is currently being used for training purposes and a fiscal receipt is currently opened.
FPTR_PS_FISCAL_RECEIPT_TOTAL	If **TrainingModeActive** is false: The Fiscal Printer has already accepted at least one payment, but the total has not been completely paid. In this state the Fiscal Printer will accept either the **printRecTotal**, **printRecNotPaid,** or **printRecMessage** methods. If **TrainingModeActive** is true: The Fiscal Printer is currently being used for training purposes and the Fiscal Printer has already accepted at least one payment, but the total has not been completely paid.
FPTR_PS_FISCAL_RECEIPT_ENDING	If **TrainingModeActive** is false: The Fiscal Printer has completed the receipt up to the total line. In this state the Fiscal Printer will accept either the **printRecMessage** or **endFiscalReceipt** methods. If **TrainingModeActive** is true: The Fiscal Printer is currently being used for training purposes and a fiscal receipt is going to be closed.

FPTR_PS_FISCAL_DOCUMENT
: The Fiscal Printer is currently processing a fiscal slip. In this state the Fiscal Printer will accept either the **printFiscalDocumentLine** or **endFiscalDocument** methods.

FPTR_PS_FIXED_OUTPUT
: The Fiscal Printer is currently processing fixed text output to one or more stations. In this state the Fiscal Printer will accept either the **printFixedOutput** or **endFixedOutput** methods.

FPTR_PS_ITEM_LIST
: The Fiscal Printer is currently processing an item list report. In this state the Fiscal Printer will accept either the **verifyItem** or **endItemList** methods.

FPTR_PS_NONFISCAL
: The Fiscal Printer is currently processing non-fiscal output to one or more stations. In this state the Fiscal Printer will accept either the **printNormal** or **endNonFiscal** methods.

FPTR_PS_LOCKED
: The Fiscal Printer has encountered a non-recoverable hardware problem. An authorized Fiscal Printer technician must be contacted to exit this state.

FPTR_PS_REPORT
: The Fiscal Printer is currently processing a fiscal report. In this state the Fiscal Printer will not accept any methods until the report has completed.

There are a few methods that are accepted in any state except FPTR_PS_LOCKED. These are **beginInsertion**, **endInsertion**, **beginRemoval**, **endRemoval**, **getDate**, **getData**, **getTotalizer**, **getVatEntry**, **resetPrinter** and **clearOutput**.

This property is initialized when the device is first enabled following the **open** method. (In releases prior to 1.5, this description stated that initialization took place by the **open** method. In Release 1.5, it was updated for consistency with other devices.)

Errors
: A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

QuantityDecimalPlaces Property *Updated in Release 1.6*

Syntax
: **QuantityDecimalPlaces:** ***int32*** **{ read-only, access after open }**

Remarks
: Holds the number of decimal digits in the fractional part that should be assumed to be in any quantity parameter.

This property is initialized when the device is first enabled following the **open** method. (In releases prior to 1.5, this description stated that initialization took place by the **open** method. In Release 1.5, it was updated for consistency with other devices.)

Errors
: A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

QuantityLength Property *Updated in Release 1.6*

Syntax **QuantityLength:** ***int32*** **{ read-only, access after open }**

Remarks Holds the maximum number of digits that may be passed as a quantity parameter, including both the whole and fractional parts.

This property is initialized when the device is first enabled following the **open** method. (In releases prior to 1.5, this description stated that initialization took place by the **open** method. In Release 1.5, it was updated for consistency with other devices.)

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

RecEmpty Property

Syntax **RecEmpty:** ***boolean*** **{ read-only, access after open-claim-enable }**

Remarks If true, the receipt is out of paper. If false, receipt paper is present.

If **CapRecEmptySensor** is false, then this property is always false.

This property is initialized and kept current while the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **RecNearEnd** Property.

RecNearEnd Property

Syntax **RecNearEnd:** ***boolean*** **{ read-only, access after open-claim-enable }**

Remarks If true, the receipt paper is low. If false, receipt paper is not low.

If **CapRecNearEndSensor** is false, then this property is always false.

This property is initialized and kept current while the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **RecEmpty** Property.

RemainingFiscalMemory Property

Syntax **RemainingFiscalMemory:** ***int32*** **{ read-only, access after open-claim-enable }**

Remarks Holds the remaining counter of Fiscal Memory.

This property is initialized and kept current while the device is enabled and may be updated by **printZReport** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **CapRemainingFiscalMemory** Property.

ReservedWord Property

Syntax **ReservedWord:** ***string*** **{ read-only, access after open }**

Remarks Holds the string that is automatically printed with the total when the **printRecTotal** method is called. This word may not occur in any string that is passed into any fiscal output methods.

This property is only valid if **CapReservedWord** is true.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

SlpEmpty Property

Syntax **SlpEmpty:** ***boolean*** **{ read-only, access after open-claim-enable }**

Remarks If true, a slip form is not present. If false, a slip form is present.

If **CapSlpEmptySensor** is false, then this property is always false.

This property is initialized and kept current while the device is enabled.

Note:

The "slip empty" sensor should be used primarily to determine whether a form has been inserted before printing. It can also be monitored to determine whether a form is still in place. This sensor is usually placed one or more print lines above the slip print head.

However, the "slip near end" sensor (when present) should be used to determine when nearing the end of the slip. This sensor is usually placed one or more print lines below the slip print head.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **SlpNearEnd** Property.

SlpNearEnd Property

Syntax **SlpNearEnd:** ***boolean*** **{ read-only, access after open-claim-enable }**

Remarks If true, the slip form is near its end. If false, the slip form is not near its end. The "near end" sensor is also sometimes called the "trailing edge" sensor, referring to the bottom edge of the slip.

If **CapSlpNearEndSensor** is false, then this property is always false.

This property is initialized and kept current while the device is enabled.

Note:

However, the "slip near end" sensor (when present) should be used to determine when nearing the end of the slip. This sensor is usually placed one or more print lines below the slip print head.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

See Also **SlpEmpty** Property.

SlipSelection Property

Syntax **SlipSelection:** ***int32*** **{ read-write, access after open-claim-enable }**

Remarks Selects the kind of document to be printed on the slip station.

This property has one of the following values:

Value	Meaning
FPTR_SS_FULL_LENGTH	Print full length documents.
FPTR_SS_VALIDATION	Print validation documents.

This property is initialized to FPTR_SS_FULL_LENGTH by the **claim** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	An invalid slip type was specified.

TotalizerType Property *Added in Release 1.6*

Syntax **TotalizerType:** ***int32*** **{ read-write, access after open-claim-enable }**

Remarks Specifies the type of totalizer to be requested when calling the **getTotalizer** method.

Values are:

Value	Meaning
FPTR_TT_DOCUMENT	Document totalizer
FPTR_TT_DAY	Day totalizer
FPTR_TT_RECEIPT	Receipt totalizer
FPTR_TT_GRAND	Grand totalizer

This property is only valid if **CapTotalizerType** is true.

This property is initialized to FPTR_TT_DAY and kept current while the device is enabled, which is the functionality supported prior to Release 1.6.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support defining totalizer types or an invalid type was specified.

See Also **getTotalizer** Method, **CapTotalizerType** Property.

TrainingModeActive Property

Syntax **TrainingModeActive:** ***boolean*** **{ read-only, access after open-claim-enable }**

Remarks Holds the current Fiscal Printer's operational state concerning the training mode. Training mode allows all fiscal commands, but each receipt is marked as non-fiscal and no internal Fiscal Printer registers are updated with any data while in training mode. Some countries' fiscal rules require that all blank characters on a training mode receipt be printed as some other character. Italy, for example, requires that all training mode receipts print a "?" instead of a blank.

This property has one of the following values:

Value	Meaning
true	The Fiscal Printer is currently in training mode. That means no data are written into the EPROM of the Fiscal Printer.
false	The Fiscal Printer is currently in normal mode. All printed receipts will also update the fiscal memory.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page Intro-20.

Methods (UML operations)

beginFiscalDocument Method *Updated in Release 1.11*

Syntax **beginFiscalDocument (documentAmount: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
documentAmount	Amount of document to be stored by the Fiscal Printer.

Remarks Initiates fiscal printing to the slip station.

This method is only supported if **CapSlpFiscalDocument** is true.

If this is the first call to the **beginFiscalDocument** method, the Fiscal Day will be started and the **DayOpened** property will be set to true.

Each fiscal line will be printed using the **printFiscalDocumentLine** method. The fiscal document handling would be as follows:

```
beginFiscalDocument()
        beginInsertion(); endInsertion()
        // print fist page
        printFiscalDocumentLine()*
        beginRemoval(); endRemoval()
        beginInsertion(); endInsertion()
        // print second page
        printFiscalDocumentLine()*
        beginRemoval(); endRemoval()
endFiscalDocument()
```

If this method is successful, the **PrinterState** property will be changed to FPTR_PS_FISCAL_DOCUMENT.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The slip station does not exist (see the **CapSlpPresent** property) or the printer does not support fiscal output to the slip station (see the **CapSlpFiscalDocument** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The printer's current state does not allow this state transition. *ErrorCodeExtended* = EFPTR_SLP_EMPTY: There is no paper in the slip station. *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: The *documentAmount* parameter is invalid. *ErrorCodeExtended* = EFPTR_MISSING_SET_CURRENCY: The new receipt cannot be opened, the Fiscal Printer is expecting the current currency to be changed by calling **setCurrency** method. *ErrorCodeExtended* = EFPTR_DAY_END_REQUIRED: The completion of the fiscal day is required by calling **printZReport**. No further fiscal receipts or documents can be started before this is done.

See Also **CapSlpFiscalDocument** Property, **CapSlpPresent** Property, **AmountDecimalPlaces** Property, **DayOpened** Property, **PrinterState** Property, **beginInsertion** Method, **endFiscalDocument** Method, **endInsertion** Method, **printFiscalDocumentLine** Method, **printZReport** Method.

beginFiscalReceipt Method *Updated in Release 1.11*

Syntax **beginFiscalReceipt (printHeader:** ***boolean*** **):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
printHeader	Indicates if the header lines are to be printed at this time.

Remarks Initiates fiscal printing to the receipt station.

If **CapFiscalReceiptStation** is true the **FiscalReceiptStation** property defines the station where the receipt will be printed. If **CapFiscalReceiptStation** is false the receipt will be printed on the receipt station. If **CapFiscalReceiptType** is true the receipt type must be defined in **FiscalReceiptType** and a header line according to the specified receipt type will be printed.

If this is the first call to the **beginFiscalReceipt** method, the Fiscal Day will be started and the **DayOpened** property will be set to true.

If *printHeader* and **CapIndependentHeader** are both true all defined header lines will be printed before control is returned. Otherwise, header lines will be printed when the first item is sold in the case they are not printed at the end of the preceding receipt. If **CapAdditionalHeader** is true, application specific header lines defined by the **AdditionalHeader** property will be printed after the fixed header lines.

If **CapMultiContractor** is true, the current receipt is assigned to the contractor specified by the **ContractorId** property.

If this method is successful, the **PrinterState** property will be changed to FPTR_PS_FISCAL_RECEIPT.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	An invalid receipt type was specified.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition. *ErrorCodeExtended* = EFPTR_MISSING_SET_CURRENCY: The new receipt cannot be opened, the Fiscal Printer is expecting the current currency to be changed by calling **setCurrency** method. *ErrorCodeExtended* = EFPTR_DAY_END_REQUIRED: The completion of the fiscal day is required by calling **printZReport**. No further fiscal receipts or documents can be started before this is done.

See Also **CapAdditionalHeader** Property, **CapFiscalReceiptStation** Property, **CapFiscalReceiptType** Property, **CapIndependentHeader** Property, **CapMultiContractor** Property, **AdditionalHeader** Property, **ContractorId** Property, **DayOpened** Property, **FiscalReceiptStation** Property, **FiscalReceiptType** Property, **PrinterState** Property, **endFiscalReceipt** Method, **printRec…** Methods.

beginFixedOutput Method

Syntax **beginFixedOutput (station: *int32*, documentType: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
station	The Fiscal Printer station to be used. May be either FPTR_S_RECEIPT or FPTR_S_SLIP.
documentType	Identifier of a document stored in the Fiscal Printer.

Remarks Initiates non-fiscal fixed text printing on a Fiscal Printer station.
This method is only supported if **CapFixedOutput** is true.

If the *station* parameter is FPTR_S_SLIP, the slip paper must be inserted into the slip station using **begin/endInsertion** before calling this method.

Each fixed output will be printed using the **printFixedOutput** method. If this method is successful, the **PrinterState** property will be changed to FPTR_PS_FIXED_OUTPUT. The **endFixedOutput** method ends fixed output modality and resets **PrinterState**.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	One of the following errors occurred: • Station does not exist (see the **CapSlpPresent** property). • Fiscal Printer does not support fixed output (see the **CapFixedOutput** property). • *station* parameter is invalid. • *documentType* is invalid.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition. *ErrorCodeExtended* = EFPTR_SLP_EMPTY: There is no paper in the slip station.

See Also **CapFixedOutput** Property, **CapSlpPresent** Property, **PrinterState** Property, **beginInsertion** Method, **endFixedOutput** Method, **endInsertion** Method, **printFixedOutput** Method.

beginInsertion Method

Syntax **beginInsertion (timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
timeout	The *timeout* parameter gives the number of milliseconds before failing the method.

If zero, the method tries to begin insertion mode, then returns the appropriate status immediately. If FOREVER (-1), the method tries to begin insertion mode, then waits as long as needed until either the form is inserted or an error occurs.

Remarks Initiates slip processing.

When called, the slip station is made ready to receive a form by opening the form's handling "jaws" or activating a form insertion mode. This method is paired with the **endInsertion** method for controlling form insertion.

If the Fiscal Printer device cannot be placed into insertion mode, a UposException is thrown. Otherwise, the device continues to monitor form insertion until either:

- The form is successfully inserted.
- The form is not inserted before *timeout* milliseconds have elapsed, or an error is reported by the Fiscal Printer device. In this case, a UposException is thrown with an *ErrorCode* of E_TIMEOUT or another value. The Fiscal Printer device remains in form insertion mode. This allows an application to perform some user interaction and reissue the **beginInsertion** method without altering the form handling mechanism.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The slip station does not exist (see the **CapSlpPresent** property) or an invalid *timeout* parameter was specified.
E_TIMEOUT	The specified time has elapsed without the form being properly inserted.

See Also **CapSlpPresent** Property, **endInsertion** Method, **beginRemoval** Method, **endRemoval** Method.

beginItemList Method

Syntax **beginItemList (vatID: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
vatID	Vat identifier for reporting.

Remarks Initiates a validation report of items belonging to a particular VAT class.

This method is only supported if **CapItemList** is true.

If this method is successful, **PrinterState** will be changed to FPTR_PS_ITEM_LIST.
After this method, only **verifyItem** and **endItemList** methods may be called.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support an item list report (see the **CapItemList** property) or the Fiscal Printer does not support VAT tables (see the **CapHasVatTable** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition. *ErrorCodeExtended* = EFPTR_BAD_VAT: The *vatID* parameter is invalid.

See Also **CapHasVatTable** Property, **CapItemList** Property, **PrinterState** Property, **endItemList** Method, **verifyItem** Method.

beginNonFiscal Method

Syntax **beginNonFiscal ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Initiates non-fiscal operations on the Fiscal Printer.

This method is only supported if **CapNonFiscalMode** is true. Output in this mode is accomplished using the **printNormal** method. This method can be successfully called only if the current value of the **PrinterState** property is FPTR_PS_MONITOR. If this method is successful, the **PrinterState** property will be changed to FPTR_PS_NONFISCAL. In order to stop non fiscal modality **endNonFiscal** method should be called.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support non-fiscal output (see the **CapNonFiscalMode** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition.

See Also **CapNonFiscalMode** Property, **PrinterState** Property, **endNonFiscal** Method, **printNormal** Method.

beginRemoval Method

Syntax **beginRemoval (timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
timeout	The *timeout* parameter gives the number of milliseconds before failing the method.

If zero, the method tries to begin removal mode, then returns the appropriate status immediately. If FOREVER (-1), the method tries to begin removal mode, then waits as long as needed until either the form is removed or an error occurs.

Remarks Initiates form removal processing.

When called, the Fiscal Printer is made ready to remove a form by opening the form handling "jaws" or activating a form ejection mode. This method is paired with the **endRemoval** method for controlling form removal.

If the Fiscal Printer device cannot be placed into removal or ejection mode, a UposException is thrown. Otherwise, the device continues to monitor form removal until either:

- The form is successfully removed.
- The form is not removed before *timeout* milliseconds have elapsed, or an error is reported by the Fiscal Printer device. In this case, a UposException is thrown with an *ErrorCode* of E_TIMEOUT or another value. The Fiscal Printer device remains in form removal mode. This allows an application to perform some user interaction and reissue the **beginRemoval** method without altering the form handling mechanism.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not have a slip station (see the **CapSlpPresent** property) or an invalid *timeout* parameter was specified.
E_TIMEOUT	The specified time has elapsed without the form being properly removed.

See Also **CapSlpPresent** Property, **beginInsertion** Method, **endInsertion** Method**,** **endRemoval** Method.

beginTraining Method

Syntax **beginTraining ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Initiates training operations.

This method is only supported if **CapTrainingMode** is true. Output in this mode is accomplished using the **printRec…** methods in order to print a receipt or other methods to print reports. This method can be successfully called only if the current value of the **PrinterState** property is FPTR_PS_MONITOR. If this method is successful, the **TrainingModeActive** property will be changed to true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support training mode (see the **CapTrainingMode** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition.

See Also **CapTrainingMode** Property, **PrinterState** Property, **TrainingModeActive** Property, **endTraining** Method, **printRec…** Methods.

clearError Method

Syntax **clearError ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Clears all Fiscal Printer error conditions.
This method is always performed synchronously.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_FAILURE	Error recovery failed.

endFiscalDocument Method

Syntax **endFiscalDocument ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Terminates fiscal printing to the slip station.

This method is only supported if **CapSlpFiscalDocument** is true.
If this method is successful, the **PrinterState** property will be changed to FPTR_PS_MONITOR.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support fiscal output to the slip station (see the **CapSlpFiscalDocument** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Document state.

See Also **CapSlpFiscalDocument** Property, **PrinterState** property, **beginFiscalDocument** Method, **printFiscalDocumentLine** Method.

endFiscalReceipt Method *Updated in Release 1.6*

Syntax **endFiscalReceipt (printHeader: *boolean*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
printHeader	Indicates if the header lines of the following receipt are to be printed at this time.

Remarks Terminates fiscal printing to the receipt station.

If *printHeader* is false, this method will close the current fiscal receipt, print the trailer lines, if they were not already printed after the total lines, and cut it.
If *printHeader* is true additionally the header of the next receipt will be printed before cutting the receipt, otherwise the header will be printed when beginning the next receipt.
All functions carried out by this method will be completed before this call returns.

If **CapAdditionalTrailer** is true application specific trailer lines defined by the **AdditionalTrailer** property will be printed after the fiscal trailer lines.

If this method is successful, the **PrinterState** property will be changed to FPTR_PS_MONITOR.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt Ending state.

See Also **beginFiscalReceipt** Method, **printRec…** Methods, **CapAdditionalTrailer** Property, **AdditionalTrailer** Property.

endFixedOutput Method

Syntax **endFixedOutput ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Terminates non-fiscal fixed text printing on a Fiscal Printer station.

This method is only supported if **CapFixedOutput** is true. If this method is successful, the **PrinterState** property will be changed to FPTR_PS_MONITOR.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support fixed output (see the **CapFixedOutput** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fixed Output state.

See Also **beginFixedOutput** Method, **printFixedOutput** Method.

endInsertion Method

Syntax **endInsertion ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Ends form insertion processing.

When called, the Fiscal Printer is taken out of form insertion mode. If the slip device has forms "jaws," they are closed by this method. If no form is present, a UposException is thrown with its *ErrorCodeExtended* property set to EFPTR_SLP_EMPTY.

This method is paired with the **beginInsertion** method for controlling form insertion. The application may choose to call this method immediately after a successful **beginInsertion** if it wants to use the Fiscal Printer sensors to determine when a form is positioned within the slip printer. Alternatively, the application may prompt the user and wait for a key press before calling this method.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer is not in slip insertion mode.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_COVER_OPEN: The device was taken out of insertion mode while the Fiscal Printer cover was open. *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The device was taken out of insertion mode without a form being inserted.

See Also **beginInsertion** Method, **beginRemoval** Method, **endRemoval** Method.

endItemList Method *Updated in Release 1.13*

Syntax **endItemList ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Terminates a validation report of items belonging to a particular VAT class.

This method is only supported if **CapItemList** is true and **CapHasVatTable** is true.

This method is paired with the **beginItemList** method.

This method can be successfully called only if current value of **PrinterState** property is equal to FPTR_PS_ITEM_LIST.

If this method is successful, the **PrinterState** property will be changed to FPTR_PS_MONITOR.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support item list report (see the **CapItemList** property) or the Fiscal Printer does not support VAT tables (see the **CapHasVatTable** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition.

See Also **CapItemList** Property, **CapHasVatTable** Property, **beginItemList** Method, **verifyItem** Method.

endNonFiscal Method

Syntax **endNonFiscal ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Terminates non-fiscal operations on one Fiscal Printer station.

This method is only supported if **CapNonFiscalMode** is true. If this method is successful, the **PrinterState** property will be changed to FPTR_PS_MONITOR.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support non-fiscal output (see the **CapNonFiscalMode** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Non-Fiscal state.

See Also **beginNonFiscal** Method, **printNormal** Method.

endRemoval Method

Syntax **endRemoval ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Ends form removal processing.

When called, the Fiscal Printer is taken out of form removal or ejection mode. If a form is present, a UposException is thrown with the *ErrorCodeExtended* property set to EFPTR_SLP_FORM.

This method is paired with the **beginRemoval** method for controlling form removal. The application may choose to call this method immediately after a successful **beginRemoval** if it wants to use the Fiscal Printer sensors to determine when the form has been removed. Alternatively, the application may prompt the user and wait for a key press before calling this method.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer is not in slip removal mode.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_SLP_FORM: The device was taken out of removal mode while a form was still present.

See Also **beginInsertion** Method, **endInsertion** Method, **beginRemoval** Method.

endTraining Method

Syntax **endTraining ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Terminates training operations on either the receipt or the slip station.

This method is only supported if **CapTrainingMode** is true. If this method is successful, the **TrainingModeActive** property will be changed to false.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support training mode (see the **CapTrainingMode** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Training state.

See Also **CapTrainingMode** property, **beginTraining** Method, **printRec…** Methods.

getData Method *Updated in Release 1.12*

Syntax **getData (dataItem: *int32*, inout optArgs: *int32*, inout data: *string*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
dataItem	The specific data item to retrieve.
optArgs	For some *dataItem* this additional argument is needed. Consult the Service vendor's documentation for further use of this argument.
data	Character string to hold the data retrieved.

The *dataItem* parameter has one of the following values:

Value	Meaning
Identification data	
FPTR_GD_FIRMWARE	Get the Fiscal Printer's firmware release number.
FPTR_GD_PRINTER_ID	Get the Fiscal Printer's fiscal ID.
Totals	
FPTR_GD_CURRENT_TOTAL	Get the current receipt total.
FPTR_GD_DAILY_TOTAL	Get the daily total.
FPTR_GD_GRAND_TOTAL	Get the Fiscal Printer's grand total.
FPTR_GD_MID_VOID	Get the total number of voided receipts.
FPTR_GD_NOT_PAID	Get the current total of not paid receipts.
FPTR_GD_RECEIPT_NUMBER	Get the number of fiscal receipts printed.
FPTR_GD_REFUND	Get the current total of refunds.
FPTR_GD_REFUND_VOID	Get the current total of voided refunds.
Fiscal memory counts	
FPTR_GD_NUMB_CONFIG_BLOCK	Get the grand number of configuration blocks.
FPTR_GD_NUMB_CURRENCY_BLOCK	Get the grand number of currency blocks.
FPTR_GD_NUMB_HDR_BLOCK	Get the grand number of header blocks.
FPTR_GD_NUMB_RESET_BLOCK	Get the grand number of reset blocks.
FPTR_GD_NUMB_VAT_BLOCK	Get the grand number of VAT blocks.
Counter	
FPTR_GD_FISCAL_DOC	Get the number of daily fiscal documents.
FPTR_GD_FISCAL_DOC_VOID	Get the number of daily voided fiscal documents.
FPTR_GD_FISCAL_REC	Get the number of daily fiscal sales receipts.
FPTR_GD_FISCAL_REC_VOID	Get the number of daily voided fiscal sales receipts.
FPTR_GD_NONFISCAL_DOC	Get the number of daily non fiscal documents.

FPTR_GD_NONFISCAL_DOC_VOID	Get the number of daily voided non fiscal documents.
FPTR_GD_NONFISCAL_REC	Get the number of daily non fiscal receipts.
FPTR_GD_RESTART	Get the Fiscal Printer's restart count
FPTR_GD_SIMP_INVOICE	Get the number of daily simplified invoices.
FPTR_GD_Z_REPORT	Get the Z report number.

Fixed fiscal printer text

FPTR_GD_TENDER	Get the payment description used in the **printRecTotal** method, defined by the given identifier in the *optArgs* argument.Valid only, if the **CapPredefinedPaymentLines** property is true.

Linecounter

FPTR_GD_LINECOUNT	Get the number of printed lines, defined by the given identifier in the *optArgs* argument. If the **CapMultiContractor** property is true, line counters depend on the contractor defined by the **ContractorId** property.

Description length

FPTR_GD_DESCRIPTION_LENGTH	Get the maximum number of characters that may be passed as a description parameter for a specific method, defined by the given identifier in the *optArgs* argument.

If *dataItem* is FPTR_GD_TENDER the *optArgs* parameter has to be set to one of the following values:

Value	Meaning
FPTR_PDL_CASH	Cash.
FPTR_PDL_CHEQUE	Cheque.
FPTR_PDL_CHITTY	Chitty.
FPTR_PDL_COUPON	Coupon.
FPTR_PDL_CURRENCY	Currency.
FPTR_PDL_DRIVEN_OFF	
FPTR_PDL_EFT_IMPRINTER	Printer EFT.
FPTR_PDL_EFT_TERMINAL	Terminal EFT.
FPTR_PDL_TERMINAL_IMPRINTER	
FPTR_PDL_FREE_GIFT	Gift.
FPTR_PDL_GIRO	Giro.
FPTR_PDL_HOME	Home.
FPTR_PDL_IMPRINTER_WITH_ISSUER	
FPTR_PDL_LOCAL_ACCOUNT	Local account.
FPTR_PDL_LOCAL_ACCOUNT_CARD	Local card account.
FPTR_PDL_PAY_CARD	Pay card.
FPTR_PDL_PAY_CARD_MANUAL	Manual pay card.
FPTR_PDL_PREPAY	Prepay.
FPTR_PDL_PUMP_TEST	Pump test.

FPTR_PDL_SHORT_CREDIT	Credit.
FPTR_PDL_STAFF	Staff.
FPTR_PDL_VOUCHER	Voucher.

If *dataItem* is FPTR_GD_LINECOUNT the *optArgs* parameter has to be set to one of the following values:

Value	Meaning
FPTR_LC_ITEM	Number of item lines.
FPTR_LC_ITEM_VOID	Number of voided item lines.
FPTR_LC_DISCOUNT	Number of discount lines.
FPTR_LC_DISCOUNT_VOID	Number of voided discount lines.
FPTR_LC_SURCHARGE	Number of surcharge lines.
FPTR_LC_SURCHARGE_VOID	Number of voided surcharge lines.
FPTR_LC_REFUND	Number of refund lines.
FPTR_LC_REFUND_VOID	Number of voided refund lines.
FPTR_LC_SUBTOTAL_DISCOUNT	Number of subtotal discount lines.
FPTR_LC_SUBTOTAL_DISCOUNT_VOID	Number of voided subtotal discount lines.
FPTR_LC_SUBTOTAL_SURCHARGE	Number of subtotal surcharge lines.
FPTR_LC_SUBTOTAL_SURCHARGE_VOID	Number of voided subtotal surcharge lines.
FPTR_LC_COMMENT	Number of comment lines.
FPTR_LC_SUBTOTAL	Number of subtotal lines.
FPTR_LC_TOTAL	Number of total lines.

If *dataItem* is FPTR_GD_DESCRIPTION_LENGTH the *optArgs* parameter has to be set to one of the following values:

Value	Meaning
FPTR_DL_ITEM	**printRecItem** method.
FPTR_DL_ITEM_ADJUSTMENT	**printRecItemAdjustment** method.
FPTR_DL_ITEM_FUEL	**printRecItemFuel** method.
FPTR_DL_ITEM_FUEL_VOID	**printRecItemFuelVoid** method.
FPTR_DL_NOT_PAID	**printRecNotPaid** method.
FPTR_DL_PACKAGE_ADJUSTMENT	**printRecPackageAdjustment** method.
FPTR_DL_REFUND	**printRecRefund** method, **printRecItemRefund** method.
FPTR_DL_REFUND_VOID	**printRecRefundVoid** method, **printRecItemRefundVoid** method.
FPTR_DL_SUBTOTAL_ADJUSTMENT	**printRecSubtotalAdjustment** method.
FPTR_DL_TOTAL	**printRecTotal** method.
FPTR_DL_VOID	**printRecVoid** method.
FPTR_DL_VOID_ITEM	**printRecItemVoid** and **printRecItemAdjustmentVoid** methods.

Remarks Retrieves data and counters from the printer's fiscal module.

If **CapMultiContractor** is true, line counters depend on the contractor defined by the **ContractorId** property.

The data is returned in a string because some of the fields, such as the grand total, might overflow a 4-byte integer.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	The *dataItem*, *optArgs* or **ContractorId** specified is invalid.

See Also **printRecTotal** Method, **CapPredefinedPaymentLines** Property, **ContractorId** Property, **PredefinedPaymentLines** Property.

getDate Method *Updated in Release 1.6*

Syntax **getDate (inout date: *string*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
date	Date and time returned as a string.

Remarks Gets the Fiscal Printer's date and time specified by the **DateType** property.

The date and time are returned as a string in the format "ddmmyyyyhhmm":

dd	day of the month (1 - 31)
mm	month (1 - 12)
yyyy	year (1997-)
hh	hour (0-23)
mm	minutes (0-59)

The fiscal controller may not support hours and minutes depending on the date type. In such cases the corresponding fields in the returned string are filled with "0".

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Retrieval of the date and time is not valid at this time.

See Also **DateType** Property.

getTotalizer Method *Updated in Release 1.6*

Syntax **getTotalizer (vatID: *int32*, optArgs: *int32*, inout data: *string*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
vatID	VAT identifier of the required totalizer.
optArgs	Specifies the required totalizer.
data	Totalizer returned as a string.

The *optArgs* parameter has one of the following values:

Value	Meaning
FPTR_GT_GROSS	Gross totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_NET	Net totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_DISCOUNT	Discount totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_DISCOUNT_VOID	Voided discount totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_ITEM	Item totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_ITEM_VOID	Voided item totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_NOT_PAID	Not paid totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_REFUND	Refund totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_REFUND_VOID	Voided refund totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_SUBTOTAL_DISCOUNT	Subtotal discount totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_SUBTOTAL_DISCOUNT_VOID	Voided discount totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_SUBTOTAL_SURCHARGES	Subtotal surcharges totalizer specified by the **TotalizerType** and **ContractorId** properties.

FPTR_GT_SUBTOTAL_SURCHARGES_VOID	Voided surcharges totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_SURCHARGE	Surcharge totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_SURCHARGE_VOID	Voided surcharge totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_VAT	VAT totalizer specified by the **TotalizerType** and **ContractorId** properties.
FPTR_GT_VAT_CATEGORY	VAT totalizer per VAT category specified by the **TotalizerType** and **ContractorId** properties associated to the given *vatID*.

Remarks Gets the totalizer specified by the *optArgs* argument Some of the totalizers such as item or VAT totalizers may be associated with the given *vatID*.

If **CapTotalizerType** is true the type of totalizer (grand, day, receipt specific) depends on the **TotalizerType** property.

If **CapMultiContractor** is true the type depends on the **ContractorId** property.

If **CapSetVatTable** is false, then only one totalizer is present.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	One of the following errors occurred: • The *vatID* parameter is invalid, or • The **ContractorId** property is invalid, or • The specified totalizer is not available.

See Also **CapTotalizerType** Property, **TotalizerType** Property, **CapMultiContractor** Property, **ContractorId** Property.

getVatEntry Method *Updated in Release 1.11*

Syntax **getVatEntry (vatID: *int32*, optArgs: *int32*, inout vatRate: *int32*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
vatID	VAT identifier of the required rate.
optArgs	For some countries, this additional argument may be needed. Consult the Fiscal Printer Service vendor's documentation for details.
vatRate	The rate associated with the VAT identifier.

Remarks Gets the rate associated with a given VAT identifier.

This method is only supported if **CapHasVatTable** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The *vatID* parameter is invalid, or **CapHasVatTable** is false.

See Also **CapHasVatTable** Property.

printDuplicateReceipt Method

Syntax **printDuplicateReceipt ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Prints a duplicate of a buffered transaction.

This method is only supported if **CapDuplicateReceipt** is true. This method will succeed if both the **CapDuplicateReceipt** and **DuplicateReceipt** properties are true.

This method resets the **DuplicateReceipt** property to false.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	The Fiscal Printer does not support duplicate receipts (see the **CapDuplicateReceipt** property) or there is no buffered transaction to print (see **DuplicateReceipt** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Monitor state. *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper.

See Also **CapDuplicateReceipt** Property, **DuplicateReceipt** Property.

printFiscalDocumentLine Method

Syntax **printFiscalDocumentLine (documentLine: *string*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
documentLine	String to be printed on the fiscal slip.

Remarks Prints a line of fiscal text to the slip station.

This method is only supported if **CapSlpFiscalDocument** is true.
This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	The Fiscal Printer does not support fiscal documents (see the **CapSlpFiscalDocument** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Document state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.)

See Also **beginFiscalDocument** Method, **endFiscalDocument** Method.

printFixedOutput Method

Syntax **printFixedOutput (documentType: *int32*, lineNumber: *int32*, data: *string*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
documentType	Identifier of a document stored in the Fiscal Printer
lineNumber	Number of the line in the document to print.
data	String parameter for placement in printed line.

Remarks Prints a line of a fixed document to the print station specified in the **beginFixedOutput** method. Each call prints a single line from a document by merging the stored text with the parameter *data*. Within a document lines must be printed sequentially. First and last lines are required; others may be optional. This method is only supported if **CapFixedOutput** is true. The Fiscal Printer state is set to FPTR_PS_FIXED_OUTPUT. This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	The Fiscal Printer does not support fixed output (see the **CapFixedOutput** property) or the lineNumber is invalid.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not in the Fixed Output state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station was specified but is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.)

See Also **beginFixedOutput** Method, **endFixedOutput** Method

printNormal Method *Updated in Release 1.7*

Syntax **printNormal (station: *int32*, data: *string*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
station	The Fiscal Printer station to be used. May be FPTR_S_RECEIPT, FPTR_S_JOURNAL, or FPTR_S_SLIP.
data[1]	The characters to be printed. May consist mostly of printable characters, escape sequences, carriage returns (13 decimal), and line feeds (10 decimal) but in many cases these are not supported.

Remarks Performs non-fiscal printing. Prints *data* on the Fiscal Printer *station*.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Special character values within *data* are:

Value	Meaning
Line Feed (10 decimal)	Print any data in the line buffer, and feed to the next print line. (A Carriage Return is not required in order to cause the line to be printed.)
Carriage Return (13 decimal)	If a Carriage Return immediately precedes a Line Feed, or if the line buffer is empty, then it is ignored. Otherwise, the line buffer is printed and the Fiscal Printer does not feed to the next print line. On some Fiscal Printers, print without feed may be directly supported. On others, a print may always feed to the next line, in which case the Device will print the line buffer and perform a reverse line feed if supported. If the Fiscal Printer does not support either of these features, then Carriage Return acts like a Line Feed.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The specified *station* does not exist. (See the **CapJrnPresent**, **CapRecPresent** and **CapSlpPresent** properties.)
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)

1. In the **OPOS** environment, the format of *data* depends upon the value of the **BinaryConversion** property. See **BinaryConversion** property on page A-29.

E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Non-Fiscal state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station was specified but is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station was specified but is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.)

See Also **beginNonFiscal** Method, **endNonFiscal** Method, **AsyncMode** Property.

printPeriodicTotalsReport Method

Syntax **printPeriodicTotalsReport (date1: *string*, date2: *string*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
date1	Starting date of report to print.
date2	Ending date of report to print.

Remarks Prints a report of totals for a range of dates on the receipt.
This method is always performed synchronously.

The dates are strings in the format "ddmmyyyyhhmm", where:

dd	day of the month (1 - 31)
mm	month (1 - 12)
yyyy	year (1997-)
hh	hour (0-23)
mm	minutes (0-59)

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition. *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. *ErrorCodeExtended* = EFPTR_BAD_DATE: One of the date parameters is invalid.

printPowerLossReport Method

Syntax **printPowerLossReport ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Prints on the receipt a report of a power failure that resulted in a loss of data stored in the CMOS of the Fiscal Printer.

This method is only supported if **CapPowerLossReport** is true.

This method is always performed synchronously.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20**.**

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support power loss reports (see the **CapPowerLossReport** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper.

See Also **CapPowerLossReport** Property.

printRecCash Method *Added in Release 1.6*

Syntax **printRecCash (amount:** ***currency*** **):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
amount	Amount to be incremented or decremented.

Remarks Prints a cash-in or cash-out receipt amount on the station defined by the **FiscalReceiptStation** property.

This method is only allowed if **CapFiscalReceiptType** is true and the **FiscalReceiptType** property is set to FPTR_RT_CASH_IN or FPTR_RT_CASH_OUT and the fiscal Fiscal Printer is in the Fiscal Receipt state.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	The Fiscal Printer does not support this method.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.)

See Also **beginFiscalReceipt** Method, **FiscalReceiptStation** Property, **FiscalReceiptType** Property.

printRecItem Method *Updated in Release 1.6*

Syntax **printRecItem (description: *string*, price: *currency*, quantity: *int32*, vatInfo: *int32*, unitPrice: *currency*, unitName: *string*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
description	Text describing the item sold.
price	Price of the line item.
quantity	Number of items. If zero, a single item is assumed.
vatInfo	VAT rate identifier or amount. If not used a zero must be transferred.
unitPrice	Price of each item. If not used a zero must be transferred.
unitName	Name of the unit i.e., “kg” or “ltr” or “pcs”. If not used an empty string (“”) must be transferred

Remarks Prints a receipt item for a sold item on the station specified by the **FiscalReceiptStation** property. If the *quantity* parameter is zero, then a single item quantity will be assumed.

Minimum parameters are *description* and *price* or *description*, *price*, *quantity*, and *unitPrice*. Most countries require *quantity* and *vatInfo* and some countries also require *unitPrice* and *unitName*.

VatInfo parameter contains a VAT table identifier if **CapHasVatTable** is true. Otherwise, it contains a VAT amount.

If **CapPostPreLine** is true additional application specific lines defined by the **PostLine** and **PreLine** properties will be printed. After printing these lines **PostLine** and **PreLine** will be reset to an empty string.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

Some possible values of the exception’s *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_JRN_EMPTY:
The journal station is out of paper.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_REC_EMPTY:
The receipt station is out of paper.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_SLP_EMPTY:
The slip station was specified, but a form is not inserted.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended =
EFPTR_BAD_ITEM_QUANTITY:
The quantity is invalid.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_BAD_PRICE:
The unit price is invalid.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended =
EFPTR_BAD_ITEM_DESCRIPTION:
The discount description is too long or contains a reserved word.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_BAD_VAT:
The VAT parameter is invalid.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended =
EFPTR_RECEIPT_TOTAL_OVERFLOW:
The receipt total has overflowed.
(Only applies if **AsyncMode** is false.)

See Also **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRec…** Methods, **AmountDecimalPlaces** Property, **FiscalReceiptStation** Property, **PostLine** Property, **PreLine** Property.

printRecItemAdjustment Method *Updated in Release 1.11*

Syntax **printRecItemAdjustment (adjustmentType: *int32*, description: *string*, amount: *currency*, vatInfo: *int32*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
adjustmentType	Type of adjustment. See below for values.
description	Text describing the adjustment.
amount	Amount of the adjustment.
vatInfo	VAT rate identifier or amount.

The *adjustmentType* parameter has one of the following values *(Note: If currency value, four decimal places are used)*:

Value	Meaning
FPTR_AT_AMOUNT_DISCOUNT	Fixed amount discount. The *amount* parameter contains a currency value.
FPTR_AT_AMOUNT_SURCHARGE	Fixed amount surcharge. The *amount* parameter contains a currency value.
FPTR_AT_PERCENTAGE_DISCOUNT	Percentage discount. The *amount* parameter contains a percentage value.
FPTR_AT_PERCENTAGE_SURCHARGE	Percentage surcharge. The *amount* parameter contains a percentage value.
FPTR_AT_COUPON_AMOUNT_DISCOUNT	Fixed amount discount for an advertising coupon. The *amount* parameter contains a currency value. The coupon is registered by the fiscal memory. If coupons are not registered at fiscal memory separately from ordinary discounts in the actual country then it is recommend to use FPTR_AT_AMOUNT_DISCOUNT instead.
FPTR_AT_COUPON_PERCENTAGE_DISCOUNT	Percentage discount for an advertising coupon. The *amount* parameter contains a percentage value. The coupon is registered by the fiscal memory. If coupons are not registered at fiscal memory separately from ordinary discounts in the actual country then it is recommend to use FPTR_AT_PERCENTAGE_DISCOUNT instead.

Remarks Applies and prints a discount or a surcharge to the last receipt item sold on the station specified by the **FiscalReceiptStation** property. This discount may be either a fixed currency amount or a percentage amount relating to the last item.

If **CapOrderAdjustmentFirst** is true, the method must be called before the corresponding **printRecItem** method. If **CapOrderAdjustmentFirst** is false, the method must be called after the **printRecItem**.

This discount/surcharge may be either a fixed currency amount or a percentage amount relating to the last item. If the discount amount is greater than the receipt

subtotal, an error occurs since the subtotal can never be negative. In many countries discount operations cause the printing of a fixed line of text expressing the kind of operation that has been performed.

The *VatInfo* parameter contains a VAT table identifier if **CapHasVatTable** is true. Otherwise, it contains a VAT amount.

Fixed amount discounts/surcharges are only supported if the property **CapAmountAdjustment** is true. Percentage discounts are only supported if **CapPercentAdjustment** is true.

If **CapPostPreLine** is true an additional application specific line defined by the **PreLine** property will be printed. After printing this line **PreLine** will be reset to an empty string.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors

A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	One of the following errors occurred: • The Fiscal Printer does not support fixed amount adjustments (see the **CapAmountAdjustment** property). • The Fiscal Printer does not support percentage discounts (see the **CapPercentAdjustment** property). • The *adjustmentType* parameter is invalid.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = FPTR_BAD_ITEM_AMOUNT: The discount amount is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION:

The discount description is too long or contains a reserved word. (Only applies if **AsyncMode** is false.)
ErrorCodeExtended = EFPTR_BAD_VAT:
The VAT parameter is invalid.
(Only applies if **AsyncMode** is false.)

See Also **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRec…** Methods, **AmountDecimalPlaces** Property, **FiscalReceiptStation** Property, **PreLine** Property.

printRecItemAdjustmentVoid Method *Added in Release 1.11*

Syntax **printRecItemAdjustmentVoid (adjustmentType: *int32*, description: *string*, amount: *currency*, vatInfo: *int32*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
adjustmentType	Type of adjustment to be voided. See below for values.
description	Text describing the adjustment to be voided.
amount	Amount of the adjustment to be voided.
vatInfo	VAT rate identifier or amount.

The *adjustmentType* parameter has one of the following values *(Note: If currency value, four decimal places are used)*:

Value	Meaning
FPTR_AT_AMOUNT_DISCOUNT	Fixed amount discount to be voided. The *amount* parameter contains a currency value.
FPTR_AT_AMOUNT_SURCHARGE	Fixed amount surcharge to be voided. The *amount* parameter contains a currency value.
FPTR_AT_PERCENTAGE_DISCOUNT	Percentage discount to be voided. The *amount* parameter contains a percentage value.
FPTR_AT_PERCENTAGE_SURCHARGE	Percentage surcharge to be voided. The *amount* parameter contains a percentage value.
FPTR_AT_COUPON_AMOUNT_DISCOUNT	Fixed amount discount for an advertising coupon to be voided. The *amount* parameter contains a currency value. The coupon is registered by the fiscal memory. If coupons are not registered at fiscal memory separately from ordinary discounts in the actual country then it is recommend to use FPTR_AT_AMOUNT_DISCOUNT instead.
FPTR_AT_COUPON_PERCENTAGE_DISCOUNT	Percentage discount for an advertising coupon to be voided. The *amount* parameter contains a percentage value. The coupon is registered by the fiscal memory. If coupons are not registered at fiscal memory separately from ordinary discounts in the actual country then it is recommend to use FPTR_AT_PERCENTAGE_DISCOUNT instead.

Remarks Cancels an adjustment that has been added to fiscal receipt before and prints a cancellation line with a negative amount on the station specified by the **FiscalReceiptStation** property. This adjustment cancellation amount may be either a fixed currency amount or a percentage amount.

The *VatInfo* parameter contains a VAT table identifier if **CapHasVatTable** is true. Otherwise, it contains a VAT amount.

Fixed amount adjustment cancellations are only supported if the property **CapAmountAdjustment** is true. Percentage adjustment cancellations are only supported if **CapPercentAdjustment** is true.

If **CapPostPreLine** is true an additional application specific line defined by the **PreLine** property will be printed. After printing this line **PreLine** will be reset to an empty string.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	One of the following errors occurred: • The Fiscal Printer does not support fixed amount adjustments (see the **CapAmountAdjustment** property). • The Fiscal Printer does not support percentage discounts (see the **CapPercentAdjustment** property). • The *adjustmentType* parameter is invalid.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = FPTR_BAD_ITEM_AMOUNT: The discount amount is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The discount description is too long or contains a

reserved word. (Only applies if **AsyncMode** is false.)
ErrorCodeExtended = EFPTR_BAD_VAT:
The VAT parameter is invalid.
(Only applies if **AsyncMode** is false.)

See Also **AmountDecimalPlaces** Property, **FiscalReceiptStation** Property, **PreLine** Property, **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRec…** Methods, **printRecItemAdjustment** Method.

printRecItemFuel Method *Added in Release 1.6*

Syntax **printRecItemFuel (description: *string*, price: *currency*, quantity: *int32*, vatInfo: *int32*, unitPrice: *currency*, unitName: *string*, specialTax: *currency*, specialTaxName: *string*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
description	Text describing the fuel product.
price	Price of the fuel item.
quantity	Number of items. If zero, a single item is assumed.
vatInfo	VAT rate identifier or amount. If not used a zero must be transferred.
unitPrice	Price of the fuel item per volume.
unitName	Name of the volume unit, i.e., “ltr”. If not used an empty string (“”) must be transferred
specialTax	Special tax amount, e.g., road tax. If not used a zero must be transferred.
specialTaxName	Name of the special tax.

Remarks Prints a receipt fuel item on the station specified by the **FiscalReceiptStation** property. *vatInfo* parameter contains a VAT table identifier if **CapHasVatTable** is true. Otherwise, it contains a VAT amount.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

Some possible values of the exception’s *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	This method is not supported.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_JRN_EMPTY:
The journal station is out of paper.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_REC_EMPTY:
The receipt station is out of paper.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_SLP_EMPTY:
The slip station was specified, but a form is not inserted.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended =
EFPTR_BAD_ITEM_QUANTITY:
The quantity is invalid.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_BAD_PRICE:
The unit price is invalid.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended =
EFPTR_BAD_ITEM_DESCRIPTION:
The discount description is too long or contains a reserved word.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_BAD_VAT:
The VAT parameter is invalid.
(Only applies if **AsyncMode** is false.)

ErrorCodeExtended =
EFPTR_RECEIPT_TOTAL_OVERFLOW:
The receipt total has overflowed.
(Only applies if **AsyncMode** is false.)

See Also **beginFiscalReceipt** Method, **FiscalReceiptStation** Property.

printRecItemFuelVoid Method *Added in Release 1.6*

Syntax **printRecItemFuelVoid (description: *string*, price: *currency*, vatInfo: *int32*, specialTax: *currency*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
description	Text describing the fuel product.
price	Price of the fuel item. If not used a zero must be transferred.
vatInfo	VAT rate identifier or amount. If not used a zero must be transferred.
specialTax	Special tax amount, e.g., road tax. If not used a zero must be transferred.

Remarks Called to void a fuel item on the station specified by the **FiscalReceiptStation** property.

If **CapOnlyVoidLastItem** is true, only the last fuel item transferred to the Fiscal Printer can be voided.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	This method is not supported.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_PRICE: The price is invalid. (Only applies if **AsyncMode** is false.)

ErrorCodeExtended = EFPTR_BAD_ITEM_DESCRIPTION:
The discount description is too long or contains a reserved word.
(Only applies if **AsyncMode** is false.)
ErrorCodeExtended = EFPTR_BAD_VAT:
The VAT parameter is invalid.
(Only applies if **AsyncMode** is false.)

See Also **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRecItemFuel** Method, **CapOnlyVoidLastItem** Property, **FiscalReceiptStation** Property.

printRecItemRefund Method *Added in Release 1.12*

Syntax **printRecItemRefund (description:** ***string*****, amount:** ***currency*****, quantity:** ***int32*****, vatInfo:** ***int32*****, unitAmount:** ***currency*****, unitName:** ***string*** **): void { raises-exception, use after open-claim-enable }**

Parameter	Description
description	Text describing the refund.
amount	The amount of the refund line.
quantity	Number of items. If zero, a single item is assumed.
vatInfo	VAT rate identifier or amount. If not used a zero must be transferred.
unitAmount	Amount of each refund item. If not used a zero must be transferred.
unitName	Name of the unit i.e., “kg” or “ltr” or “pcs”. If not used an empty string (“”) must be transferred

Remarks Processes one or more item refunds. The *amount* is positive, but it is printed as a negative number and the totals registers are decremented.

If *unitAmount* and *quantity* are non zero then the *amount* parameter corresponds to the product of *quantity* and *unitAmount*. Otherwise this method has the same functionality as the method **printRecRefund**.

Some fixed text, along with the *description*, will be printed on the station defined by the **FiscalReceiptStation** property to indicate that a refund has occurred.

The *vatInfo* parameter contains a VAT table identifier if **CapHasVatTable** is true. Otherwise it, contains a VAT amount.

If **CapPostPreLine** is true an additional application specific line defined by the **PreLine** property will be printed. After printing this line, **PreLine** will be reset to an empty string.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_QUANTITY: The quantity is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_PRICE: The unit price is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: The refund amount is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The discount description is too long or contains a reserved word. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_VAT: The VAT parameter is invalid. (Only applies if **AsyncMode** is false.)

See Also **CapHasVatTable** Property, **CapPostPreLine** Property, **FiscalReceiptStation** Property, **PreLine** Property, **printRecItemRefundVoid** Method, **printRecRefund** Method.

printRecItemRefundVoid Method *Added in Release 1.12*

Syntax **printRecItemRefundVoid (description: *string*, amount: *currency*, quantity: *int32*, vatInfo: *int32*, unitAmount: *currency*, unitName: *string*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
description	Text describing the refund.
amount	The amount of the refund line.
quantity	Number of items. If zero, a single item is assumed.
vatInfo	VAT rate identifier or amount. If not used a zero must be transferred.
unitAmount	Amount of each refund item. If not used a zero must be transferred.
unitName	Name of the unit i.e., “kg” or “ltr” or “pcs”. If not used an empty string (“”) must be transferred

Remarks Processes a void of one or more item refunds. The *amount* is positive and the totals registers are incremented.

If *unitAmount* and *quantity* are non zero then the *amount* parameter corresponds to the product of *quantity* and *unitAmount*. Otherwise this method has the same functionality as the method **printRecRefundVoid**.

Some fixed text, along with the *description*, will be printed on the station defined by the **FiscalReceiptStation** property to indicate that a void of a refund has occurred.

The *vatInfo* parameter contains a VAT table identifier if **CapHasVatTable** is true. Otherwise it, contains a VAT amount.

If **CapOnlyVoidLastItem** is true, only the last refund item transferred to the Fiscal Printer can be voided.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

Some possible values of the exception’s *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	Cancelling is not allowed at this ticket state. May be because no item has been sold previously. (See **CapOnlyVoidLastItem**.)
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state.

ErrorCodeExtended = EFPTR_COVER_OPEN:
The Fiscal Printer cover is open.
(Only applies if **AsyncMode** is false.)
ErrorCodeExtended = EFPTR_JRN_EMPTY:
The journal station is out of paper.
(Only applies if **AsyncMode** is false.)
ErrorCodeExtended = EFPTR_REC_EMPTY:
The receipt station is out of paper.
(Only applies if **AsyncMode** is false.)
ErrorCodeExtended = EFPTR_SLP_EMPTY:
The slip station was specified, but a form is not inserted.
(Only applies if **AsyncMode** is false.)
ErrorCodeExtended =
EFPTR_BAD_ITEM_QUANTITY:
The quantity is invalid.
(Only applies if **AsyncMode** is false.)
ErrorCodeExtended = EFPTR_BAD_PRICE:
The unit price is invalid.
(Only applies if **AsyncMode** is false.)
ErrorCodeExtended =
EFPTR_BAD_ITEM_AMOUNT:
The refund amount is invalid.
(Only applies if **AsyncMode** is false.)
ErrorCodeExtended =
EFPTR_BAD_ITEM_DESCRIPTION:
The discount description is too long or contains a reserved word.
(Only applies if **AsyncMode** is false.)
ErrorCodeExtended = EFPTR_BAD_VAT:
The VAT parameter is invalid.
(Only applies if **AsyncMode** is false.)
ErrorCodeExtended =
EFPTR_RECEIPT_TOTAL_OVERFLOW:
The receipt total has overflowed.
(Only applies if **AsyncMode** is false.)

See Also **CapHasVatTable** Property, **CapPostPreLine** Property, **FiscalReceiptStation** Property, **PreLine** Property, **printRecItemRefund** Method, **printRecRefundVoid** Method.

printRecItemVoid Method *Added in Release 1.11*

Syntax **printRecItemVoid (description: *string*, price: *currency*, quantity: *int32*, vatInfo: *int32*, unitPrice: *currency*, unitName: *string*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
description	Text describing the item to be voided.
price	Price of the item to be voided.
quantity	Quantity of item to be voided. If zero, a single item is assumed.
vatInfo	VAT rate identifier or amount. If not used a zero must be transferred.
unitPrice	Price of each item. If not used a zero must be transferred.
unitName	Name of the unit i.e., “kg” or “ltr” or “pcs”. If not used an empty string (“”) must be transferred

Remarks Cancels one or more items that has been added to the receipt and prints a void description on the station defined by the **FiscalReceiptStation** property.

Minimum parameters are *description* and *price* or *description*, *quantity*, and *unitPrice*. Most countries require *quantity* and *vatInfo* and some countries also require *unitPrice* and *unitName*.

price is a positive number, it will be printed as a negative and will be decremented from the totals registers. In some countries *price* will be ignored, instead the computation from *quantity* and *unitPrice* will be printed as a negative amount. The *vatInfo* parameter contains a VAT table identifier if **CapHasVatTable** is true. Otherwise, it contains a VAT amount.

If **CapOnlyVoidLastItem** is true, only the last item transferred to the Fiscal Printer can be voided exclusive an adjustment line for this item.

If **CapPostPreLine** is true, additional application specific lines defined by the **PostLine** and **PreLine** properties will be printed. After printing these lines **PostLine** and **PreLine** will be reset to an empty string.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

Some possible values of the exception’s *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	Cancelling is not allowed at this ticket state. May be because no item has been sold previously. (See **CapOnlyVoidLastItem**.)

E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: The *price* is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_QUANTITY: The *quantity* is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_VAT: The VAT information is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The *description* is too long or contains a reserved word. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_NEGATIVE_TOTAL: The computed total is less than zero. (Only applies if **AsyncMode** is false.)

See Also **AmountDecimalPlaces** Property, **CapOnlyVoidLastItem** Property, **FiscalReceiptStation** Property, **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRecItem** Method, **printRec…** Methods.

printRecMessage Method *Updated in Release 1.13*

Syntax **printRecMessage (message: *string*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
message	Text message to print.

Remarks Prints a message on the fiscal receipt on the station specified by the **FiscalReceiptStation** property. The length of an individual message is limited to the number of characters given in the **MessageLength** property. The kind of message to be printed is defined by the **MessageType** property**.**

This method is only supported if **CapAdditionalLines** is true. This method is only supported when the Fiscal Printer is in one of the Fiscal Receipt states.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not in the Fiscal Receipt, Fiscal Receipt total, or Fiscal Receipt Ending state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The message is too long or contains a reserved word. (Only applies if **AsyncMode** is false.)

See Also **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRec…** Methods, **CapAdditionalLines** Property, **FiscalReceiptStation** Property, **MessageLength** Property, **MessageType** Property.

printRecNotPaid Method *Updated in Release 1.11*

Syntax **printRecNotPaid (description:** ***string*****, amount:** ***currency*** **):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
description	Text describing the not paid amount.
amount	Amount not paid.

Remarks Indicates a part of the receipt's total to not be paid.

Some fixed text, along with the *description*, will be printed on the station defined by the **FiscalReceiptStation** property to indicate that part of the receipt total has not been paid. This method is only supported if **CapReceiptNotPaid** is true. If this method is successful, the **PrinterState** property will remain in FPTR_PS_FISCAL_RECEIPT_TOTAL state or change to the value FPTR_PS_FISCAL_RECEIPT_ENDING depending upon whether the entire receipt total is now accounted for or not. This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in either the Fiscal Receipt or Fiscal Receipt Total state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The *description* is too long or contains a reserved word. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: The *amount* is invalid. (Only applies if **AsyncMode** is false.)

See Also **AmountDecimalPlaces** Property, **CapReceiptNotPaid** Property, **FiscalReceiptStation** Property, **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRec…** Methods.

printRecPackageAdjustment Method *Added in Release 1.6*

Syntax **printRecPackageAdjustment (adjustmentType: *int32*, description: *string*, vatAdjustment: *string*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
adjustmentType	Type of adjustment. See below for values.
description	Text describing the adjustment.
vatAdjustment	String containing a list of adjustment(s) for different Vat(s).

The *adjustmentType* parameter has one of the following values:

Value	Meaning
FPTR_AT_DISCOUNT	Discount.
FPTR_AT_SURCHARGE	Surcharge.

The *vatAdjustment* parameter consists of ASCII numeric semicolon delimited pairs of values which denote each the VAT identifier of the package item to be adjusted and adjustment amount, separated by a comma.

The number of pairs is delimited by the **NumVatRates** property.

Remarks Called to give an adjustment for a package of some items booked before. This adjustment (discount/surcharge) may be either a fixed currency amount or a percentage amount relating to items combined to an adjustment package.

Each item of the package must be transferred before.

Fixed amount adjustments are only supported if **CapPackageAdjustment** is true.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	The Fiscal Printer does not support package adjustments (see the **CapPackageAdjustment** property), or the *adjustmentType* parameter is invalid.

E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The *description* is too long or contains a reserved word. (Only applies if **AsyncMode** is false.)

See Also **printRecPackageAdjustVoid** Method, **CapPackageAdjustment** Property.

printRecPackageAdjustVoid Method *Added in Release 1.6*

Syntax **printRecPackageAdjustVoid (adjustmentType: *int32*,**
vatAdjustment: *string*):
void { raises-exception, use after open-claim-enable }

Parameter	Description
adjustmentType	Type of adjustment. See below for values.
vatAdjustment	String containing a list of adjustment(s) to be voided for different VAT(s).

The *adjustmentType* parameter has one of the following values:

Value	Meaning
FPTR_AT_DISCOUNT	Discount.
FPTR_AT_SURCHARGE	Surcharge.

The *vatAdjustment* parameter consists of ASCII numeric semicolon delimited pairs of values which denote each the VAT identifier of the package item to be adjusted and adjustment amount, separated by a comma.

The number of pairs is delimited by the **NumVatRates** property.

Remarks Called to void the adjustment for a package of some items. This adjustment (discount/surcharge) may be either a fixed currency amount or a percentage amount relating to the current receipt subtotal.

Fixed amount void adjustments are only supported if **CapPackageAdjustment** is true.

If **CapPostPreLine** is true an additional application specific line defined by the **PreLine** property will be printed. After printing this line **PreLine** will be reset to an empty string.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	The Fiscal Printer does not support package adjustments (see the **CapPackageAdjustment** property), or the *adjustmentType* parameter is invalid.

E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The *description* is too long or contains a reserved word. (Only applies if **AsyncMode** is false.)

See Also **printRecPackageAdjustment** Method, **CapPackageAdjustment** Property, **PreLine** Property.

printRecRefund Method *Updated in Release 1.12*

Syntax **printRecRefund (description:** ***string*****, amount:** ***currency*****, vatInfo:** ***int32*** **): void { raises-exception, use after open-claim-enable }**

Parameter	Description
description	Text describing the refund.
amount	Amount of the refund.
vatInfo	VAT rate identifier or amount.

Remarks Processes a refund. The *amount* is positive, but it is printed as a negative number and the totals registers are decremented.

Some fixed text, along with the *description*, will be printed on the station defined by the **FiscalReceiptStation** property to indicate that a refund has occurred.

The *vatInfo* parameter contains a VAT table identifier if **CapHasVatTable** is true. Otherwise it, contains a VAT amount.

If **CapPostPreLine** is true an additional application specific line defined by the **PreLine** property will be printed. After printing this line **PreLine** will be reset to an empty string.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

If several items of the same item type are to be refunded, then it is recommended to use **printRecItemRefund**.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)

E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The *description* is too long or contains a reserved word. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: The *amount* is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_VAT: The VAT information is invalid. (Only applies if **AsyncMode** is false.)

See Also **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRec…** Methods, **AmountDecimalPlaces** Property, **FiscalReceiptStation** Property, **PreLine** Property, **printRecItemRefund** Method.

printRecRefundVoid Method *Updated in Release 1.12*

Syntax **printRecRefundVoid (description:** ***string*****, amount:** ***currency*****,**
vatInfo: ***int32*** **):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
description	Text describing the refund.
amount	Amount of the voided refund.
vatInfo	VAT rate identifier or amount.

Remarks Called to process a void of a refund.

The *amount* is positive and the totals registers are incremented.

Some fixed text, along with the *description*, will be printed on the station defined by the **FiscalReceiptStation** property to indicate that a void of a refund has occurred.

The *vatInfo* parameter contains a VAT table identifier if **CapHasVatTable** is true. Otherwise it, contains a VAT amount.

If **CapOnlyVoidLastItem** is true, only the last refund item transferred to the Fiscal Printer can be voided.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

If the refund of several items of the same item type is to be voided, then it is recommended to use **printRecItemRefundVoid**.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)

E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The *description* is too long or contains a reserved word. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: The *amount* is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_VAT: The VAT information is invalid. (Only applies if **AsyncMode** is false.)

See Also **printRecRefund** Method, **printRecItemRefundVoid** Method, **FiscalReceiptStation** Property.

printRecSubtotal Method *Updated in Release 1.6*

Syntax **printRecSubtotal (amount:** ***currency*** **):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
amount	Amount of the subtotal.

Remarks Checks and prints the current receipt subtotal on the station defined by the **FiscalReceiptStation** property.

If **CapCheckTotal** is true, the *amount* is compared to the subtotal calculated by the Fiscal Printer. If the subtotals match, the subtotal is printed on the station defined by the **FiscalReceiptStation** property. If the results do not match, the receipt is automatically canceled. If **CapCheckTotal** is false, then the subtotal is printed on the station defined by the **FiscalReceiptStation** property and the parameter is never compared to the subtotal computed by the Fiscal Printer.

If **CapPostPreLine** is true an additional application specific line defined by the **PostLine** property will be printed. After printing this line **PostLine** will be reset to an empty string.

If this method compares the application's subtotal with the Fiscal Printer's subtotal and they do not match, the **PrinterState** property will be changed to FPTR_PS_FISCAL_RECEIPT_ENDING.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)

E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: The subtotal from the application does not match the subtotal computed by the Fiscal Printer. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_NEGATIVE_TOTAL: The total computed by the Fiscal Printer is less than zero. (Only applies if **AsyncMode** is false.)

See Also **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRec…** Methods, **AmountDecimalPlaces** Property, **FiscalReceiptStation** Property, **PostLine** Property.

printRecSubtotalAdjustment Method *Updated in Release 1.11*

Syntax **printRecSubtotalAdjustment (adjustmentType: *int32*, description: *string*, amount: *currency*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
adjustmentType	Type of adjustment. See below for values.
description	Text describing the discount or surcharge.
amount	Amount of the adjustment (discount or surcharge).

The *adjustmentType* parameter has one of the following values *(Note: If currency value, four decimal places are used)*:

Value	Meaning
FPTR_AT_AMOUNT_DISCOUNT	Fixed amount discount. The *amount* parameter contains a currency value.
FPTR_AT_AMOUNT_SURCHARGE	Fixed amount surcharge. The *amount* parameter contains a currency value.
FPTR_AT_PERCENTAGE_DISCOUNT	Percentage discount. The *amount* parameter contains a percentage value.
FPTR_AT_PERCENTAGE_SURCHARGE	Percentage surcharge. The *amount* parameter contains a percentage value.
FPTR_AT_COUPON_AMOUNT_DISCOUNT	Fixed amount discount for an advertising coupon. The *amount* parameter contains a currency value. The coupon is registered by the fiscal memory. If coupons are not registered at fiscal memory separately from ordinary discounts in the actual country then it is recommend to use FPTR_AT_AMOUNT_DISCOUNT instead.
FPTR_AT_COUPON_PERCENTAGE_DISCOUNT	Percentage discount for an advertising coupon. The *amount* parameter contains a percentage value. The coupon is registered by the fiscal memory. If coupons are not registered at fiscal memory separately from ordinary discounts in the actual country then it is recommend to use FPTR_AT_PERCENTAGE_DISCOUNT instead.

Remarks Applies and prints a discount/surcharge to the current receipt subtotal on the station defined by the **FiscalReceiptStation** property. This discount/surcharge may be either a fixed currency amount or a percentage amount relating to the current receipt subtotal.

If the discount/surcharge amount is greater than the receipt subtotal, an error occurs since the subtotal can never be negative.

In many countries discount/surcharge operations cause the printing of a fixed line of text expressing the kind of operation that has been performed.

Fixed amount discounts are only supported if **CapSubAmountAdjustment** is true. Percentage discounts are only supported if **CapSubPercentAdjustment** is true. Surcharges are only supported if **CapPositiveSubtotalAdjustment** is true.

If **CapPostPreLine** is true an additional application specific line defined by the **PreLine** property will be printed. After printing this line **PreLine** will be reset to an empty string.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors

A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	One of the following errors occurred: • Fixed amount discounts are not supported (see the **CapSubAmountAdjustment** property). • Percentage discounts are not supported (see the **CapSubPercentAdjustment** property). • Surcharges are not supported (see the **CapPositiveSubtotalAdjustment** property). • The *adjustmentType* parameter is invalid.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: The discount *amount* is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The discount *description* is too long or contains a reserved word. (Only applies if **AsyncMode** is false.)

See Also

beginFiscalReceipt Method, **endFiscalReceipt** Method, **printRec…** Methods, **AmountDecimalPlaces** Property, **CapPositiveSubtotalAdjustment** Property, **FiscalReceiptStation** Property, **PreLine** Property.

printRecSubtotalAdjustVoid Method *Added in Release 1.6*

Syntax **printRecSubtotalAdjustVoid (adjustmentType:** ***int32*****,**
amount: ***currency*** **):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
adjustmentType	Type of adjustment. See below for values.
amount	Amount of the adjustment (discount or surcharge).

The *adjustmentType* parameter has one of the following values *(Note: If currency value, four decimal places are used)*:

Value	Meaning
FPTR_AT_AMOUNT_DISCOUNT	Fixed amount discount. The *amount* parameter contains a currency value.
FPTR_AT_AMOUNT_SURCHARGE	Fixed amount surcharge. The *amount* parameter contains a currency value.
FPTR_AT_PERCENTAGE_DISCOUNT	Percentage discount. The *amount* parameter contains a percentage value.
FPTR_AT_PERCENTAGE_SURCHARGE	Percentage surcharge. The *amount* parameter contains a percentage value.

Remarks Called to void a preceding subtotal adjustment on the station defined by the **FiscalReceiptStation** property. This discount/surcharge may be either a fixed currency amount or a percentage amount relating to the current receipt subtotal.

Fixed amount void discounts are only supported if **CapSubAmountAdjustment** is true. Percentage void discounts are only supported if the property **CapSubPercentAdjustment** is true.

If **CapPostPreLine** is true an additional application specific line defined by the **PreLine** property will be printed. After printing this line **PreLine** will be reset to an empty string.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors

A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	One of the following errors occurred: • Fixed amount discounts are not supported (see the **CapSubAmountAdjustment** property). • Percentage discounts are not supported (see the **CapSubPercentAdjustment** property). • The *adjustmentType* parameter is invalid.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: The discount *amount* is invalid. (Only applies if **AsyncMode** is false.)

See Also

beginFiscalReceipt Method, **endFiscalReceipt** Method, **printRec…** Methods, **AmountDecimalPlaces** Property, **FiscalReceiptStation** Property, **PreLine** Property.

printRecTaxID Method *Added in Release 1.6*

Syntax **printRecTaxID (taxId: *string*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
taxId	Customer identification with identification characters and tax number.

Remarks Called to print the customers tax identification on the station defined by the **FiscalReceiptStation** property.

This method is only supported when the Fiscal Printer is in the Fiscal Receipt Ending state.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	The Fiscal Printer does not support printing tax identifications.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt Ending state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.)

See Also **FiscalReceiptStation** Property.

printRecTotal Method *Updated in Release 1.14*

Syntax **printRecTotal (total:** ***currency*****, payment:** ***currency*****, description:** ***string*** **): void { raises-exception, use after open-claim-enable }**

Parameter	Description
total	Application computed receipt total.
payment	Amount of payment tendered.
description	Text description of the payment or the index of a predefined payment description.

Remarks Checks and prints the current receipt total on the station defined by the **FiscalReceiptStation** property and to tender a payment.

If **CapCheckTotal** is true, the *total* is compared to the total calculated by the Fiscal Printer. If the totals match, the total is printed on both the receipt and journal along with some fixed text. If the results do not match, the receipt is automatically canceled. If **CapCheckTotal** is false, then the total is printed on the receipt and journal and the parameter is never compared to the total computed by the Fiscal Printer.

If **CapPredefinedPaymentLines** is true, then the *description* parameter contains the index of one of the Fiscal Printer's predefined payment descriptions. The index is typically a single character of the alphabet. The set of allowed values for this index is to be described in the description of the service and stored in the **PredefinedPaymentLines** property.

If *payment* = *total*, a line containing the *description* and *payment* is printed. The **PrinterState** property will be set to FPTR_PS_FISCAL_RECEIPT_ENDING.

If *payment* > *total*, a line containing the *description* and *payment* is printed followed by a second line containing the change due. If **CapChangeDue** property is true, a description for the change due defined by the **ChangeDue** property is printed as the second line. The **PrinterState** property will be set to FPTR_PS_FISCAL_RECEIPT_ENDING.

If *payment* < *total*, a line containing the *description* and *payment* is printed. Since the entire receipt total has not yet been tendered, the **PrinterState** property will be set to FPTR_PS_FISCAL_RECEIPT_TOTAL.

If *payment* = *0*, no line containing the *description* and *payment* is printed. The **PrinterState** property will be set to FPTR_PS_FISCAL_RECEIPT_TOTAL.

If **CapAdditionalLines** is false, then receipt trailer lines, fiscal logotype and receipt cut are executed after the last total line, whenever receipt's total became equal to the payment from the application. Otherwise these lines are printed calling the **endFiscalReceipt** method.

If **CapPostPreLine** is true an additional application specific line defined by the **PostLine** property will be printed. After printing this line **PostLine** will be reset to an empty string.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: • The application computed total does not match the Fiscal Printer computed total, or • the *total* parameter is invalid, or • the *payment* parameter is invalid (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The *description* is too long or contains a reserved word. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_NEGATIVE_TOTAL: The computed total is less than zero. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_WORD_NOT_ALLOWED: The description contains the reserved word.

See Also **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRec…** Methods, **PredefinedPaymentLines** Property, **AmountDecimalPlaces** Property, **ChangeDue** Property, **FiscalReceiptStation** Property, **PostLine** Property.

printRecVoid Method *Updated in Release 1.6*

Syntax **printRecVoid (description: *string*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
description	Text describing the void.

Remarks Cancels the current receipt.

The receipt is annulled but it is not physically canceled from the Fiscal Printer's fiscal memory since fiscal receipts are printed with an increasing serial number and totals are accumulated in registers. When a receipt is canceled, its subtotal is subtracted from the totals registers, but it is added to the canceled receipt register.

Some fixed text, along with the *description*, will be printed on the station defined by the **FiscalReceiptStation** property to indicate that the receipt has been canceled.

Normally only a receipt with at least one transaction can be voided. If **CapEmptyReceiptIsVoidable** is true also an empty receipt (only the **beginFiscalReceipt** method was called) can be voided.

If this method is successful, the **PrinterState** property will be changed to FPTR_PS_FISCAL_RECEIPT_ENDING.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)

E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The description is too long or contains a reserved word. (Only applies if **AsyncMode** is false.)

See Also **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRec…** Methods **CapEmptyReceiptIsVoidable** Property, **FiscalReceiptStation** Property.

printRecVoidItem Method *Deprecated in Release 1.11*

Syntax **printRecVoidItem (description: *string*, amount: *currency*,
quantity: *int32*, adjustmentType: *int32*,
adjustment: *currency*, vatInfo: *int32*):
void { raises-exception, use after open-claim-enable }**

Parameter	Description
description	Text description of the item void.
amount	Amount of item to be voided.
quantity	Quantity of item to be voided.
adjustmentType	Type of adjustment. See below for values.
adjustment	Amount of the adjustment (discount or surcharge).
vatInfo	VAT rate identifier or amount.

The *adjustmentType* parameter has one of the following values *(Note: If currency value, four decimal places are used)*:

Value	Meaning
FPTR_AT_AMOUNT_DISCOUNT	Fixed amount discount. The *adjustment* parameter contains a currency value.
FPTR_AT_AMOUNT_SURCHARGE	Fixed amount surcharge. The *adjustment* parameter contains a currency value.
FPTR_AT_PERCENTAGE_DISCOUNT	Percentage discount. The *adjustment* parameter contains a percentage value.
FPTR_AT_PERCENTAGE_SURCHARGE	Percentage surcharge. The *adjustment* parameter contains a percentage value.

Remarks Cancels an item that has been added to the receipt and prints a void description on the station defined by the **FiscalReceiptStation** property.

amount is a positive number, it will be printed as a negative and will be decremented from the totals registers.

The *vatInfo* parameter contains a VAT table identifier if **CapHasVatTable** is true. Otherwise, it contains a VAT amount. Fixed amount discounts/surcharges are only supported if **CapAmountAdjustment** is true. Percentage discounts are only supported if **CapPercentAdjustment** is true.

If **CapOnlyVoidLastItem** is true, only the last item transferred to the Fiscal Printer can be voided.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further

information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress. (Only applies if **AsyncMode** is false.)
E_ILLEGAL	One of the following errors occurred: • Fixed amount adjustments are not supported (see the **CapAmountAdjustment** property), or • Percentage discounts are not supported (see the **CapPercentAdjustment** property), or • The *adjustmentType* parameter is invalid.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Fiscal Receipt state. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_SLP_EMPTY: The slip station was specified, but a form is not inserted. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_AMOUNT: The *amount* is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_QUANTITY: The *quantity* is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_VAT: The VAT information is invalid. (Only applies if **AsyncMode** is false.) *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The *description* is too long or contains a reserved word. (Only applies if **AsyncMode** is false. *ErrorCodeExtended* = EFPTR_NEGATIVE_TOTAL: The computed total is less than zero. (Only applies if **AsyncMode** is false.)

See Also **beginFiscalReceipt** Method, **endFiscalReceipt** Method, **printRec…** Methods, **CapOnlyVoidLastItem** Property, **AmountDecimalPlaces** Property, **FiscalReceiptStation** Property.

printReport Method *Updated in Release 1.11*

Syntax **printReport (reportType: *int32*, startNum: *string*, endNum: *string*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
reportType	The kind of report to print.
startNum	ASCII string identifying the starting record in Fiscal Printer memory from which to begin printing
endNum	ASCII string identifying the final record in Fiscal Printer memory at which printing is to end. See *reportType* table below to find out the exact meaning of this parameter.

The *reportType* parameter has one of the following values:

Value	Meaning
FPTR_RT_ORDINAL	Prints a report between two fiscal memory record numbers. If both *startNum* and *endNum* are valid and *endNum* > *startNum*, then a report of the period between *startNum* and *endNum* will be printed. If *startNum* is valid and *endNum* is zero, then a report relating only to *startNum* will be printed.
FPTR_RT_DATE	Prints a report between two dates. The dates are strings in the format "ddmmyyyyhhmm", where: dd day of the month (01 - 31) mm month (01 - 12) yyyy year (1997- ...) hh hour (00-23) mm minutes (00-59)
FPTR_RT_EOD_ORDINAL	Prints a report between two Z reports where *startNum* and *endNum* represent a Z report number. If both *startNum* and *endNum* are valid and *endNum* > *startNum*, then a report of the period between *startNum* and *endNum* will be printed. If *startNum* is valid and *endNum* is zero, then a report relating only to *startNum* will be printed.

Remarks Prints a report of the fiscal EPROM contents on the receipt that occurred between two end points.

This method is always performed synchronously.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot perform while output is in progress.
E_ILLEGAL	One of the following errors occurred: • The *reportType* parameter is invalid, or • One or both of s*tartNum* and *endNum* are invalid, or • s*tartNum* > *endNum*.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper.

printXReport Method

Syntax **printXReport ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Prints a report of all the daily fiscal activities on the receipt. No data will be written to the fiscal EPROM as a result of this method invocation.

This method is only supported if **CapXReport** is true. This method is always performed synchronously.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support X reports (see the **CapXReport** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition. *ErrorCodeExtended* = EFPTR_COVER_OPEN: The Fiscal Printer cover is open. *ErrorCodeExtended* = EFPTR_JRN_EMPTY: The journal station is out of paper. *ErrorCodeExtended* = EFPTR_REC_EMPTY: The receipt station is out of paper.

See Also **CapXReport** Property.

printZReport Method *Updated in Release 1.6*

Syntax **printZReport ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Prints a report of all the daily fiscal activities on the receipt. Data will be written to the fiscal EPROM as a result of this method invocation.

Since running **printZReport** is implicitly a fiscal end of day function, the **DayOpened** property will be set to false.

This method is always performed synchronously.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer's current state does not allow this state transition.
	ErrorCodeExtended = EFPTR_COVER_OPEN: The Fiscal Printer cover is open.
	ErrorCodeExtended = EFPTR_JRN_EMPTY: The journal station is out of paper.
	ErrorCodeExtended = EFPTR_REC_EMPTY: The receipt station is out of paper.

See Also **beginFiscalDocument** Method, **beginFiscalReceipt** Method, **DayOpened** Property.

resetPrinter Method

Syntax **resetPrinter ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Forces the Fiscal Printer to return to Monitor state. This forces any interrupted operations to be canceled and closed. This method must be invoked when the Fiscal Printer is not in a Monitor state after a successful call to the **claim** method and successful setting of the **DeviceEnabled** property to true. This typically happens if a power failures occurs during a fiscal operation.

Calling this method does not close the Fiscal Printer, i.e., does not force a Z report to be printed.

The Device will handle this command as follows:

- If the Fiscal Printer was in either Fiscal Receipt, Fiscal Receipt Total or Fiscal Receipt Ending state, the receipt will be ended without updating any registers.
- If the Fiscal Printer was in a non-fiscal state, the Fiscal Printer will exit that state.
- If the Fiscal Printer was in the training state, the Fiscal Printer will exit the training state.

This method is always performed synchronously.

Errors A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

setCurrency Method *Added in Release 1.6*

Syntax **setCurrency (newCurrency: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
newCurrency	The new currency.

The *newCurrency* parameter has one of the following values:

Value	Meaning
FPTR_SC_EURO	Change to the EURO currency.

Remarks Called to change to a new currency, e.g., EURO.

This method is only supported if **CapSetCurrency** is true and can only be called while **DayOpened** is false.

The actual currency is kept in the **ActualCurrency** property.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	One of the following errors occurred: • The Fiscal Printer does not support this method (see the **CapSetCurrency** property), or • The Fiscal Printer has already begun the fiscal day (see the **DayOpened** property), or • the specified *newCurrency* value is not valid.

See Also **ActualCurrency** Property, **CapSetCurrency** Property, **DayOpened** Property.

setDate Method

Syntax **setDate (date: *string*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
date	Date and time as a string.

Remarks Sets the Fiscal Printer's date and time.

The date and time is passed as a string in the format "ddmmyyyyhhmm", where:

dd	day of the month (1 - 31)
mm	month (1 - 12)
yyyy	year (1997-)
hh	hour (0-23)
mm	minutes (0-59)

This method can only be called while **DayOpened** is false.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer has already begun the fiscal day (see the **DayOpened** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_BAD_DATE: One of the entries of the *date* parameters is invalid.

See Also **DayOpened** Property.

setHeaderLine Method *Updated in Release 1.6*

Syntax **setHeaderLine (lineNumber: *int32*, text: *string*, doubleWidth: *boolean*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
lineNumber	Line number of the header line to set.
text	Text to which to set the header line.
doubleWidth	Print this line in double wide characters.

Remarks Sets one of the fiscal receipt header lines. The text set by this method will be stored by the Fiscal Printer and retained across power losses.

If **CapMultiContractor** property is true, header lines can be defined for different contractors specified by the **ContractorId** property.

The *lineNumber* parameter must be between 1 and the value of the **NumHeaderLines** property. If *text* is an empty string (“”), then the header line is unset and will not be printed. The *doubleWidth* characters will be printed if the Fiscal Printer supports them. See the **CapDoubleWidth** property to determine if they are supported. This method is only supported if **CapSetHeader** is true. This method can only be called while **DayOpened** is false.

Errors A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

Some possible values of the exception’s *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	One of the following errors occurred: • The Fiscal Printer does not support setting header lines (see the **CapSetHeader** property), or • The Fiscal Printer has already begun the fiscal day (see the **DayOpened** property), or • the *lineNumber* parameter was invalid.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The *text* parameter is too long or contains a reserved word.

See Also **CapDoubleWidth** Property, **CapMultiContractor** Property, **CapSetHeader** Property, **ContractorId** Property, **DayOpened** Property, **NumHeaderLines** Property.

setPOSID Method

Syntax **setPOSID (POSID: *string*, cashierID: *string*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
POSID	Identifier for the POS system.
cashierID	Identifier of the current cashier.

Remarks Sets the POS and cashier identifiers. These values will be printed when each fiscal receipt is closed.

This method is only supported if **CapSetPOSID** is true. This method can only be called while **DayOpened** is false.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	One of the following errors occurred: • The Fiscal Printer does not support setting the POS identifier (see the **CapSetPOSID** property), or • The printer has already begun the fiscal day (see the **DayOpened** property), or • Either the *POSID* or *cashierID* parameter is invalid.

See Also **CapSetPOSID** Property, **DayOpened** Property.

setStoreFiscalID Method

Syntax **setStoreFiscalID (ID: *string*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
ID	Fiscal identifier.

Remarks Sets the store fiscal ID. This value is retained by the Fiscal Printer even after power failures. This *ID* is automatically printed by the Fiscal Printer after the fiscal receipt header lines.

This method is only supported if **CapSetStoreFiscalID** is true. This method can only be called while **DayOpened** is false.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	One of the following errors occurred: • The Fiscal Printer does not support setting the store fiscal identifier (see the **CapSetStoreFiscalID** property), or • The Fiscal Printer has already begun the fiscal day (see the **DayOpened** property), or • The *ID* parameter was invalid.

See Also **CapSetStoreFiscalID** Property, **DayOpened** Property.

setTrailerLine Method

Syntax **setTrailerLine (lineNumber: *int32*, text: *string*, doubleWidth: *boolean*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
lineNumber	Line number of the trailer line to set.
text	Text to which to set the trailer line.
doubleWidth	Print this line in double wide characters.

Remarks Sets one of the fiscal receipt trailer lines. The text set by this method will be stored by the Fiscal Printer and retained across power losses.

The *lineNumber* parameter must be between 1 and the value of the **NumTrailerLines** property. If *text* is an empty string (“”), then the trailer line is unset and will not be printed. The *doubleWidth* characters will be printed if the Fiscal Printer supports them. See the **CapDoubleWidth** property to determine if they are supported. This method is only supported if **CapSetTrailer** is true. This method can only be called while **DayOpened** is false.

Errors A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

Some possible values of the exception’s *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	One of the following errors occurred: • The Fiscal Printer does not support setting the receipt trailer lines (see the **CapSetTrailer** property), or • The Fiscal Printer has already begun the fiscal day (see the **DayOpened** property), or • the *lineNumber* parameter was invalid.
E_EXTENDED	*ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The *text* parameter is too long or contains a reserved word.

See Also **CapDoubleWidth** Property, **CapSetTrailer** Property, **DayOpened** Property, **NumTrailerLines** Property.

setVatTable Method *Updated in Release 1.11*

Syntax **setVatTable ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Sends the VAT table built inside the Service to the Fiscal Printer. The VAT table is built one entry at a time using the **setVatValue** method.

This method is only supported if **CapHasVatTable** and **CapSetVatTable** are true. This method can only be called while **DayOpened** is false.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	One of the following errors occurred: • The Fiscal Printer does not support VAT tables or their setting (see the **CapHasVatTable** or **CapSetVatTable** property), or • The Fiscal Printer has already begun the fiscal day (see the **DayOpened** property).

See Also **CapHasVatTable** Property, **CapSetVatTable** Property, **DayOpened** Property, **setVatValue** Method.

setVatValue Method *Updated in Release 1.11*

Syntax **setVatValue (vatID: *int32*, vatValue: *string*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
vatID	Index of the VAT table entry to set.
vatValue	Tax value as a percentage.

Remarks Sets the value of a specific VAT class in the VAT table. The VAT table is built one entry at a time in the Service using this method. The entire table is then sent to the Fiscal Printer at one time using the **setVatTable** method.

This method is only supported if **CapHasVatTable** and **CapSetVatTable** are true. This method can only be called while **DayOpened** is false.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	One of the following errors occurred: • The Fiscal Printer does not support VAT tables (see the **CapHasVatTable** or **CapSetVatTable** property), or • The Fiscal Printer has already begun the fiscal day (see the **DayOpened** property), or • The Fiscal Printer does not support changing an existing VAT value (see the **CapSetVatTable** property).

See Also **CapHasVatTable** Property, **CapSetVatTable** Property, **DayOpened** Property, **setVatTable** Method**.**

verifyItem Method *Updated in Release 1.13*

Syntax **verifyItem (itemName: *string*, vatID: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
itemName	Item to be verified.
vatID	VAT identifier of the item.

Remarks Compares *itemName* and its *vatID* with the values stored in the Fiscal Printer.

This method is only supported if **CapHasVatTable** and **CapItemList** are true. This method can only be called while the Fiscal Printer is in the Item List state.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page Intro-20.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The Fiscal Printer does not support an item list report (see the **CapItemList** property) or the Fiscal Printer does not support VAT tables (see the **CapHasVatTable** property).
E_EXTENDED	*ErrorCodeExtended* = EFPTR_WRONG_STATE: The Fiscal Printer is not currently in the Item List state. *ErrorCodeExtended* = EFPTR_BAD_ITEM_DESCRIPTION: The item name is too long or contains a reserved word. *ErrorCodeExtended* = EFPTR_BAD_VAT: The VAT parameter is invalid.

See Also **CapHasVatTable** Property, **CapItemList** Property.

Events (UML interfaces)

DirectIOEvent

<< event >> **upos::events::DirectIOEvent**
EventNumber: *int32* { read-only }
Data: *int32* { read-write }
Obj: *object* { read-write }

Description Provides Service information directly to the application. This event provides a means for a vendor-specific Fiscal Printer Service to provide events to the application that are not otherwise supported by the Control.

Attributes This event contains the following attributes:

Attributes	Type	Description
EventNumber	*int32*	Event number whose specific values are assigned by the Service.
Data	*int32*	Additional numeric data. Specific values vary by the *EventNumber* and the Service. This property is settable.
Obj	*object*	Additional data whose usage varies by the E*ventNumber* and Service. This property is settable.

Remarks This event is to be used only for those types of vendor specific functions that are not otherwise described. Use of this event may restrict the application program from being used with other vendor's Fiscal Printer devices which may not have any knowledge of the Service's need for this event.

See Also **"Events"** on page Intro-19, **directIO** Method.

ErrorEvent *Updated in Release 1.13*

<< event >> **upos::events::ErrorEvent**
ErrorCode: *int32* { read-only }
ErrorCodeExtended: *int32* { read-only }
ErrorLocus: *int32* { read-only }
ErrorResponse: *int32* { read-write }

Description Notifies the application that a Fiscal Printer error has been detected and that a suitable response by the application is necessary to process the error condition.

Attributes This event contains the following attributes:

Attributes	Type	Description
ErrorCode	*int32*	Error code causing the error event. See a list of Error Codes on page 0-20.
ErrorCodeExtended	*int32*	Extended Error code causing the error event. If *ErrorCode* is E_EXTENDED, then see values below. Otherwise, it may contain a Service-specific value.
ErrorLocus	*int32*	Location of the error, and is set to EL_OUTPUT indicating that the error occurred while processing asynchronous output.

ErrorResponse *int32* Error response, whose default value may be overridden by the application (i.e., this property is settable). See values below.

If *ErrorCode* is E_EXTENDED, then *ErrorCodeExtended* has one of the following values:

Value	Meaning
EFPTR_COVER_OPEN	The Fiscal Printer cover is open.
EFPTR_JRN_EMPTY	The journal station is out of paper.
EFPTR_REC_EMPTY	The receipt station is out of paper.
EFPTR_SLP_EMPTY	A form is not inserted in the slip station.
EFPTR_SLP_FORM	A form is still present in the slip station even though it should have been removed by the last action.
EFPTR_WRONG_STATE	The requested method could not be executed in the Fiscal Printer's current state.
EFPTR_TECHNICAL_ASSISTANCE	The Fiscal Printer has encountered a severe error condition. Calling for Fiscal Printer technical assistance is required.
EFPTR_CLOCK_ERROR	The Fiscal Printer's internal clock has failed.
EFPTR_FISCAL_MEMORY_FULL	The Fiscal Printer's fiscal memory has been exhausted.
EFPTR_FISCAL_MEMORY_DISCONNECTED	The Fiscal Printer's fiscal memory has been disconnected.
EFPTR_FISCAL_TOTALS_ERROR	The Grand Total in working memory does not match the one in the EPROM.
EFPTR_BAD_ITEM_QUANTITY	The Quantity parameter is invalid.
EFPTR_BAD_ITEM_AMOUNT	The Amount parameter is invalid.
EFPTR_BAD_ITEM_DESCRIPTION	The Description parameters is either to long, contains illegal characters or contains the reserved word.
EFPTR_RECEIPT_TOTAL_OVERFLOW	The receipt total has overflowed.
EFPTR_BAD_VAT	The Vat parameter is invalid.
EFPTR_BAD_PRICE	The Price parameter is invalid.
EFPTR_BAD_DATE	The date parameter is invalid.
EFPTR_WORD_NOT_ALLOWED	The description contains a reserved word.

EFPTR_NEGATIVE_TOTAL	The Fiscal Printer's computed total or subtotal is less than zero.
EFPTR_MISSING_DEVICES	Some of the other devices which according to the local fiscal legislation are to be connected has been disconnected. In some countries in order to use a fiscal Fiscal Printer a full set of peripheral devices are to be connected to the POS (such as cash drawer and customer display). In case one of these devices is not present sales are not allowed.
EFPTR_BAD_LENGTH	The length of the string to be printed as post or pre line is too long.
EFPTR_MISSING_SET_CURRENCY	The Fiscal Printer is expecting the activation of a new currency.
EFPTR_DAY_END_REQUIRED	The completion of the fiscal day is required by calling **printZReport**. No further fiscal receipts or documents can be started before this is done.

The contents of the *ErrorResponse* property are preset to a default value, based on the *ErrorLocus*. The application's error processing may change *ErrorResponse* to one of the following values:

Value	Meaning
ER_CLEAR	Clear all buffered output data, including all asynchronous output. The error state is exited.
ER_RETRY	Retry the asynchronous output. The error state is exited. The default.

Remarks Enqueued when an error is detected and the Service's **State** transitions into the error state.

See Also **"Device Output Models"** on page Intro-25, **"Device Information Reporting Model"** on page Intro-30.

OutputCompleteEvent

<< event >> upos::events::OutputCompleteEvent
OutputID: *int32* { read-only }

Description Notifies the application that the queued output request associated with the *OutputID* attribute has completed successfully.

Attributes This event contains the following attribute:

Attributes	Type	Description
OutputID	*int32*	The ID number of the asynchronous output request that is complete.

Remarks This event is enqueued after the request's data has been both sent and the Service

has confirmation that is was processed by the device successfully.

See Also **"Device Output Models"** on page Intro-25.

StatusUpdateEvent *Updated in Release 1.8*

<< event >> upos::events::StatusUpdateEvent
Status: *int32* { read-only }

Description Notifies the application that a Fiscal Printer has had an operation status change.

Attributes This event contains the following attribute:

Attributes	Type	Description
Status	*int32*	Indicates the status change, and has one of the following values:

Value	Meaning
FPTR_SUE_COVER_OPEN	Fiscal Printer cover is open.
FPTR_SUE_COVER_OK	Fiscal Printer cover is closed.
FPTR_SUE_JRN_EMPTY	No journal paper.
FPTR_SUE_JRN_NEAREMPTY	Journal paper is low.
FPTR_SUE_JRN_PAPEROK	Journal paper is ready.
FPTR_SUE_REC_EMPTY	No receipt paper.
FPTR_SUE_REC_NEAREMPTY	Receipt paper is low.
FPTR_SUE_REC_PAPEROK	Receipt paper is ready.
FPTR_SUE_SLP_EMPTY	No slip form is inserted, and no slip form has been detected at the entrance to the slip station. (See "Model" on page 15-14 for further details on slip properties and events.)
FPTR_SUE_SLP_NEAREMPTY	Almost at the bottom of the slip form.
FPTR_SUE_SLP_PAPEROK	Slip form is inserted.
FPTR_SUE_IDLE	All asynchronous output has finished, either successfully or because output has been cleared. The Fiscal Printer **State** is now S_IDLE. The **FlagWhenIdle** property must be true for this event to be delivered, and the property is automatically reset to false just before the event is delivered.

Note that Release 1.3 added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values*.

The Update Firmware capability, added in ***Release 1.9***, added additional *Status* values for communicating the status/progress of an asynchronous update firmware process.

See "**StatusUpdateEvent**" description on page 1-34.

Release 1.8 and later – Specific Cover State Reporting

Starting with Release 1.8, **StatusUpdateEvent**s for specific stations' covers are supported. If a Fiscal Printer has only one cover or if it cannot determine/report which covers are open, then only the original FPTR_SUE_COVER_OPEN and FPTR_SUE_COVER_OK events should be fired.

For Fiscal Printers supporting multiple covers, the original events should also be fired for compatibility with current applications. In these cases, the station-specific event should be fired **first**, followed by the original event.

If more than one cover is open, the original FPTR_SUE_COVER_OPEN event should only be fired once after a cover is opened. A FPTR_SUE_COVER_OK event should only be fired after all the covers are closed.

The event's *Status* attribute can contain one of the following additional values to indicate a status change.

Value	Meaning
FPTR_SUE_JRN_COVER_OPEN	Journal station cover is open.
FPTR_SUE_JRN_COVER_OK	Journal station cover is closed.
FPTR_SUE_REC_COVER_OPEN	Receipt station cover is open.
FPTR_SUE_REC_COVER_OK	Receipt station cover is closed.
FPTR_SUE_SLP_COVER_OPEN	Slip station cover is open.
FPTR_SUE_SLP_COVER_OK	Slip station cover is closed.

Remarks Enqueued when a significant status event has occurred.

See Also **"Events"** on page Intro-19.

CHAPTER 16

Gate

This Chapter defines the Gate device category.

Summary

Properties (UML attributes)

Common	*Type*	*Mutability*	*Version*	*May Use After*
AutoDisable:	*boolean*	{ read-write }	1.12	Not Supported
CapCompareFirmwareVersion:	*boolean*	{ read-only }	1.12	open
CapPowerReporting:	*int32*	{ read-only }	1.12	open
CapStatisticsReporting:	*boolean*	{ read-only }	1.12	open
CapUpdateFirmware:	*boolean*	{ read-only }	1.12	open
CapUpdateStatistics:	*boolean*	{ read-only }	1.12	open
CheckHealthText:	*string*	{ read-only }	1.12	open
Claimed:	*boolean*	{ read-only }	1.12	open
DataCount:	*int32*	{ read-only }	1.12	Not Supported
DataEventEnabled:	*boolean*	{ read-write }	1.12	Not Supported
DeviceEnabled:	*boolean*	{ read-write }	1.12	open
FreezeEvents:	*boolean*	{ read-write }	1.12	open
OutputID:	*int32*	{ read-only }	1.12	Not Supported
PowerNotify:	*int32*	{ read-write }	1.12	open
PowerState:	*int32*	{ read-only }	1.12	open
State:	*int32*	{ read-only }	1.12	--
DeviceControlDescription:	*string*	{ read-only }	1.12	--
DeviceControlVersion:	*int32*	{ read-only }	1.12	--
DeviceServiceDescription:	*string*	{ read-only }	1.12	open
DeviceServiceVersion:	*int32*	{ read-only }	1.12	open
PhysicalDeviceDescription:	*string*	{ read-only }	1.12	open
PhysicalDeviceName:	*string*	{ read-only }	1.12	open