`

2017

Fw21 API

SPECIFICATION OF THE SOFTWARE INTERFACE OF THE FW21 MODULE FOR ACCESS TO THE FUNCTIONS: ECR POSPRINT FP4 (5) 10-F, SKY-PRINT

WITH 14.X VERSION FIRMWARE

Table of Contents

Introduction

Fw21 software module is designed to integrate POSPrint FP4(5)10-Ф, SKY-PRINT Electronic Cash Register Equipment (ECR) (firmware 14.x) with POS SW in Windows environment. The document describes Fw21 module software specification. Intended readers:

- POS SW developers;
- ECR maintenance SW developers.

General info on ECR

POSPrint FP4(5)10-Ф, SKY-PRINT cash registers (firmware 14.x) is successor to FP4(5)10-К, SKY-PRINT models (firmware 12.x). Models with firmware 14.x are designed in accordance with new regulatory requirements (Federal Law 54-FZ):

- Support of Fiscal Storage Device version ФН-М in Format of Fiscal Documents 1.2 mode.
- Support of Fiscal Document Format version 1.2.
- Support of registration of accounting objects that are subject to tagging by identification facilities in accordance with the Russian Federation Government Decree on mandatory tagging of specific groups of goods (hereinafter — goods “subject to tagging” or “taggable” goods).

ECR of specified models:

- Generates fiscal documents in accordance with commands issued by control computer, and saves them in Fiscal Storage Device (FS).
- Prints generated documents on paper tape attending them with fiscal attribute generated by FS.
- Transmits fiscal documents from FS to Fiscal Data Operator without control computer action, provided that ECR is connected to network infrastructure via a device (switch):
    - having possibility to connect external devices via USB;
    - having possibility to recognize USB devices with Ethernet protocol (RNDIS technology) implemented;
    - providing TCP/IP packets switching.

- Interacts with Marking Information Systems Operator to verify codes of accounting objects being tagged, and to send notifications on sales and returns thereof.

Specification of ECR interaction interface is described in “Electronic Cash Register Programming Guide” document (FW21_spec.docx). “FW21_spec.docx” document defines:

- Set of commands for ECR control.
- Data types used by commands for ECR control.
- Format of “packet” with data being transmitted to ECR (parameters), and data being returned by ECR (results).

Fiscal Document Format

Regulatory act Attachment No. 2 to the Order of Federal Tax Service of the Russian Federation dated 14.09.2020 No. ЕД-7-20/662@ “MANDATORY FISCAL DOCUMENT FORMATS” specifies the following fiscal document formats:

1. FFD 1.0 — discontinued on 31.12.2018
2. FFD 1.05 — effective
3. FFD 1.1 — effective
4. FFD 1.2 — replaces all previous versions.

ECR with 14.x firmware and fw21.dll module only support FFD 1.2. For the transition period driven by the need for gradual transition from FFD 1.05 to FFD 1.2 with big number of ECR within the same chain store, parallel support of both fw21.dll and fw16.dll modules by the same cash register equipment SW is recommended. Cash register SW shall have settings providing “connection” of either one or another module. For gradual ECR fleet update from 12.x to 14.x firmware version.

fw16 and fw21 API specifications consistency

In general, API specification implemented in fw21 API duplicates fw16 specification but still, they are not fully compatible. Specification differences are listed below:

1. ‘Fw16’ root namespace is replaced by ‘Fw21’.
2. ‘Fs.Native’ root namespace is replaced by ‘Fw21.Fs’.
3. Discount and surcharge assignment methods for accounting objects (were only valid for FFD 1.0) are deleted.
4. Specific methods for fiscal “correction sales receipt” generation are deleted. Starting from FFD 1.1, correction documents are created as “conventional receipts” with transmission of additional correction credentials.
5. Methods and properties related to 1162 property setting (goods code) are deleted. 1162 property is removed from FFD 1.2.
6. Goods quantity unit type is changed (arbitrary string is replaced by list of possible values).
7. Methods for taggable goods tag codes preliminary verification are added.
8. Credentials added: industry, operational, fractional.

API provides information property (`EcrCtrl.Info.FfdVersion`) reflecting Fiscal Document Format version supported by ECR connected. The named property for 14.x version only takes the value of 4 (FFD 1.2). Fw21 can interact with ECR with 12.x firmware, but, if `EcrCtrl.Info.FfdVersion` values other than 4 are detected, interaction with ECR is limited to data collection commands only.

Fw21 software module general information

Fw21 software module (hereinafter, Fw21 module) is designed to integrate ECR to goods distribution accounting systems in Windows environment (hereinafter, “Accounting System”), and provides:

1. Windows application data type conversion to ECR exchange data types, and backwards.
2. Representation of “linear” set of ECR commands as hierarchical set of objects and control methods thereof.

Fw21 software module DOES NOT PERFORM control of data being transmitted to ECR, or modification of any values being returned. Data control necessary to generate fiscal documents, is performed by ECR. If conditions for successful Fiscal Document generation are specified in this document within methods description, this shall be understood as *“conditions of successful Fiscal Document creation are controlled by ECR”*.

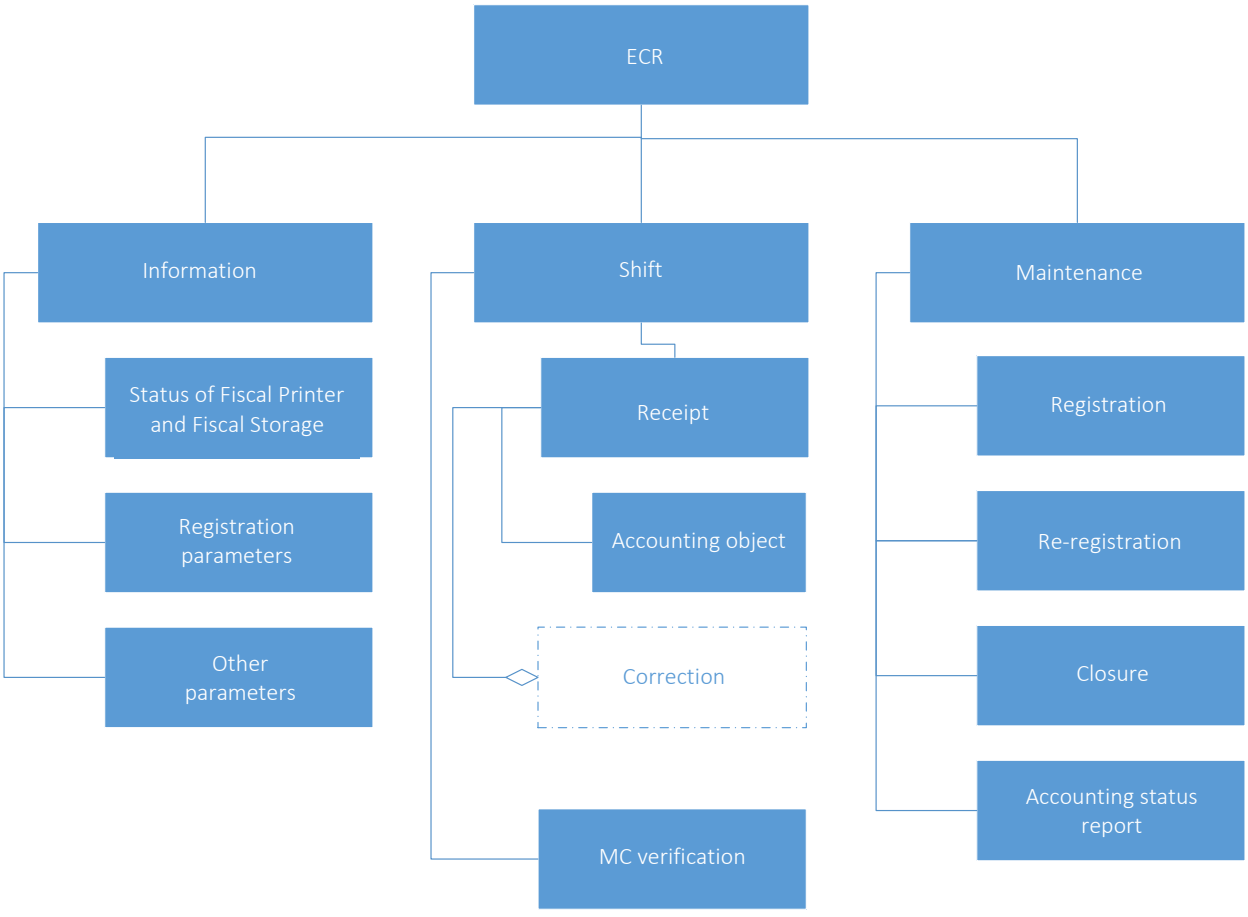
If data does not match ECR requirements, ECR returns an error reflected by Fw21 module onto the accounting system being integrated.

Purpose of ECR integration to the accounting system by means of Fw21 module is to simplify the following:

- Data collection from ECR;
- Generation of Fiscal Documents.

Fw21 software module operates in .Net Framework 2.0 environment. It is accessible to applications executed in Windows 2000 Pro SP4 environment and above, including embedded versions of operation system. Fw21 module is accessible to applications via ‘managed’ integration interfaces.

Functional diagram of set of objects provided by Fw21, is shown in the following figure:



Getting started

Fw21 module is implemented as dynamic link library (DLL). Depending on selected integration interface type, Fw21 module is activated by creation of managed object of Fw21.EcrCtrl type.

Fw21 module as settings that determine the way of ECR connection to control computer. Module settings are incorporated into settings of application using it (*.exe.config) as separate section.

Example of section with Fw21 module settings is shown in Attachment 1.

Upon successful connection to ECR via Fw21 module, the software using the module can get ECR status, and evaluate ECR readiness to perform specific functions.

Below are the diagram of verification of ECR readiness to perform “daily” functions of cash register SW, and actions to make ECR ready if needed. Actions specified in the diagram not only include the actions of cash register (cashier) program, but also possible service desk actions. Service desk actions are shown in the diagram in order for cash register SW to be able to display corresponding recommendations to cashier as messages.

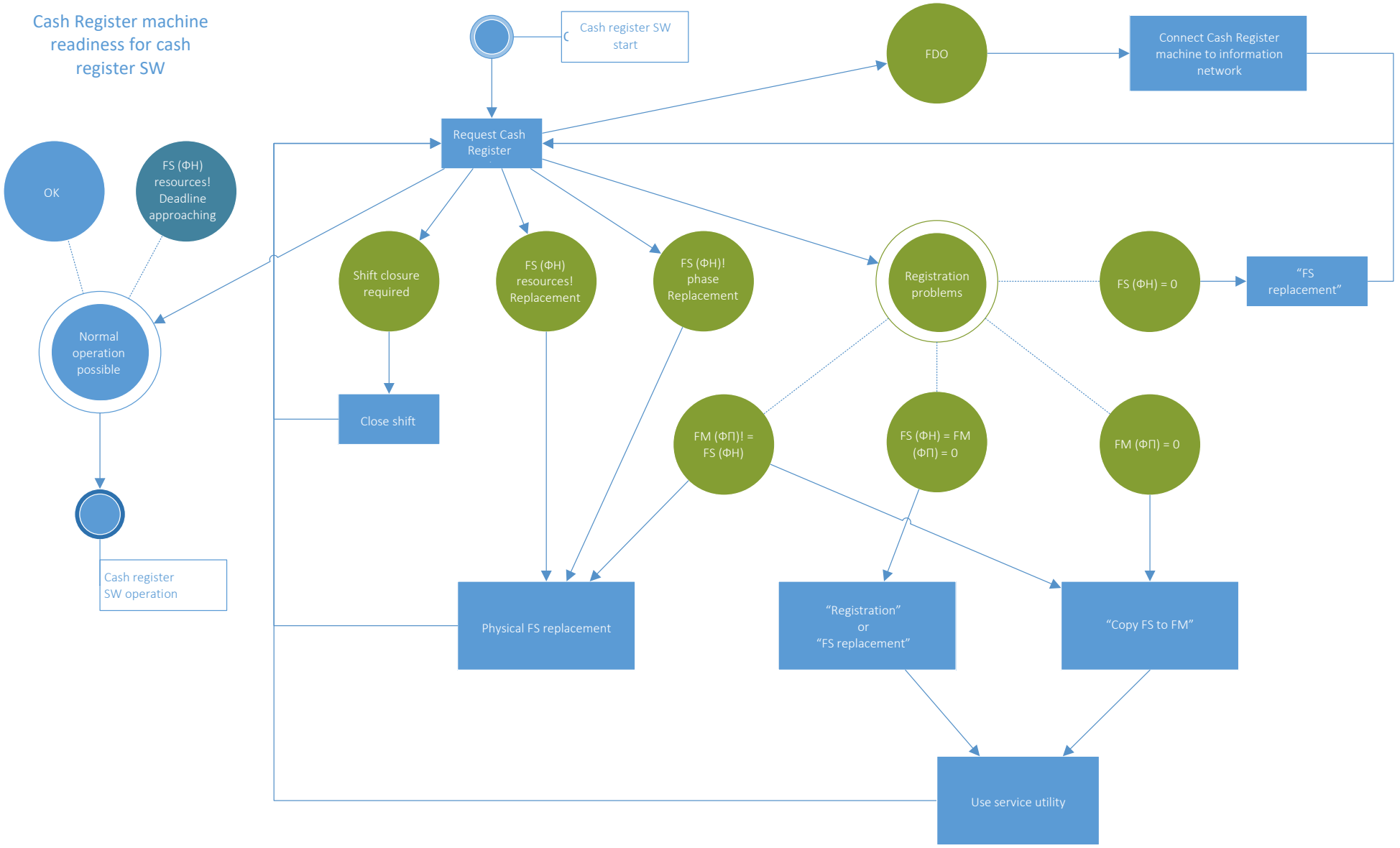
The diagram only shows the statuses resulting from natural reasons or service desk erroneous actions in the course of ECR setup after technical issues are resolved. The diagram does not show the cases of ECR malfunctions, including Fiscal Memory Device malfunctions (printer error, FS not connected, ECR clock not operational…)

The following acronyms are used in the diagram:

- FS (ФН) — Fiscal Storsage Device.
- FM (ФП) — ECR ROM.
- FS (ФН) = 0 — FS is in “Fiscalization readiness” phase
- FM (ФП) = 0 — ECR memory does not contain registration data.
- FS (ФН) != FM (ФП) — FS and FM registration data do not match.
- FS (ФН) resources! — validity period, archive volume, crypto co-processor limit.
- FDO (ОФД)! — data volume that was not transmitted from ECR to Fiscal Data Operator
- FS (ФН)! phase — FS is in one of post-fiscalization phases.

The diagram shows names of some actions in quotes. The meaning thereof is explained here:

- **”Registration”** — operation of “ECR registration report” fiscal document generation. Performed for ECR registration number newly received from the Federal Tax Agency.
- **”FS replacement”** — operation of “ECR parameters change due to FS replacement” fiscal document generation. Performed at ECR for which “Registration report” has been generated earlier but new FS is installed thereto.
- **”Copy FS to FM”** — operation of registration data transfer from FS to ECR memory if ECR memory was cleared in the course of repair. The operation does not generate any fiscal documents.


Cash Register machine readiness for cash register SW
Cash register SW start
FDO
Connect Cash Register machine to information network
Request Cash Register
OK
FS (ΦН) resources! Deadline approaching
Shift closure required
FS (ΦН) resources! Replacement
FS (ΦН)! phase Replacement
Registration problems
FS (ΦН) = 0
"FS replacement"
Normal operation possible
Close shift
FM (ΦП)! = FS (ΦН)
FS (ΦН) = FM (ΦП) = 0
FM (ΦП) = 0
Cash register SW operation
Physical FS replacement
"Registration" or "FS replacement"
"Copy FS to FM"
Use service utility

Correspondence of ECR and Fw21 module data types

This section describes correspondence of data types used in the course of data exchange between ECR and application being integrated. Data types used in the course of ECR data exchange, are described in “ECR programming guide”.

Note: *the phrase “ECR data types” in this document shall always be understood as “data types used in the course of data exchange with ECR”. Internal representation of ECR data is beyond the scope of this document.*

Data types:

ECR	Managed
BYTE(n)	byte[]
BYTE(1)	byte,
BYTE(1)	bool
BYTE(1)	enum
UINT16	ushort
UINT32	uint
UINT64	ulong
CHAR(n)	string
PCHAR (n)	string
BDATA(n)	byte[]
BDATA(n)	object
DATETIME	DateTime
DECIMAL	decimal

Depending on usage context, ECR data type can be represented by different application types, and vice versa.

All string type data is represented in Unicode. In the course of exchange with ECR, Fw21 module re-encodes data from Unicode into CP866, and backwards.

DateTime type always contains “time”. If this parameter usage context only involves date, “time” value is not used:

- When data is transmitted to ECR, “time” value is ignored;
- When data is transmitted from ECR, “time” value is set to 00:00:00

DateTime type values corresponding to Fiscal Document credentials and having “time”, round this value to minutes (seconds are discarded).

Marking Code verification

In accordance with the Russian Federation legal requirements, specific goods are subject to marking by identification facilities that guarantee legitimacy of the goods’ origin and turnover. Identification facilities contain unique goods item code that can be verified in the State Information System of Marking Goods (GIS MT = MISO). This system support is provided by Tagging Information System Operator (MISO) assigned by the Government of the Russian Federation.

Unique goods item code (hereinafter MC) can be represented in printed form (Datamatrix code), or in electronic form (RFID, NFC tags). Code read by corresponding technical facilities can be verified at any phase of goods turnover at the territory of the Russian Federation: production (import), transportation, sales (including return). MC verification at retail sales phase (return) is performed by means of ECR. To do so, ECR shall have appropriate FS model (ФН-М) and support FFD 1.2. Beyond that:

1. To check MC, ECR shall be registered with “marking” application attribute.
2. Verification can be performed within open shift not exceeding 24 hours.

MC verification can be performed both just prior to adding accounting object to “sales receipt (corrections)”, and in advance prior to “sales receipt (corrections)” fiscal document generation. The verification result can be:

- Positive.
- Negative.
- Indefinite (verification is technically not possible at the moment of performance thereof)

Verification results are stored in FS, and allow to add accounting objects with MC to Fiscal Document. It is not allowed to add accounting objects with TCs that are not verified, to Fiscal Document. In case of negative or indefinite verification result, such result can only be saved in FS if there is a consent. In case of sale, buyer’s consent is required. In case of return, recipient’s (store’s) consent is required.

When fiscal document is being finished, the saved verification results are deleted. Verification results shall also be deleted:

- By special command.
- In case of ECR power supply cutoff (failure).

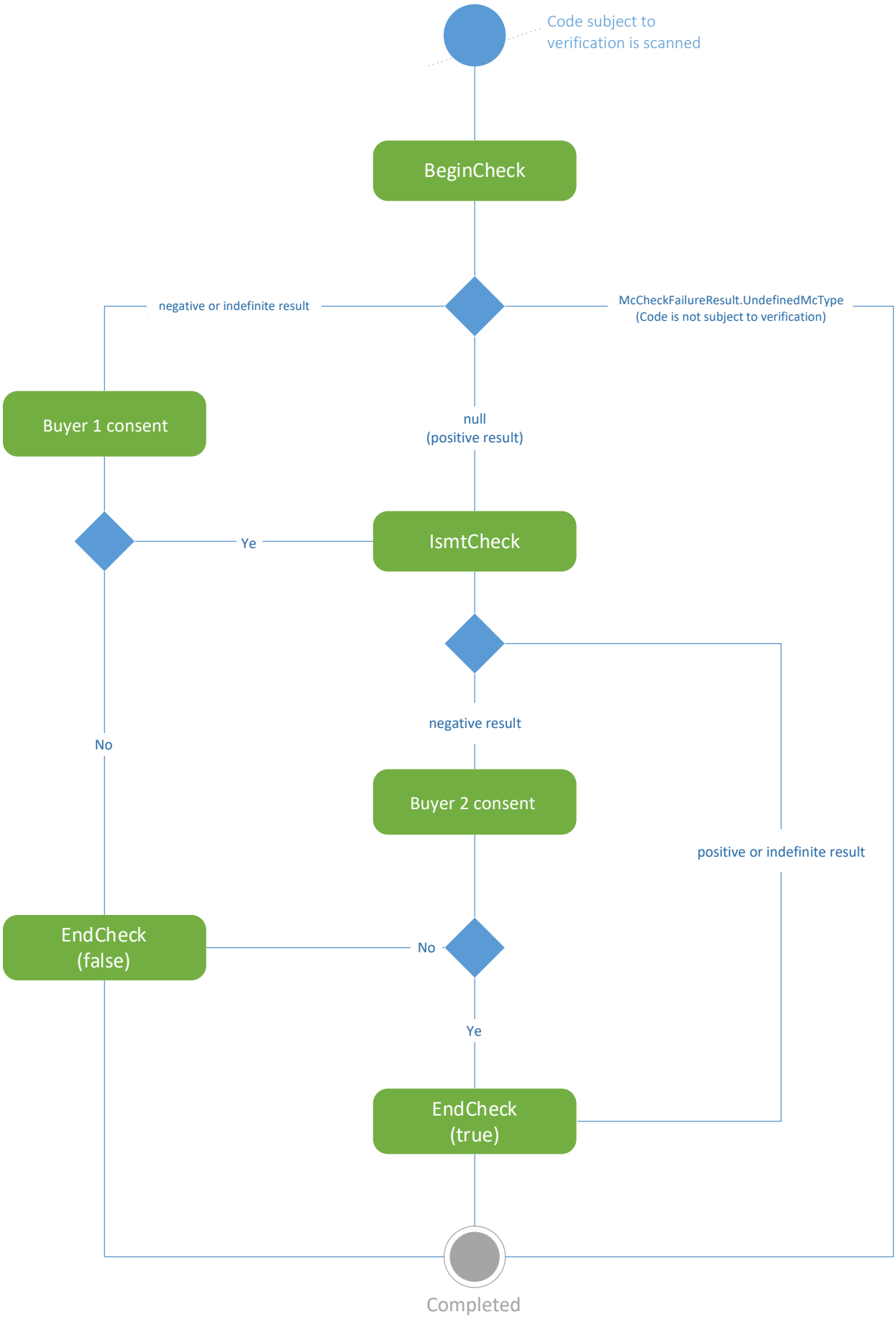
FS has limited resource for verified MCs storage (128). If exceeded, MC verification will be impossible up to Fiscal Document closure.

To verify MC, use Ecr.Shift.McCheck object. The object implements methods that can perform verification in two ways:

1. Call of one CheckKm method with callback parameter to notify the calling party on necessity to get consent to add FS verification result if it is negative or indefinite.
2. Consecutive (consistent) call of BeginCheck, IsmtCheck, EndCheck methods between which, if necessary, consents are received to add FS verification result if it is negative or indefinite.

Sequence of calls is assumed for the second way, as shown in the figure:

For network ECR



Adding accounting objects with MC to “Receipt” is performed the same way as for accounting objects without MC (Receipt.NewItemCosted, Receipt.NewItemPriced).

Upon completion of Fiscal Document generation, notification is generated in FS on tagged goods sale, that is sent by ECR to MISO automatically in background mode if Internet connection is up. Same as ending FD to FDO. FS has limited resource for storage of unsent notifications. If exceeded, generation of FD with accounting objects having MC will be impossible.

To assess FS resource utilization in “tagging” context, use information on corresponding FS status represented by McCheck.State property.

“Fast” MC verification

MC verification for ECR in network mode must include “MC verification at MISO” step. This verification requires real-time interaction with resource external to ECR (MISO). This requirement creates potential for additional buyer’s delay “at cash register” if there are many goods requiring MISO verification in the “shopping cart”.

In accordance with FS specification, to avoid such delays, ignoring MISO response to previous MC request is possible if next MC needs to be verified already. In this case, “ignoring” means that “local” verification result is only stored in the FS from the previous MC.

It is assumed that cash register SW is able to scan MC in a thread separated from ECR interaction thread. By calling verification methods to verify the next MC in the thread separated from the thread verifying the previous MC, cash register SW has possibility to “interrupt” the previous MC verification (ignore “delayed” MISO response).

To support the possibilities to avoid delays (possibility to ignore MISO response). MISO verification methods are implemented considering the possibility to verify different TCs in different threads of calling application (cash register SW). Therewith, it is necessary to have all verification methods (BeginCheck, IsmtCheck, EndCheck) called in one MC verification context to be called from the same thread in case of “way 2”. This is how it happens in case of “way 1”.

API

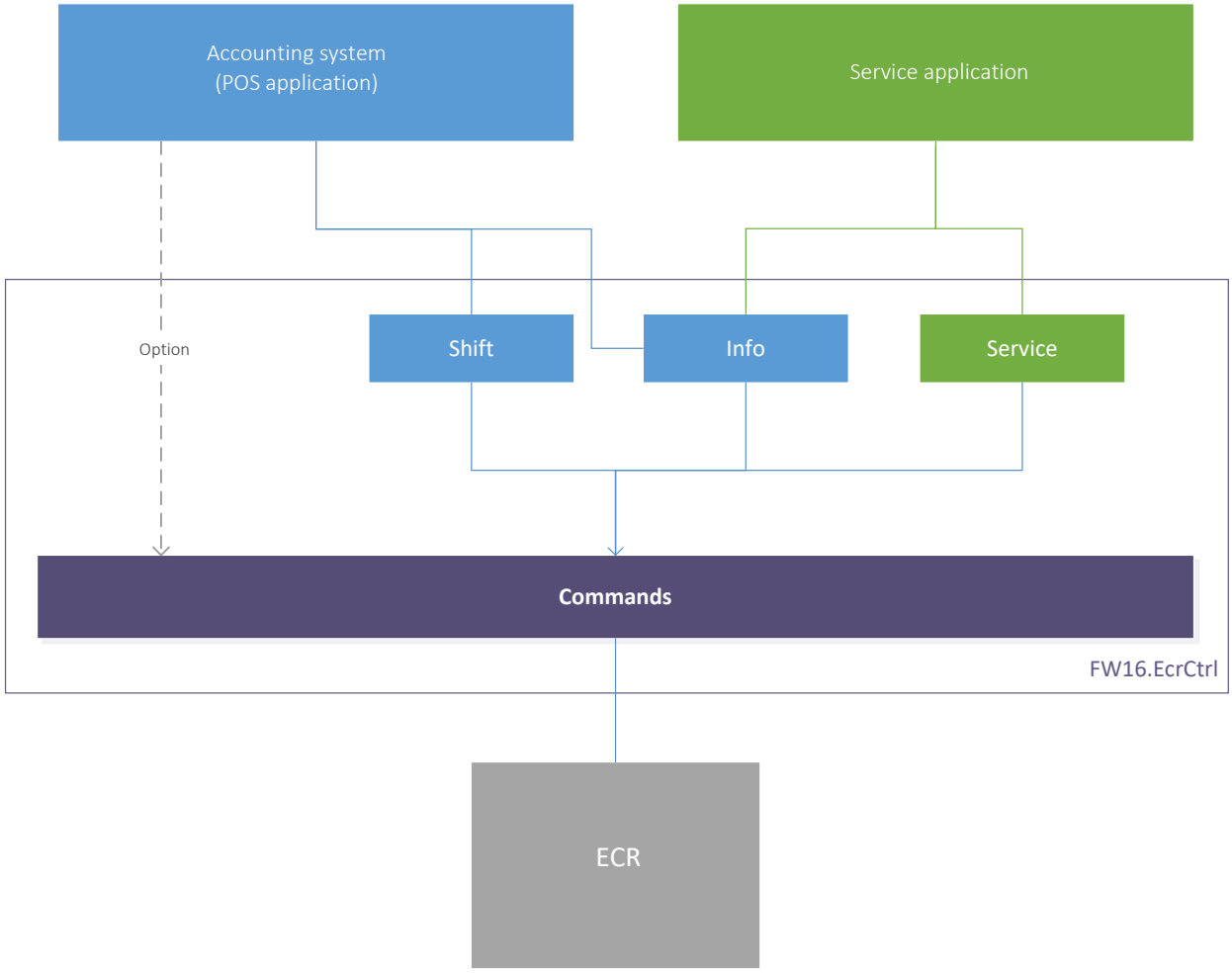
Integration is based on Fw21.EcrCtrl type usage.

Fw21.EcrCtrl class defines 3 properties, each being an object providing access to ECR function group:

- Info — “Info” group is for collecting information by ECR. Both on current status of ECR itself and on current state of data stored therein.
- Shift — “Shift” group is designed to create data structures which are then transformed into ECR command sequence resulting in generation of “daily” fiscal documents. Namely, shift opening and closure, receipts, correction receipts.
- Service — “Service” group is designed to generate fiscal documents providing “lifecycle”, and to perform other actions necessary to maintain ECR operational state, including setup thereof.

Each property listed above implements its own interface (own set of properties and methods).

Additionally, Fw21.EcrCtrl class object defines Commands property. Commands object methods reflect the ECR commands described in “ECR programming guide”. Info, Pos, Service objects use Commands to access ECR. If needed, Commands object can be used by external applications. This is illustrated by the following figure:



EcrCtrl(Fw21.EcrCtrl)

Properties

Name	Type	Description
Info	Ecr.Info	Miscellaneous info on ECR
Shift	Ecr.Shift	Shift documents control
Service	Ecr.Service	Document control beyond ECR shift in general
Commands	Native.CmdExecutor	Instance of access to ECR native commands
Fw16	Native	Instance of access to internal services

Methods

Init

Specification	**void Init (string connectionName)** **void Init (int serialPort, int baudRate = 57600)**		
Description	Connect to ECR with specified parameters		
Returns	No		
Parameters	Name	Type	Description
	connectionName	String	Name of connection instance specified in *.config (`../Instance/@name`), or connection parameters in URI format. Examples: serial://com:1 serial://com:4?baudrate=115200 udp://192.168.100.15:8888
	serialPort	int	Serial port number
	baudRate	int	Exchange rate via serial port (only valid for rs-232, not USB)

Reconnect

Specification	**void IDisposable.`Dispose` ()**		
Description	Release resources occupied by ECR connection		
Returns	No		
Parameters	Name	Type	Description
No			

Reconnect

Specification	**void `Reconnect` ()**		
Description	Update data collected at ECR connection		
Returns	No		
Parameters	Name	Type	Description
no			

Info (Ecr.Info)

Returns the following info from ECR:

Properties

Name	Type	Description
EcrInfo	EcrModelInfo	ECR manufacturer info
Clock*)	DateTime	ECR clock data
Status*)	Ecr.GeneralStatus	Generalized status of ECR readiness to use cash register SW
EcrStatus*)	Ecr.Status	ECR detailed status
FsStatus*)	Fs.Native.StatusData	FS detailed status
OfdStatus*)	Fs.Native.OfdStatus	FS and FDO data exchange detailed status
RegInfo	Ecr.RegInfo	Up-to-date registration info
FsId	string	Fiscal Memory Device number
FsVersion	Fs.Native.VersionData	Version of Fiscal Memory Device installed in ECR

**) Any property value collection always results in device connection session. That is, the value returned is entirely up-to-date, but, in case of extensive usage, may cause significant SW performance degradation.*

Ecr.EcrModelInfo

ECR data set at manufacture time.

Properties

Name	Type	Description
Id	`string`	ECR serial number
Model	`string`	ECR model
Version	`ushort`	ECR SW version number

Ecr.GeneralStatus

Status is returned as 32-bit value. Any bit set to 1 means a problem restricting fiscal document generation. The program can perform actions to mitigate the problems that can be resolved by software. The problem reason can be ascertained by states and properties of specific ECR components (EcrStatus, FsStatus, OfdStatus, RegInfo)

Values

Name	Value	Description
Inoperable	`0x0001`	ECR is inoperable
NotConfigured	`0x0002`	ECR is not configured
NoReg	`0x0004`	ECR registration required

FsReadyToReg	0x0008	FS registration required
FsExhausted	0x0010	FS replacement required
ShiftOver	0x0020	Shift closure required
OfdRequired	0x0040	ECR connection to network required to transmit data to FDO
ShiftOpened	0x0080	Shift opened
DocOpened	0x0100	Document opened
CoverOpebed	0x0200	Printer cover opened
PaperOut	0x0400	Receipt tape out
HasPending	0x0800	There are FDs pending acknowledgement from FDO

Ecr.Status

Status is returned as 32-bit value. Each bit is responsible for one value of ECR status ECR status is defined by combination of specified values (bits).

Values

Name	Value	Description
IdSet	00000001h	ECR serial number is set
ClockSet	00000002h	ECR clock is set
ClicheSet	00000004h	Cliche is set
Registred	00000008h	ECR is registered
FsClosed	00000010h	FS fiscal mode is closed
ShiftOpened	00000020h	Shift opened
DocOpened	00000040h	Fiscal document opened
NDocOpened	00000080h	Non-fiscal document opened
ShiftOver	00000100h	Open shift limit exceeded (> 24 hrs)
NoFs	00000200h	FS not connected
FsError	00000400h	FS error
FsBlocked	00000800h	FS blocked
HasOfdMessage	00001000h	There is a message for ECR
AlertFs3DaysLeft	00002000h	Urgent replacement of FS cryptographic coprocessor (CC) (3 days left)
AlertFs1MonthLeft	00004000h	FS CC resource exhausted (30 days left)
AlertFsNearFull	00008000h	FS archive is 90 % full
AlertOfdAckTimeout	00010000h	FDO acknowledgement timeout expired
HWError	00020000h	ECR hardware error
FWError	00040000h	ECR logical error
	00080000h	-
	00100000h	-
	00200000h	-
	00400000h	-
	00800000h	-
NoPrinter	01000000h	Printer not connected
NoPrinterRespond	02000000h	Printer not responding
CoverOpened	04000000h	Printer cover opened
PaperOut	08000000h	Receipt tape out
PaperNearEnd	10000000h	Receipt tape almost out
DrawerOpened	20000000h	Cash drawer opened
	40000000h	-
	80000000h	-

Fs.Native.StatusData

Class describing Fiscal Memory Device (FS, ФН) status

Name	Type	Description
Phase	LifecyclePhaseType	Lifecycle phase status Manufactor – Setting ReadyToReg – Readiness for fiscalization. FS configured. Registered – fiscal mode open. (FD adding possible) PostRegistred – PostRegistered (post-fiscal) mode, FD is being transmitted to FDO (upon fiscal mode closure, FDs are left that have not been transmitted to FDO) Readonly – It is only possible to read data from FS Archive. Fiscal data transmission to FDO is completed
CurrentDocType	DocumentType	Type of started document if started
CurrentDocReceived	bool	Document data False — no document data; True — document data received
ShiftOpened	bool	Shift status False — shift closed; True — shift opened
Alerts	AlertType	Warning flags None – No warnings TreeDayToTheEnd – Urgent FS replacement (3 days to validity period expiration) OneMonthToTheEnd – CCresource exhausted (30 days to validity period expiration) ArchiveNearFull – FS memory overflow (FS Archive is 90 % full) AcknowledgeTimeout – FDO response waiting time exceeded CriticalError – Critical FS error
LastDocTime	DateTime	Date and time of the last document
FsId	string	FS number as ASCII string
LastDocNum	uint	Last FD number

Fs.Native.OfdStatus

Class describing the status of Fiscal Memory Device (FS, ФН) in relation to fiscal documents transmission to Fiscal Data Operator (FDO, ОФД). This information is only to be displayed by calling program. It is not assumed that calling program will use this information to modify its own state.

Name	Type	Description
ExchageStatus	ExchangeStatusType	Data exchange status
MessageReadStatus	MessageReadStatusType	Message for FDO read status
MessageCount	ushort	Number of messages to be transmitted to FDO 0 — if no messages subject to transmission to FDO
DocumentNumber	uint	Document number of the first-in-queue document for FDO. If the document has already been transmitted, this would be document number of the document awaiting acknowledgement. 0 if no documents in queue.
DocumentDT	DateTime	Date of the first-in-queue document for FDO

Ecr.RegInfo

Name	Type	Description
DT	DateTime	Registration date
RegId	String	ECR registration number
OwnerTaxId	String	ECR owner's Taxpayer ID (INN)
OwnerName	String	ECR owner's name
SenderAddress	String	Receipt (ФФД2) sender’s e-mail address
OfdTaxId	String	FDO’s Taxpayer ID (INN)
OfdName	String	FDO’s name
Location	String	Settlement location
Address	String	Settlement address
Taxations	FS.Native.TaxationKind	Acceptable taxation types for fiscal documents generated by ECR
Mode	Ecr.RegOptions	Registration parameters defining ECR usage area (mode)

Ecr RegOptions

Name	Type	Description
Standalone	bool	ECR standalone mode (true) or data transmission to FDO (false)
Form		Cash sale receipt mode (false) or accountable Form — AF, БСО (true)
Encrypted		FD encryption mode (true — enabled)
Internet		Internet settlement attribute. Do not print FD, printer may be absent (true — enabled)
Excise		Excise goods sales attribute (true — enabled)
Service		Service payment attribute (true — enabled)
Gambling		Lottery payment attribute (true — enabled)
Lottery		Lottery payment attribute (true — enabled)

Agent	AgentType (enum)	Settlement agent attribute
		`/// not a payment agent`
		`Unspecified = 0,`
		`/// Bank payment agent`
		`Bank = 0x01,`
		`/// Bank payment subagent`
		`BankSub = 0x02,`
		`/// Payment agent`
		`Payment = 0x04,`
		`/// Payment subagent`
		`PaymentSub = 0x08,`
		`/// Attorney`
		`Attorney = 0x10,`
		`/// Commissioner`
		`Comissioner = 0x20,`
		`/// Agent, none of the above`
		`Other = 0x40,`

SCO	Ecr.SCOption	`/// SCO number`
		`string ScoId;`
		`/// Printer outside ECR enclosure but inside SCO`
		`bool RemotePrinter;`

Methods

GetCounter

Specification	`ushort GetCounter(ushort index)`		
Description	Receives the value of specified ECR counter		
Returns	Counter value		
Parameters	Name	Type	Description
	index	`ushort`	Counter index (see ECR description)

GetRegister

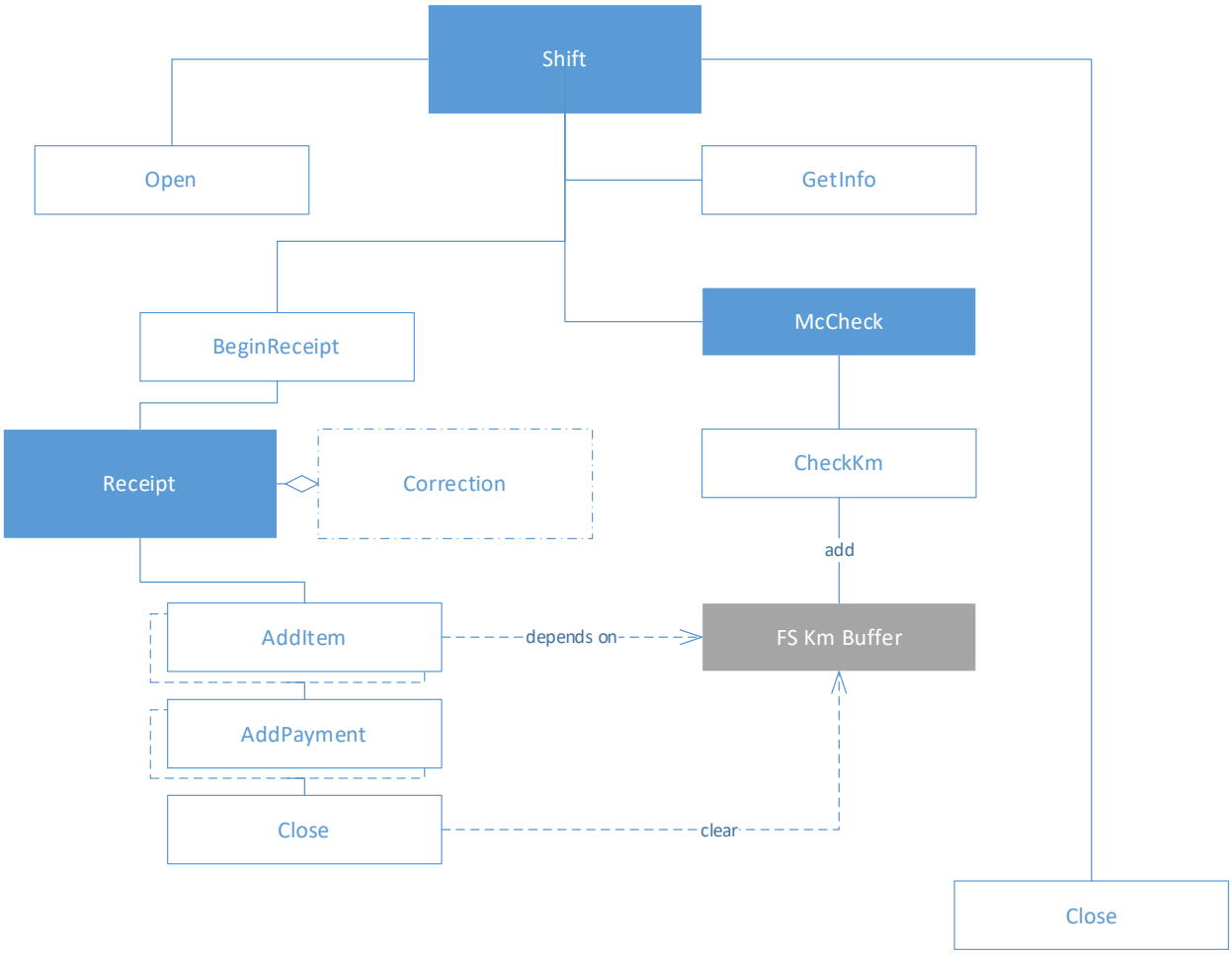
Specification	`decimal GetCounter(ushort index)`		
Description	Receives the value of specified ECR adding register		
Returns	Register / adder value.		
Parameters	Name	Type	Description
	index	`decimal`	Register / adder index (see ECR description)

GetRegister

Specification	`IEnumerable<Native.CmdExecutor.TenderInfo> GetTendersList()`		
Description	Receives list of payment facilities known to ECR		
Returns	List of payment facilities		
Parameters	Name	Type	Description
	No		

Shift(Ecr.Shift)

Ecr.Shift object provides generation of data structures which are then transformed into ECR command sequence resulting in generation of “daily” FS fiscal documents. The figure below shows basic objects of this functionality part of Fw21 module and essential methods thereof.



Fiscal documents are generated by successful call of the following methods:

- Shift.Open
- Receipt.Complete
- Correction.Complete
- Shift.Close

Properties

Ecr.Shift.Info

Name	Type	Description
Opened	bool	Attribute of open shift
Number	ushort	Shift number If the shift is closed, number of the last closed shift, if open — number of current shift.
DT	DateTime	Shift opening time
DocCounter	ushort	If the shift is closed, number of documents in the previous closed shift (0 if this is the first shift). If the shift is open but there are no receipts, then 0. In other cases, number of the last generated receipt

Methods

GetInfo

Specification	`Info` **GetInfo()**		
Description	Receives current shift credentials		
Returns	Info		
Parameters	Name	Type	Description
	no		

Open

Specification	`void Open(string operatorName)`
	`void Open(EcrOperator operat)`
	`public void Open(EcrOperator operat, bool print)`		

Description	Opens new ECR shift if current shift is closed		
Returns	no		
Parameters	Name	Type	Description
	userName	string	Cashier name
	`operat`	`EcrOperator`	Cashier credentials (Name, INN, position)
	`print`	`bool`	Hard copy printout attribute (false — do not print)

BeginReceipt

Specification	`Receipt BeginReceipt(`
	`  string userName,`
	`  Model.ReceiptKind operation,`
	`  object optionalRequisites)`
	`Receipt BeginReceipt(`
	`  EcrOperator operat,`
	`  Model.ReceiptKind operation,`
	`  object optionalRequisites)`
	`Receipt BeginReceipt(`
	`  string userName,`
	`  Model.ReceiptKind operation,`
	`  bool print)`

Description	Creates and returns blank document of “Sales receipt” type

Returns	Ecr.Receipt type object		
Parameters	Name	Type	Description
	userName	string	Cashier
	operation	Model.ReceiptKind	Receipt type
	`operat`	`EcrOperator`	Cashier credentials (Name, INN, position)
	`print`	`bool`	Hard copy printout attribute (false — do not print)
	`optionalRequisites`	`object`	"arbitrary" container of fiscal document additional credentials Each FD credential shall be represented by public property with corresponding name and type compatible with the expected, and, at least, available to read (get). See credentials list in notes. The list is subject to extension as FD formats extend.
Notes	Method can also be called when the shift is in "closed" state. In this case, the shift will be opened "automatically". To fill and close the document, "Ecr.ReceiptDoc" methods of returned object are used. Additional credentials transmitted via container: 1. Taxation (`TaxationType`) Taxation system to which the FD belongs. The one specified at ECR registration is used by default. If several taxation systems were specified at registration time, and parameter is not specified, "Default Taxation System" setting will be used. If setting is missing, error is generated. 2. CustomerAddress (`string`) buyer's electronic address (e-mail address or phone number for messages) to deliver FD electronic copy to the specified address upon buyer's request. 3. SenderAddress (string) — sender's electronic address (e-mail address or phone number). Mandatory credential if CustomerAddress is specified. 4. Print (bool) — hard copy printout attribute (false — do not print). Default value is true. CustomerAddress and SenderAddress credentials can also be set by separate Receipt.SetExtra method prior to Receipt.Complete() call. If CustomerAddress and SenderAddress credentials were set when Shift.BeginReceipt is called, overriding thereof in Receipt.SetExtra is not allowed (exception). If CustomerAddress is set (in any way) but SenderAddress is not set, an attempt will be made in the course of Receipt.Complete() execution to use SenderAddress value from ECR registration data		

Example	The example demonstrates use of anonymous type to represent container of “arbitrary” data.

```
//Buyer’s address not required, hard copy not required
Fs.Native.TaxationType docTaxation = ...

var rcpt = m_shift.BeginReceipt(new EcrOperator(MainForm.User),
              (Fw21.Model.ReceiptKind)cbChequeType.SelectedValue,
              new {
                     Taxation = docTaxation,
                     Print = false,
               });


//Buyer’s address needs to be added to FD, hard copy required
Fs.Native.TaxationType docTaxation = ...
string customerAddress = ...
string senderAddress = ...

var rcpt = m_shift.BeginReceipt(MainForm.User,
              (Fw21.Model.ReceiptKind)cbChequeType.SelectedValue,
              new {
                     Taxation = docTaxation,
                     CustomerAddress = customerAddress,
                     SenderAddress = senderAddress,
               });
```

BeginReceipt

Changes in Fiscal Document Format (FDF) dated 03.2018 extend “Correction sales receipt” FD data up to substantial correspondence to “Sales receipt” FD. Creation of “sales receipt” or “correction sales receipt” document is defined by ‘`reason`’ parameter presence or absence.

Specification	**Receipt BeginReceipt(** **EcrOperator operat,** **Model.ReceiptDocument.CorrectionReason reason,** **Model.ReceiptKind operation,** **TaxationType taxation,** **bool print)**		
Description	Creates and returns document of “Correction sales receipt” type		
Returns	Ecr.**Receipt** type object		
Parameters	Name	Type	Description
	operat	EcrOperator	Operator (cashier) credentials
	reason	CorrectionReason	Reason for correction
	operation	ReceiptKind	Receipt type
	taxation	TaxationType	Document taxation system
	print	bool	Hard copy printout attribute (false — do not print)

Example	`var reason = new Fw21.Model.ReceiptDocument.CorrectionReason() {`
	`  DT = new DateTime(2018,1,1),`
	`  Name = string.Empty,`
	`  Number = «123-A»,`
	`  Kind = Fw1`
	`6.Model.CorrectionKind.Request`
	`};`
	`var rcpt = m_shift.BeginReceipt(operat, reason, operation, taxation, true);`
	`...rcpt.Addentry...`
	`...rcpt.Payment...`
	`...rcpt.Complete...`

BeginNonFiscal

Specification	**`NonFiscalBase BeginNonFiscal(Native.CmdExecutor.NFDocType nfdType, string title = null)`** **`NonFiscalBase BeginNonFiscal(Native.CmdExecutor.NFDocType nfdType, bool print, string title)`**		
Description	Creates and returns document of “Non-fiscal” type		
Returns	Ecr.`NonFiscalBase` type object		
Parameters	Name	Type	Description
	nfdType	`Native.CmdExecutor.NFDocType`	Non-fiscal document type
	`print`	`bool`	Hard copy printout attribute (false — do not print)
	title	`string`	Own title. If not specified (null), standard (depending on *nfdType*) is used.

PtintXReport

Specification	**PrintXReport()**		
Description	ECR shift report (X-Report) printout		
Returns	no		
Parameters	Name	Type	Description
	no		

Close

Specification	**`void Close(string operatorName)`** **`void Close(EcrOperator operat)`** **`public void Close(EcrOperator operat, bool print)`**
Description	Closes current ECR shift if current shift is open. Generates fiscal document of “Shift closure report” type. Additionally, currency balance in cash drawer is printed out (if not disabled explicitly by print=false). ECR shift counter is cleared.
Returns	no

Parameters	Name	Type	Description
	userName	string	Cashier name
	operat	EcrOperator	Cashier credentials (Name, INN, position)
	print	bool	Hard copy printout attribute (false — do not print)

McCheck (Ecr.Shift.McCheck)

Properties

Name	Type	Description
State	Fs.FsMcState	

Fs.FsMcState

FS status related to work with tagged goods

Properties

Name	Type	Description
CheckStatus	McCheckStatus	Status of MC under check
NotifFormState	McNotifFormState	Status of tagged goods sales notification generation
CheckNext	McCheckNext	FS: Expected further commands
Confirmed	byte	Number of “approved” (saved in FS pool) TCs. Up to 128.
Included	byte	Number of TCs included in tagged goods sales notification
Usage	byte	Occupation index of tagged goods sales notification storage area (0-3 exponential, 4-full).
NotifsCount	ushort	Number of notifications expecting acknowledgement

McCheckStatus values

Name	Value	Description
Overflow	0	Pool of TCs under verification is full
Ready	1	No TCs to verify
WaitAccept1	2	FsCmd.McStartCheck command issued
WaitAccept2	3	FsCmd.McGetCheckRequest command issued
WaitAccept3	4	FsCmd.McPutCheckResponse command issued

McNotifFormState values

Name	Value	Description
Ready	0	Notification not generated
InProgress	1	Notification is generated
Suspended	2	Notification generation is suspended due to overflow

McCheckNext values (set of attributes)

Name	Value	Description
b1	`0x01`	FsCmd.McStartCheck expected
b2	`0x02`	FsCmd expected. McAcceptCheck
b3	`0x04`	FsCmd expected. McClear
b4	`0x08`	FsCmd expected. McGetCheckRequest
b5	`0x10`	FsCmd expected. McPutCheckResponse
b6	`0x20`	FsCmd expected. McAdd(1)
b7	`0x40`	FsCmd expected. McAdd(2)
b8	`0x80`	FsCmd expected. McAdd(3)

Methods

CheckKm

Specification	**bool CheckKm(string code, Fs.McCheckData info, out Model.McIsmtCheckResult ismtCheckResult, GetConsumerConsetntCallback getConsent)**		
Description	Checks the specified code and adds it to buffer if check was successful or if there is buyer's consent. Adapted to multi-thread execution, to have possibility to ignore MISO response if result of MC verification therein is delayed (see. "Fast" MC verification).		
Returns	`True if MC was added to FS buffer`		
Parameters	Name	Type	Description
	code	string	tag code content for verification
	info	Fs.McCheckData	Additional information on the goods to the tag code for verification at MISO
	ismtCheckResult	out Model.McIsmtCheckResult	Information on MC verification at MISO
	getConsent	bool foo(McCheckFailureResult, Model.McIsmtCheckResult)	callback for request of buyer's consent operation with accounting object with "unclear" MC. The gets complex verification result and info on results of MC verification at MISO as input If the function is not transferred (null), the response is deemed "no" if buyer's "consent" is needed.

BeginCheck

Specification	**McCheckFailureResult? BeginCheck(string code)**		
Description	First step of alternative MC verification procedure (verification in FS) Adapted to multi-thread execution, to have possibility to ignore MISO response if result of MC verification therein is delayed (see. "Fast" MC verification).		
Returns	FS verification result McCheckFailureResult (null — FS verification result is positive)		
Parameters	Name	Type	Description
	code	string	tag code content for verification

IsmtCheck

Specification	**McCheckResult? IsmtCheck(Fs.McCheckData info)**		
Description	Second step of alternative MC verification procedure (verification at MISO) Adapted to multi-thread execution, to have possibility to ignore MISO response if result of MC verification therein is delayed (see. "Fast" MC verification).		
Returns	FS verification result McCheckResult. 'null' means that it is not possible to get info from MISO. If != null, "bits" analysis is required for the lack of positive result. If there is no positive result, buyer's consent to proceed is needed		
Parameters	Name	Type	Description
	info	Fs.McCheckData	Additional information on the goods to the tag code for verification at MISO
	ismtCheckResult	out Model.Model.McIsmtCheckedData	Information on MC verification at MISO. Null in case of positive return means "undefined" result of MISO verification, caused either by MISO response timeout, or by ignoring response thereof due to next MC verification (see "Fast" MC verification).

EndCheck

Specification	**Model.McCheckResult EndCheck(bool approve)**		
Description	Third (final) step of alternative MC verification procedure (MC commit to FS buffer or deletion from buffer). Adapted to multi-thread execution, to have possibility to ignore MISO response if result of MC verification therein is delayed (see. "Fast" MC verification).		
Returns	true — MC commit to FS buffer, false — deletion from buffer		
Parameters	Name	Type	Description
	info	Fs.McCheckData	Additional information on the goods to the tag code for verification at MISO

Clear

Specification	**void Clear()**		
Description	Deletes verification results stored in FS		
Returns	no		
Parameters	Name	Type	Description
No			

McCheckFailureResult

Name	**Value**	**Description**
UndefinedMcType	0	Code was not recognized as subject to check (i. e., as one of the types listed in 2100 credential. Buyer's consent to operation with accounting object with such code is not required (will be ignored). Adding to FS buffer for the checked TCs was not performed. The code can be used when accounting object is being added to sales receipt (correction sales receipt) "on common basis"
FsCheckFailed	1	Negative result of FS verification
FsCheckNa	2	FS verification is not possible MC of this type is not subject to FS verification
FsCheckNoKey	3	FS verification is not possible no verification key
FsCheckNoCheckData	4	FS verification is not possible Verification data (GS1 AI 92, 93) are not valid or missing
FsCheckOther	5	FS verification is not possible Other reason
NotChecked	6	MC verification cannot be performed
CheckCodeFailed	7	Negative result of MC verification code (VC) check
IsmtCheckFailed	8	Negative result of MC verification at MISO

Fs.McCheckData

Name	**Type**	**Description**
McStatusPlan	McStatus	Planned goods status See FFD 1.2 credential 2003
McMode	McMode	Tag code processing mode See FFD 1.2 credential 2102
Item_Quantity	decimal	Goods quantity
Item_Measure	QuantityMeasure	Goods quantity measurement unit See FFD 1.2 credential 2108
Item_PartialQuantity	PartialQuantity	Partial quantity of goods from tagged "package". See FFD 1.2 Credential 1291.

McCheckResult

Corresponds to FD credential 2106 (goods info verification result). Combination of values:

Name	Bit	Description
Checked	0	"0" — tag code not verified by FS and (or) MISO "1" — tag code verified
Ok	1	"0" — MC VC verification result is negative or tag code not verified "1" — MC VC verification result is positive
IsmtCheck	2	"0" — goods status info not received from MISO "1" — status verified by MISO
IsmtOk	3	"0" — info received from MISO that planned goods status is incorrect, or goods status info not received from MISO "1" — info received from MISO that planned goods status is correct
Standalone	4	"0" — MC VC and goods status verification result is generated by ECR functioning in data transmission mode "1" — MC VC verification result is generated by ECR functioning in standalone mode

Model.McIsmtCheckedData

Name	Type	Description
McRequestDt	DateTime	Request date and time See FFD 1.2 credential 2114
McIsmtAnsCode	McIsmtAnsCode	MISO request processing code See FFD 1.2 credential 2105
McIsmtResponse	McIsmtResponse	MISO result See FFD 1.2 credential 2005
McProductStatus	McProductStatus	MISO request processing code See FFD 1.2 credential 2109
McType	McType	Tag code type See FFD 1.2 credential 2100

Ecr.Receipt

Ecr.Receipt object is created as a result of Shift.BeginReceipt methods call. Designed to generate sales receipt. Does not provide comprehensive info on generated receipt, since it is assumed that all info necessary for calling program is supported thereby. Ecr.Receipt methods return additional info which is "concatenated" by the calling program to its info as reference.

Properties

Name	Type	Description
Total	`decimal`	Receipt total amount
UserInfo	`NamedValue`	Sets additional user credential
Recipient	`RecipientData`	Goods recipient
Industry	`Industry`	accounting object industry credential
Operating	`Operating`	Receipt operating credential

Methods

Abort

Specification	**void Abort ()**		
Description	Abort incomplete document		
Returns	no		
Parameters	Name	Type	Description
no			

NewItemPriced

Specification	`ReceiptEntry NewItemPriced (`
	`  string barcode,`
	`  string name,`
	`  Native.CmdExecutor.VatCodeType vatCode,`
	`  decimal price,`
	`  decimal quantity)`		

Description	Prepares commodity item based on unit price for further adding to receipt. Used when unit price controllable by the calling program is needed. ECR calculates the item price by the following formula: price * quantity with mathematical rounding to copecks. See below for comparison NewItemCosted.		
Returns	`ReceiptEntry`		
Parameters	Name	Type	Description
	barcode	string	Goods identifier See Tag code interpretation Mandatory at taggable product registration
	name	String	Product name. Maximum name length 128. If necessary, can be "trimmed" from the right down to acceptable length, to transmit to ECR. Depending on accounting object attribute (1212 = 15,16), the product name is defined by Federal Tax Agency (FFD), and shall be transmitted to ECR as name index "1". ."31".
	`vatCode`	`VatCodeType`	Tax rate code
	`price`	`decimal`	Goods price, all discounts considered
	`quantity`	`decimal`	Goods quantity

NewItemCosted

Specification	`ReceiptEntry NewItemCosted (`
	`  string barcode,`
	`  string name,`
	`  decimal quantity,`
	`  Native.CmdExecutor.VatCodeType vatCode,`
	`  decimal cost)`		

Description	Prepares commodity item based on price of specified goods quantity for further adding to receipt. Used when price per specified quantity controllable by the calling program is needed. E. g., 3 pcs. for 100.00 rubles. ECR calculates the goods unit price rounded to copecks. The difference with backward-calculated price is automatically compensated by ECR by correction (special commodity item) for the amount of the difference. See above for comparison NewItemPriced.		
Returns	`ReceiptEntry`		
Parameters	Name	Type	Description
	barcode	string	Goods identifier See additional note to parameter in NewItemPriced
	name	String	Product name. See additional note to parameter in NewItemPriced
	`quantity`	`decimal`	Goods quantity
	`vatCode`	`VatCodeType`	Tax rate code
	`cost`	`decimal`	Price of commodity item based on quantity thereof

Tag code interpretation

When accounting object instance is created, goods identifier (*Barcode* parameter) is specified in order to add it to "Sales receipt" FD. In accordance with known rules of tag code interpretation, value of this parameter is used to automatically generate `ReceiptEntry` properties related to goods code: `MerchCode`, `Id`. Based on successful tag code interpretation result, credential 1163 (goods code) will be generated automatically for this accounting object.

The attribute of successful tag code interpretation transmitted as not-empty value of *Barcode* parameter in NewItemPriced and NewItemCosted methods, is lack of `ReceiptEntry.Id` property value and configured `ReceiptEntry.MerchCode` property value. If transmitted value of *Barcode* parameter was not successfully interpreted, the value of *Barcode* parameter will be assigned to `ReceiptEntry.Id` property value. Not verified value will be assigned to `ReceiptEntry.Id` property, and this may cause ECR rejection to add the accounting object to "Sales receipt" FD.

Tag codes for which interpretation is supported in Fw21:

- Data in GS1Data format considering specific designated purpose features determined by Center for Development of Advanced Technologies (CDAT, ЦРПТ). A number of commodity groups are tagged by this code, excluding tobacco packs and fur ware.
- Tobacco pack code. 29 symbols in accordance with CDAT format.
- GTIN-8,12,13,14 (EAN, UPC, ITF) shall have correct control digit.
- Excise labels data in EGAIS format.
- Fur ware tagging data. 21 symbols.

Prior to Fw21 tag code interpretation, based on *Barcode* parameter content, based on indirect indicators (basically, length) tag code affiliation with specific type is determined. Depending on tag code

type, *Barcode* data is further interpreted with possibility to get both positive and negative interpretation result.

Top level application can detect tag code type independently, and provide it to Fw21. Tag code type can be detected by code reading device (barcode scanner), received in top level application, and explicitly transmitted further to Fw21. In such case, tag code type is transmitted as prefix to data coded in barcode. Such prefix is also called Barcode Symbology Identifier (BSI). Barcode scanners can have settings that allow to include BSI to data flow being read (together with possibility to get value thereof apart from the main data).

Tag code types (BSI) recognizable by Fw21:

1. Standard (supported by barcode scanners):
    a. `]d2 GS1 DataMatrix (tobacco pack, shoes, medicines, milk products, and further through the list)`
    b. `]Q3 GS1 QR`
    c. `]C1 GS1 Code128`
    d. `]e0 GS1 DataBar`
    e. `]I1 ITF14`
    f. `]E0 EAN13, UPC-A`
    g. `]E4 EAN8, UPC-E`
    h. `]C0 Basic Code128`
2. Extensions (not supported by barcode scanners):
    a. `]dt DataMatrix for tobacco pack`
    b. `]dx DataMatrix АМ ЕГАИС 3.0`
    c. `]Lx PDF417 АМ ЕГАИС 2.0`
    d. `]Xr RFID tagging of fur ware`

Possibility to use BSI to generate tag code on Fw21 level is determine by EcrConfig/ReceiptEntry/DetectBsi setting.

AddEntry

Specification	`decimal AddEntry(ReceiptEntry it)`		
Description	Adds preliminarily created and configured commodity item to the receipt, and finalizes it (blocks further attempts to add discounts/surcharges and additional credentials).		
Returns	Receipt subtotal calculated by ECR. Apart from that, values of `ReceiptEntry` parameter properties become available: • Cost — accounting object cost calculated by ECR • Position — accounting object position number • VatAmount — amount corresponding to VAT rate		
Parameters	Name	Type	Description
	it	`ReceiptEntry`	Commodity item instance returned by NewItemPriced or NewItemCosted calls

AddPayment

Specification	`decimal AddPayment(`
	`  TenderCode Code,`
	`  decimal Amount`
	`)`		

Description	Adds payment to receipt		
Returns	Total amount of all payments added before, including the current payment		
Parameters	Name	Type	Description
	Code	`TenderCode`	Payment facility code
	`Amount`	`decimal`	Amount

SetAdustment

Applied for total correction required when receipt amount is rounded.

Specification	`decimal SetAdjustment(decimal delta, string description)`		
Description	Modifies receipt total by specified value. Initially, receipt total is determined as sum of all prices (Cost) of all accounting objects. This total (base) can be modified by correction amount transmitted in `delta` parameter. Therewith, such modification (decrease or increase) is only possible that does not affect the ruble part of the total. If receipt total is changed int any direction, VAT amount correction is not performed. Specified correction amount is printed on paper receipt. It is not transmitted to FDO and is not present in electronic form of sales receipt.		
Returns	Receipt total calculated by ECR.		
Parameters	Name	Type	Description
	`delta`	`decimal`	Correction amount. Positive or negative
	`description`	`string`	Description of reason for correction. Can be empty

SetExtra

Applied to set additional FD credentials.

Specification	`public void SetExtra(object container)`		
Description	Configures additional credentials not specified when Receipt (Shift.BeginReceipt(…optionalParametes)) are created. If attempt to configure credentials already specified in Shift.BeginReceipt is made, exception is generated. Recall of this method is possible with other credential values. The last value will be used. Call after Complete() generates exception.		
Returns	no		
Parameters	Name	Type	Description
	`container`	`object`	Anonymous class (for inter-version compatibility) containing properties having the same name as the ones of FD credentials being set. The following properties are accepted: • `string` CustomerAddress • `string` SenderAddress The list can be extended in the future versions.
Example	`rcpt.SetExtra(new { CustomerAddress = "foo@gu.da", SenderAddress = "foo@gu.net" });`		

Complete

Specification	`Fs.Native.IfiscalRequisite` **Complete ()**		
Description	Complete FD creation.		
Returns	`Fs.Native.IfiscalRequisite` The returned value can extend the specified type (IfiscalRequisite) by Receipt.QRCodeInfo type inherited from it, which provides access to data used by ECR for sales receipt QR code printout. Check is required for correspondence of returned type to QRCodeInfo type, since returned value can only support IFiscalRequisite type (due to theoretically possible failures in the course of Receipt.QRCodeInfo type object instance generation)		
Parameters	Name	Type	Description
No			

Specification	**void Complete (Native.CmdExecutor.DocEndMode mode)**		
Description	Complete FD creation specifying the ending printout mode		
Returns	Fs.Native.IfiscalRequisite (see additional above)		
Parameters	Name	Type	Description
	mode	DocEndMode	Combination of document ending printout modes.

Specification	**void Complete (Native.CmdExecutor.FootLineData fld, Native.CmdExecutor.DocEndMode mode = Default)**		
Description	Complete FD creation. Printed form will contain data for printout transmitted in parameters. Printout will be performed after QR code and footing		
Returns	Fs.Native.IfiscalRequisite (see additional above)		
Parameters	Name	Type	Description
	fld	FootLineData	Data for printing in the end of receipt (optional text + barcode)
	mode	DocEndMode	Combination of document ending printout modes.

PrintText

Specification	**void PrintText(Native.CmdExecutor.PrintTextMode mode, string text)**		
Description	Prints out the specified text		
Returns	no		
Parameters	Name	Type	Description
	mode	PrintTextMode	reserved
	Text	String	Printout text. Can be separated by “new line” symbol. If paper tape width is exceeded, the rest of line is automatically wrapped to new line. Text length is limited by 255 symbols. Printout is automatically finished by wrapping to new line.

PrintText

Specification	`void PrintText(string text)`		
Description	Prints out the specified text		
Returns	no		
Parameters	Name	Type	Description
	`Text`	`String`	Similar to method with mode parameter (see above) but with the following differences: • Text length is not limited. • Content is not controlled for unprintable symbols. • Wrapping to new line at the end of printout is not performed. • Prior to start and at the end of printout, controlling sequences are transmitted to printer, that reset print settings to initial state: alignment, scale, font thickness and type.

Native.CmdExecutor.VatCodeType

Values

Name	**Value**	**Description**
Vat18Included	1	VAT with calculated 18% rate
Vat10Included	2	VAT with calculated 10% rate
Vat0	3	VAT with 0% rate
NoVat	4	VAT free
Vat18	5	VAT with 18% rate
Vat10	6	VAT with 10% rate

Native.CmdExecutor.TenderCode

Values

Name	**Value**	**Description**
Cash	0	Cash rubles
Max	7	max value of payment facility index.

ReceiptEntry

Represents accounting object (commodity item) data in sale receipt.

Properties

Name	Type	Description
MerchCode	MerchCode	Structured representation of product tag code if tag code interpretation was successful (see Id, Code, CodeBytes)
Cost	decimal	Cost considering all discounts. 1. NewPricedItem: Price * Quantity with mathematical rounding. 2. NewCostedItem: Transmitted value
Id	string	Accounting object identifier specified at creation hereof — NewPriced(Costed)Item. Can be absent if product code (PC) was successfully generated based on it (see MerchCode, Code, CodeBytes)
Name	string	Product name.
Position	int	Position number, starting from 0
price	decimal	1. NewPricedItem: Transmitted value. 2. NewCostedItem: 0. Will be calculated by ECR. Automatic ECR actions to compensate differences against backward-computed cost, are possible.
Quantity	decimal	Quantities
Portion	PartialQuantity	Partial (fractional) quantity specified at registration of piece goods being sold “out of the package”, having the same tag code for all goods from the package. In accordance with FFD, if value for this credential is specified, Quantity credential shall be = 1, Measure = pcs(0) — piece. To set the credential value, it is recommended to use constructor specifying numerator ”**devident**” that determines quantity of goods being sold “out of the package”, and “**divider**” that determines quantity of goods “in the package”: `new` Fw21.Model.**PartialQuantity**(`int` devident, `int` divider).
VatCode	VatCodeType	Product tax rate code
VatAmount	decimal	Tax amount calculated by ECR. Values therein appear after Fw21.Ecr.Receipt.AddEntry() call. Attempt to get AddEntry property values may cause Exception
Measure	QuantityMeasure	Measure of accounting object quantity. Exhaustive list.
Transfer	TransferInfo	Agent participation data at payment transfer to supplier
PaymentKind	ItemPaymentKind	Settlement method
Kind	ItemFlags	Settlement object attribute
Requisite	string	User credential for settlement object.
Industry	Industry	accounting object industry credential
McCheckResult	McCheckResult	Final result of MC verification achieved by IsmtCheck call

Enumerations

ItemPaymentKind

Settlement method attribute (commodity item credential — settlement object)

Name	Value	Description
Unspecified	`0`	Not specified
Prepay	`1`	Full prepayment
PartlyPrepay	`2`	Partial prepayment
Advance	`3`	Advance (gift certificate sale)
Payoff	`4`	Full payment including advance
PartlyLoanCredit	`5`	Partial settlement and loan
LoanCredit	`6`	Delivery on loan
PayCredit	`7`	Loan payment

ItemFlags

Settlement object attributes

Name	Value	Description
Regular	`1`	Regular non-excise goods
Excise	`2`	Regular excise goods (alcohol, cigarettes)
Work	`3`	Work
Service	`4`	Service
Bet	`5`	Gambling bet
BetPayout	`6`	Gambling win payout
Lottery	`7`	Lottery participation
LoteryPayout	`8`	Lottery win payout
NotMaterial	`9`	Intellectual property
Advance	`10`	Payment (advance, loan payment, fines) or Payout (bonuses, etc.)
Comission	`11`	Agent commission

TransferInfo

Funds transfer operation credentials: Supplier [of accounting object] (transfer details), agent, transfer operator.

Many instances related to transfers, contain “**Phones**” property. “`string`” data type is defined for this property. Nevertheless, credential in FD structure corresponding to this property, is usually “reproducible”, i. e., there shall be a possibility to specify several phones. Therefore, the property can be specified as “list” of phone numbers. The following “rules” are specified for that:

- Phone number can only contain digits 0..9 and “+” sign
- Phone numbers are separated by space.
- In FD structure, each phone number “transforms” into separate credential.

Properties

Name	Type	Description
Kind	`AgentType`	Types of agent participation
Supplier	`SupplierInfo`	Settlement object supplier
BankAgent	`BankAgentInfo`	Bank agent info
Agent	`AgentInfo`	Non-bank agent info

Types

AgentType

Types of agent participation in the settlement. Receipt (1057), Goods (1222)

Name	Value	Description
Unspecified	`0`	Not specified
Bank	`0x01`	Bank payment agent
BankSub	`0x02`	Bank payment subagent
Payment	`0x04`	Payment agent
PaymentSub	`0x08`	Payment subagent
Attorney	`0x10`	Attorney
Comissioner	`0x20`	Commissioner
Other	`0x40`	Agent, none of the above

SupplierInfo (Properties)

Name	Type	Description
Name	`string`	Supplier name
TaxId	`string`	Settlement object supplier
Phones	`string`	supplier phones

BankAgentInfo

Bank agent info

Name	Type	Description
Operation	`string`	payment agent operation
Phones	`string`	agent phones
Operator	`TransferOperator`	Bank agent transfer operator

TransferOperator

Bank agent transfer operator data

Name	Type	Description
Name	`string`	Cash transfer operator name
TaxId	`string`	Cash transfer operator INN
Address	`string`	cash transfer operator address
Phones	`string`	transfer operator phones

AgentInfo

Non-bank agent data = payment agent (and other roles of agent-commissioner-attorney)

Name	Type	Description
Operation	`string`	payment agent operation
Phones	`string`	agent phones
AcceptorPhones	`string`	Payment receipt operator phones

ReceiptDocument.CorrectionReason

Properties

Name	Type	Description
Kind	`CorrectionKind`	Correction type: `Self`: Independently `Request`: By request
Name	`string`	Correction description
DT	`DateTime`	Date of correction settlement (not later than current)
Number	`string`	Number of tax authority instruction if correction is made by instruction

Ecr.CorrectionCause

FD credentials defining “Correction sales receipt” (unlike just “sales receipt”).

Properties

Name	Type	Description
Name	`string`	Correction description
Cause	`DateTime`	date of settlement under correction (not later than current date)
Number	`string`	number of tax authority instruction if correction is made by instruction

Ecr. NonFiscalBase

Ecr. object NonFiscalBase is created as a result of Shift methods. BeginNonFiscal. Object instances returned by these methods control creation and printout of non-fiscal documents of different types: cash deposit and withdrawal to / from cash register drawer, reports in arbitrary format.

Methods

PrintText

Specification	`void PrintText(string text)`
Description	Prints out the specified text. Unlike similar method of Receipt type, has no limitations on length of text being transmitted. Printout into non-fiscal document uses ECR command that does not add “end of line” sequence to data being transmitted to printer. On one hand, this increases flexibility to generate required receipt type (considering printer specific features). On the other hand, it "imposes requirements" to understanding these specific features. Known “specific features”: 1. Printer only prints out barcode in the beginning of line. Text printed out BEFORE barcode printout, shall explicitly contain “end of line” sequence (Environment.NewLine). 2. Epson-compatible esc-sequences can be transmitted to generate text for printing. See examples below.

Returns	no		
Parameters	Name	Type	Description
	text	`string`	“New line”, “carriage return” symbols are permissible. Long lines are automatically wrapped to new line at while printing.
Examples of formatting esc-sequences	Each example contains two esc-sequences: original — enables mode, final — disables mode. In many cases, new line disables previously enabled mode, but not always so. Combination of several modes is possible.		
	Bold (emphasize, bold)		
	`"\x1b\x45\x01" + "Text example" + "\x1b\x45\x00"`		
	Double width and height		
	`"\x1d\x21\x11" + "Text example" + "\x1d\x21\x00"`		
	Triple width and height		
	`"\x1d\x21\x12" + "Text example" + "\x1d\x21\x00"`		
	Line center alignment		
	`"\x1b\x61\x01" + "Text example" + "\x1b\x61\x00"`		
	Line right alignment		
	`"\x1b\x61\x02" + "Text example" + "\x1b\x61\x00"`		
	Line left alignment		
	`"\x1b\x61\x00" + "Text example"`		

AddTender

Specification	`void AddTender(Tender tender)`		
Description	Adds amount of specified payment facility to the document. Operation is only permissible for non-fiscal documents of Income and Outcome type		
Returns	no		
Parameters	Name	Type	Description
	tender	`Tender`	Payment facility code and amount. Payment facility with this code shall be configured in ECR as “cash”.

Complete

Specification	`void Complete(Native.CmdExecutor.DocEndMode mode)`		
Description	Completes generation and printout of non-fiscal document		
Returns	no		
Parameters	Name	Type	Description
	mode	`DocEndMode`	Document printout completion options

Native.CmdExecutor.NFDocType

Values

Name	Description
Income	Non-fiscal document of cash deposit to cash register
Outcome	Non-fiscal document of cash withdrawal from cash register
Report	Non-fiscal document with arbitrary content

Fs.Native.IFiscalRequisite

Fiscal document credentials generated by FS. Objects implementing this interface are returned when any type of fiscal document is created, not only documents within Shift. However, account shall be taken that Number property is only valid for documents of Shift group. When Number property of object received when other documents are closed (e. g., registration ore registration data correction) is accessed, exception will be generated.

Properties

Name	Type	Description
Number	ushort	Receipt number in the shift.
FiscalNumber	uint	Fiscal document number in FS.
FiscalSignature	uint	Document fiscal attribute

Service (Ecr.Service)

Properties

Name	Type	Description
Header	Cliche	Control of printed document headers
Footer	Cliche	Control of printed document footers
CodeImager	CodeImageCreator	object factory providing barcodes printout.

Methods

BeginRegistration

Returns object controlling creation of “ECR registration report” fiscal document

Successful call of the method is only possible for ECR in “ECR registration required” and “FS registration required” state.

Calling program shall fill all registration parameters by setting corresponding properties of returned object. Then, call Complete() method of returned object to generate ECR commands sequence resulting in FD generation.

RegId property must be entered by user fully independently, including check numbers, without autofill by program.

ECR will only perform registration operation if check digits of register number (last 6 digits) entered by user matches the value calculated based on:

- first 10 digits of RegId,
- TaxId
- EcrId.

BeginChangeFs

Returns object controlling creation of “ECR parameters change due to FS replacement” fiscal document.

Successful call of the method is only possible for ECR in “FS registration required” and “FS replacement required” state. Filling is not required. All necessary info for generation of FD of corresponding type, is taken from ECR.

In case of “emergency” (ECR “memory loss” after repair set in new FS) it can be filled with full registration data.

call Complete() method of returned object to generate ECR commands sequence resulting in FD generation.

BeginRegChange

Returns object controlling creation of “ECR parameters change without FS replacement” fiscal document.

Calling program can change some registration parameters by setting corresponding properties of returned object. Then, call Complete() method of returned object to generate ECR commands sequence resulting in FD generation.

MaintainReg

Method is designed to “control” registration data stored in ECR memory.

MaintainRegType.AcceptFs — for ECR that “lost memory” after repair where FS is installed that previously worked in that ECR. All registration information from FS is moved to ECR memory.

MaintainRegType.Clear — for ECR having “unnecessary” registration data in memory.

BeginReportFs

Returns object controlling creation of “Settlement status report” fiscal document Call Complete() method of returned object to generate ECR commands sequence resulting in FD generation.

The report is printed out

BeginCloseFs

Returns object controlling creation of “Fiscal memory device closure report” fiscal document Call Complete() method of returned object to generate ECR commands sequence resulting in FD generation.

Moves FS to post-fiscal phase (prior to replacement thereof in ECR by new device)

AbortDoc

Aborts any document that was not completed due to any reason

GetParameter

Gets ECR parameter value.

SetParameter

Sets ECR parameter value.

SetTender

Sets ECR payment facility data. Current data can be received from EcrCtrl.Info.GetTendersList().

LoadPictures

Loads “pictures” to printer from specified files in bmp, png, jpg, pim formats for further printing. File names are transmitted as string array. Deletes all previously loaded pictures. Sequence number from 1

to 8 is assigned to each loaded picture. “Picture” printout is performed by specifying number thereof in special sequence being transmitted to PrintText method, or in Header\Footer content: [p<n>].

Recommendations for “pictures”. Max size in pixels: width — 576, height 288x8. Max height limits the total height of all “pictures” being loaded. Two-color palette. Use dithering to simulate semitones.

Fw21 optimizes repeated load of the same set of “pictures” by creating cache in local host data (copy of data once loaded to ECR with this particular serial number). If cache exists and matches the data loaded again, actual download is not performed. To force repeated “pictures” download if valid cash exists, delete file with cache data. Cache folder: “%localappdata%\Pilot\Fw21”. Cache file name: “<ecrId>.images.cache”.

Ecr.Service.CodeImager

The only Create method creates instances of objects enabling printout of different types of barcodes at ECR printer.

Methods

Create

Specification	`CodeImage Create(CodeSymbology sym)`		
Description	Returns instance of objects enabling printout of specified type of barcode at ECR printer.		
Returns	Ecr.CodeImage		
Parameters	Name	Type	Description
	sym	CodeSymbology	EAN13, EAN8, UPCA, UPCE, Code39, Code128, QR,

Ecr.CodeImage

Properties

Name	Type	Description
Height	Int	Image height in millimeters (5-31). For QR — desired image height in millimeters (10-70).
Width	Int	Code width. For QR — desired width in millimeters (10-70). For one-dimension codes — module thickness factor (min bar thickness) (1-6).
HorizontalAlignment	HAlignmentKind	Left Center Right

HRPosition	HRPositionKind	Methods of source coded data printout (not applicable to QR):
		`Above,`
		`Below,`

Methods

Print

Specification	**void Print(string text)**		
Description	Codes source text and prints barcode image thereof at ECR printer with set width, height, alignment parameters. Values of specified parameters and suitable text for coding are determined by selected barcode type.		
Returns	no		
Parameters	Name	Type	Description
	Text	String	Text being coded by barcode. Text content and length limitations are determined by selected barcode type.

Commands(Ecr.Native)

This object's interface corresponds to commands description in "ECR programming guide" document in relation to methods starting from "Cmd…" prefix, and considering the following:

1. Data types are used in accordance with "ECR and Fw21 module data types matching" section.
2. PASSWORD and MODE parameters are omitted.
3. If ECR error is received, exception is generated that includes:
    a. ECR error code
    b. ECR error code description

Methods

CmdGetStatus

Specification	`byte[] CmdGetStatus()`		
Description	Get ECR status		
Returns	4 bytes of ECR status		
Parameters	Name	Type	Description
No			

CmdGetSerial

Specification	`string CmdGetSerial()`		
Description	Get ECR serial number		
Returns	String containing ECR serial number		
Parameters	Name	Type	Description
No			

CmdSetTimer

Specification	`CmdSetTimer(DateTime dateTime)`		
Description	Set ECR clock		
Returns	no		
Parameters	Name	Type	Description
	dateTIme	DateTime	New value of ECR clock

CmdSetTimer

Specification	`Fs.Native.StatusData CmdFnGetStatus()`		
Description	Get FS status		
Returns	Object describing Fiscal Memory Device status		
Parameters	Name	Type	Description
No			

Fs.Native.StatusData

Properties

Name	Type	Description
`Phase`	`LifecyclePhaseType`	FS lifecycle phase status
`CurrentDocType`	`DocumentType`	Current document (type)
`CurrentDocReceived`	`bool`	Document data False — no document data; True — document data received
`ShiftOpened`	`bool`	Shift status False — shift closed; True — shift opened
`Alerts`	`AlertType`	Warning flags
`LastDocTime`	`DateTime`	Date and time of the last document
`FsId`	`string`	FS serial number
`LastDocNum`	`uint`	Last FD number

LifecyclePhaseType

Values

Name	Value	Description
`Manufactor`	0	Setting
`ReadyToReg`	1	Readiness for fiscalization. FS configured.
`Registred`	3	Fiscal mode open. (FD adding possible)
`PostRegistred`	7	Post-fiscal mode, FD is being transmitted to FDO (upon fiscal mode closure, FDs are left that have not been transmitted to FDO)
`Readonly`	15	(It is only possible to) Read data from FS Archive. Fiscal data transmission to FDO is completed

DocumentType

Values

Name	Value	Description
NoDocument	0	No open document
Reg	1	ECR registration report
OpenShift	2	Shift opening report
Receipt	4	Sales receipt
CloseShift	8	Shift closure report
CloseReg	16	Fiscal mode closure report
RegChangeWithFS	18	Report on ECR parameters change due to FS replacement
RegChange	19	Report on ECR parameters change (without FS replacement)
ReciptCorrection	20	Correction sales receipt
ShiftState	23	Current accounting status report

AlertType

Values

Name	Value	Description
None	0	
TreeDayToTheEnd	1	Urgent CC replacement (3 days to validity period expiration)
OneMonthToTheEnd	2	CC resource exhausted (30 days to validity period expiration)
ArchiveNearFull	4	FS memory overflow (FS Archive is 90 % full)
AcknowledgeTimeout	8	FDO acknowledgement timeout expired
CriticalError	128	Critical FS error

CmdFnGetTerm

Specification	`DateTime CmdFnGetTerm()`		
Description	Get FS validity period		
Returns	FS validity expiration date		
Parameters	Name	Type	Description
No			

CmdFnGetVersion

Specification	`Fs.Native.VersionData CmdFnGetVersion()`
Description	Get FS version
Returns	Object of type `Fs.Native.VersionData`

Parameters	Name	Type	Description
No			

Fs.Native.VersionData

Properties

Name	Type	Description
`Version`	string	FS software version string
`IsProduction`	bool	FS software type False — debug version; True — production version

CmdFnRegistration

Specification	`Fs.Native.IFiscalRequisite CmdFnRegistration(RegData data)`
Description	ECR registration / re-registration
Returns	Object of type `Fs.Native.IFiscalRequisite`

Parameters	Name	Type	Description
	`data`	`RegData`	Parameters of registration document or registration parameter modification document (including FS replacement).

Fs.Native.IFiscalRequisite

Properties

Name	Type	Description
`Number`	ushort	Document number in shift (may not be applicable to some document types)
`FiscalNumber`	uint	Fiscal document number
`FiscalSignature`	uint	document fiscal attribute

RegData

Properties

Name	Type	Description
`Type`		registration type
`TaxId`	`string`	(user's) INN
`RegId`	`string`	ECR registration number
`Taxations`	`TaxationType`	Taxation code
`Mode`	`RegModeType`	Operation mode
`tlvs`	`byte[]`	Additional registration document data in TLV format

RegType

Values

Name	Value	Description
`First`	0	The first and the only registration with RegId data
`ChangeFs`	1	re-registration due to FS replacement
`ChangeOfd`	2	re-registration due to FDO INN change
`ChangeOwner`	3	re-registration due to change of installation address or user data
`ChangeOwnerAndOfd`	4	re-registration due to FDO INN change, and change of installation address or user data

TaxationType

Values

Name	Value	Description
`General`	1	General
`Revenue`	2	Simplified Revenue
`Profit`	4	Simplified Revenue less Expenses
`Charge`	8	Unified Tax on Imputed Income
`Agro`	16	Unified Agricultural Tax
`Patent`	32	Patent Taxation System

RegModeType

Values

Name	Value	Description
`Crypto`	1	Encryption
`Standalone`	2	Standalone mode
`Automatic`	4	Automatic mode
`Service`	8	Application in Services
`FormMode`	16	AF (БСО) mode (1) otherwise Receipt mode (0)
`PaymentAgent`	32	Application by payment agents (subagents)
`BankAgent`	64	Application by bank agents (subagents)

CmdFnCloseFiscalMode

Specification	`Fs.Native.IFiscalRequisite CmdFnCloseFiscalMode(byte[] tlvs)`		
Description	FS fiscal mode closure		
Returns	Object of type `Fs.Native.IFiscalRequisite`		
Parameters	Name	Type	Description
	`tlvs`	`byte[]`	Additional document data in TLV format

CmdFnGetShift

Specification	`Fs.Native.ShiftData CmdFnGetShift()`		
Description	Get FS shift parameters		
Returns	Object of type `Fs.Native.ShiftData`		
Parameters	Name	Type	Description
No			

Fs.Native.ShiftData

Properties

Name	Type	Description
`Open`	`byte`	0 — shift closed 1 — shift open
`Number`	`ushort`	Shift number. If the shift is closed, number of the last closed shift, if open — number of current shift.
`DocCounter`	`ushort`	If the shift is closed, number of documents in the previous closed shift (0 if this is the first shift). If the shift is open but there are no receipts, then 0. In other cases, number of the last generated receipt.

CmdFnCloseFiscalMode

Specification	`Fs.Native.IFiscalRequisite CmdFnCloseFiscalMode(byte[] tlvs)`		
Description	FS fiscal mode closure		
Returns	Object of type `Fs.Native.IFiscalRequisite`		
Parameters	Name	Type	Description
	`tlvs`	`byte[]`	Additional document data in TLV format

CmdShiftBegin

Specification	`Fs.Native.IFiscalRequisite CmdShiftBegin(byte[] tlvs)`		
Description	Start shift		
Returns	Object of type `Fs.Native.IFiscalRequisite`		
Parameters	Name	Type	Description
	`tlvs`	`byte[]`	Additional document data in TLV format

CmdShiftEnd

Specification	`Fs.Native.IFiscalRequisite CmdShiftEnd(byte[] tlvs)`		
Description	Close the open shift		
Returns	Object of type `Fs.Native.IFiscalRequisite`		
Parameters	Name	Type	Description
	`tlvs`	`byte[]`	Additional document data in TLV format

CmdDocBegin

Specification	`void CmdDocBegin(byte[] tlvs)`		
Description	Start “Receipt” type document		
Returns	no		
Parameters	Name	Type	Description
	`tlvs`	`byte[]`	Additional document data in TLV format

CmdDocAddItem

Specification	`void CmdDocAddItem(byte[] tlvs)`		
Description	Add goods to open document		
Returns	no		
Parameters	Name	Type	Description
	`tlvs`	`byte[]`	ALL product data in TLV format, including discounts/surcharges and additional credentials

CmdDocDelItem

Specification	`void CmdDocAddItem(byte[] tlvs)`		
Description	Reverse commodity item		
Returns	no		
Parameters	Name	Type	Description
	`tlvs`	`byte[]`	ALL product data in TLV format, including discounts/surcharges and additional credentials

CmdDocEnd

Specification	`Fs.Native.IFiscalRequisite CmdDocEnd(DocEndData data)`		
Description	Close document		
Returns	`Fs.Native.IFiscalRequisite`		
Parameters	Name	Type	Description
	data	`DocEndData`	Mandatory document data

DocEndData

Properties

Name	Type	Description
`Operation`	`ReceiptKind`	Operation type
`Total`	`ulong`	Receipt total (in copecks)
`tlvs`	`byte[]`	additional document data

Model.ReceiptKind

Values

Name	Value	Description
`Income`	1	Income
`IncomeBack`	2	Income return
`Outcome`	3	Expense
`OutcomeBack`	4	Expense return

CmdDocAbort

Specification	`void CmdDocAbort()`		
Description	Abort document		
Returns	no		
Parameters	Name	Type	Description
no			

CmdFnReport

Specification	`Fs.Native.ReportRequisite CmdFnReport(byte[] tlvs)`
Description	Accounting status report
Returns	`Fs.Native.ReportRequisite`

Parameters	Name	Type	Description
	`tlvs`	`byte[]`	Additional document data in TLV format

Fs.Native.ReportRequisite

Properties

Name	Type	Description
`Base`	`IFiscalRequisite`	Fiscal credentials
`NoAckCounter`	`uint`	Number of not acknowledged documents
`FirstNoAck`	`DateTime`	Date of first not acknowledged document (YY, MM, DD)

Fw21(Fw21.Native)

Properties

Name	Type	Description
Encoding	`Encoding`	Text encoding in ECR (cp-866)
Password	`string`	Password within ECR data exchange. Default value is “0000”, which corresponds to factory settings. Password can be changed in ECR by Fw21Adm utility. If password is changed in ECR, application shall install this property to appropriate value.

Events

Name	Description
OnAlertReceived	Asynchronous. ECR events requiring attention of "external" SW. Event argument is type `Fw21.EcrCtrl.AlertArgs`
OnSessionCompleted	Synchronous. Generated upon completion of ECR data exchange session. Contains info already used for exchange data recoding to log file. Can be used by external application for alternative log.

Fw21.EcrCtrl.AlertArgs

Properties

Name	Type	Description
Alerts	`AlertType`	Event types (any combination of values below): **FiscalStorage** (0x1000) — Fiscal memory device status has at least one active warning flag. To clarify, get FS status. **OfdMessage** (0x0200) — FS detected message for ECR in fiscal data operator acknowledgement. Message is buffered and can be received by separate command.
Accepted	`bool`	Event processing acknowledgement. Event handler shall set this property to true if it does not “want” to get this event at each ECR exchange. After Accepted=true is set, next event will only be received if “Alerts” flags set is changed (excluding Alerts=None change)

Annex 1

Module settings are stored and controlled in accordance with .Net Framework "policy".

Fw21 module configurable parameters

Module has possibility for parallel control of several CREs, each connected to "control" PC via separate port. Therefore, module parameters are divided into two groups: "global" and "local".

"Local" parameters are related to one of the possible connection instances. "Global" parameters are related to any of established connections. "Global" parameters are marked as *) below

Parameter	x-path	description
Data and protocols catalog *)	EcrCtrlConfig/@log-root	Full or relative path to folder that will be root for all files created for the purpose of module operation diagnostics. If the folder does not exist, it is created by the module. If it is not possible to create the folder and access it, module will not be operational. No default value. This means that root catalog for logs is %localappdata%\Pilot\Fw21. Depending on operational environment, this can be, e. g., XP: C:\Documents and Settings\<user>\Local Settings\Application Data\Pilot\Fw21 Windows 7-10: C:\Users\<user>\AppData\Local\Pilot\Fw21
	EcrCtrlConfig/@log-journal-name	Full or relative path (relative to log-root) of log files (*.log). If not set (absent), the name used is Logs-<connection_name>
	EcrCtrlConfig/@log-data-name	Full or relative path (related to log-root) of application data files. If not set (absent), the name used is Data-<connection_name>. Relevant for corresponding ECR #4 parameter value.
	EcrCtrlConfig/@log-data-keep-days	Store messages transmitted to FDO for specified number of days. Permissible values: -1, 7..1000. Value -1 has no limits (data is not deleted). Default value (including "unacceptable") — 200 days. Specified number of days does not include current date. Data date is determined by folder name. Folder name change "from outside" can cause both deletion

		thereof and lack of possibility to delete by fw21 module. Deletion is initiated at fw21.dll module activation, and at the change of date. Deletion is registered in FO log. As well as errors and impossibility to delete. Setting is relevant for corresponding ECR #4 parameter value.
Runtime settings	EcrCtrlConfig/FW16Run	
	../RepeatOnPrinterError	Processing of printer error (code=8) rarely occurring at execution of specific commands.
	RepeatOnPrinterError/@count="5"	Number of repetitions of erroneous command
	RepeatOnPrinterError/@interval-ms	Value of delay between repetitions, in milliseconds
Connections*)	EcrCtrlConfig/Instances	List of possible connection instances
Connection	../Instance	One of possible connection instances
Connection name	../Instance/@name	This name is specified when EcrCtrl.Init(name) method is called. If name missing in configuration is specified at methods call, connection with "default" name will be used, or (if none) first name on the list, or (if no connections in configuration) autoconfig will be created with "COM1, 57600" parameters.
Connection parameters	../Instance/FW16	
Port name	../FW16/PortName	E. g., Com1, Com9
Port rate	../FW16/Baudrate	Possible values are 57600, 115200 for ports with physical RS-232 interface. For ports with physical USB interface, this parameter is not relevant. It is recommended to set it to 115200. Actual exchange rate via USB connection is controlled by OS and may exceed 115200.
Default taxation system	EcrCtrlConfig/DefaultReceiptTaxation	Taxation system to which sales receipt (AF) will be assigned if: 1. ECR is registered to use several taxation systems, and... 2. Taxation system was not explicitly specified at sales receipt (AF) creation.
Use cashier's positions and name	EcrCtrlConfig/AddPositionToCashierName	Indicates necessity to add cashier's position (if specified) in front of cashier's name when data is being transmitted to ECR
Log parameters	EcrCtrlConfig/Log	

	Log/@prefix	Indicates log type to which the setting belong. That is, there can be several EcrCtrlConfig/Log elements in config. Each for its own separate log. Log types: • na — any log type for which "own" EcrCtrlConfig/Log element is not defined • FP — main ECR control thread • FO — thread of data sending to FDO (Relevant for corresponding ECR parameter #4 value.) • FC — MISO interaction thread (Relevant in presence of "tagging" attribute specified at ECR/FS registration) • FK — Russian National Classification of Products (OKP) interaction thread. • FM — remote control thread
	Log/@keep-days	Store log files for specified number of days. Permissible values are 1..1000. Default value is 7 days.
	Log/@level	extent of data detail in log files (default value is general): • brif — the log only reflects SW activation fact and errors thereof. • general — brief execution tracing • verbose — execution tracing with hexadecimal data representation.
Autostart	EcrCtrlConfig/Start/…	Actions performed by Fw21 module at ECR connection in the course of Init method execution. Actions are performed in the order they are listed in the document. Execution order is not modified by settings. **Attention!** Syntax error at action parameters setting in the may cause action and, possibly, all actions specified in action settings, being ignored. Without error detail in the log.
	…/SyncClock/@max-sec-difference	Synchronize ECR time with host time if the difference between them is less than specified number of seconds. If value < 1 is specified, synchronization is not performed.
	…/Pictures/File[]	List of elements containing full path to file with allowable graphical image. The

		first 8 files are accepted for downloading.
	…/Header/Line[] …/Footer/Line[]	List of elements containing one text string used for document header/footer printout. Strings are transmitted to Cash Register machine in the order specified in the settings.
	…/HeaderFile …/FooterFile	Path to text file containing header strings. File encoding is utf-8. Only the first 600 symbols are used AS IS (without content analysis, including endline). If valid file is set, …/Header/Line[] (…/Footer/Line[]) setting is ignored.
	…/Parameters/Ecr[]	List of Cash Register machine parameters being set <Ecr index="" value=""/> where • index — parameter number from allowable list • value – allowable parameter value Parameters are set in the order specified in the settings. **Attention!** It is not recommended to set parameters that become active after ECR connection or (rather) after ECR power off/on, in “Autostart” settings.
Additional FD credentials	FdOptionals	List of not mandatory, from the perspective of Federal Tax Agency, fiscal document credentials that are not included by ECR into electronic FD but shall be enabled upon user request. **Attention!** At forming this settings section, it is recommended to use the list of credentials and rules of inclusion thereof into FD approved by Federal Tax Agency.
	FdOptionals\Fd[doc-tag="DocReceipt"]	Additional credentials for “Sales receipt” document
	..\Fd\Optionals\Tag	One credential value. Credential is specified by enumeration element name TLVTag.
	EcrConfig/ReceiptEntry	“Accounting object (1059)” credential generation settings
	…DetectBsi/@enabled	BSI usage setting for “Goods code (1163)” credential generation

		`Enabled = true` — use BSI to generate 1163 credential for all known BSIs other than listed in `Exclude` elements `Enabled = false` – do not use BSI to generate 1163 credential Default value: true
	`…DetectBsi/Exclude`	With `DetectBsi/@enabled` setting enabled, exclude specified BSI to generate 1163 credential. The `Exclude` credential can be specified several times, with one of the following values: • `BsiCustomGS1DataMatrix` • `BsiGS1DataMatrix` • `BsiGS1Ucc128` • `BsiGS1Qr` • `BsiGS1Databar` • `BsiITF14` • `BsiTabacoBox` • `BsiEgais30` • `BsiEAN13` • `BsiEAN8` • `BsiEgais20` • `BsiRFIDFurs` Default value includes (codes not related to taggable goods): • `BsiGS1Ucc128` • `BsiGS1Qr` • `BsiGS1Databar` • `BsiITF14` • `BsiEAN13` • `BsiEAN8`

Example

of Fw21 module settings section inclusion into main application settings

```
<configuration>
  <configSections>
    <sectionGroup name="applicationSettings" type="System.Configuration.ApplicationSettingsGroup, System,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
      <section name="Fw21.Properties.Settings" type="System.Configuration.ClientSettingsSection, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
    </sectionGroup>
  </configSections>
  <applicationSettings>
    <!—Other application settings-->
    <Fw21.Properties.Settings>
      <setting name="EcrConfig" serializeAs="Xml">
        <value>
          <EcrCtrlConfig
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                log-root=""
                log-data-keep-days="-1">
            <FW16Run>
              <RepeatOnPrinterError count="5" interval-ms="3000" />
              <IgnoreFfdRestrictions>true</IgnoreFfdRestrictions>
            </FW16Run>
          <Instances>
            <Instance name="default">
              <FW16>
                <PortName>COM1</PortName>
                <Baudrate>38400</Baudrate>
              </FW16>
            </Instance>
          </Instances>
          <DefaultReceiptTaxation>General</DefaultReceiptTaxation>
          <AddPositionToCashierName>false</AddPositionToCashierName>
            <FdOptionals>
              <Fd doc-tag="DocReceipt">
                <Optionals>
                  <Tag>LocationAddress</Tag>
                </Optionals>
              </Fd>
            </FdOptionals>
            <Log keep-days="7" level="general" />
            <Start>
              <SyncClock max-sec-difference="300" />
              <Pictures>
                <File>D:\Projects\OFD\pic1.pim</File>
                <File>D:\Projects\OFD\pic2.png</File>
              </Pictures>
              <Header>
                <Line>[p2]</Line> <!-- pic2.png -->
              </Header>
              <Footer>
                <Line>[p1]</Line> <!-- pic1.pim -->
              </Footer>
              <Parameters>
                <Ecr index="55" value="55"
                  description="Printout in xz-reports of cumulative totals (0-do not print, 55-
print)"/>
              </Parameters>
            </Start>
            <ReceiptEntry>
              <DetectBsi enabled="false">
            </ReceiptEntry>
          </EcrCtrlConfig>
        </value>
      </setting>
    </Fw21.Properties.Settings>
    <!—other application settings-->
  </applicationSettings>
  <startup>
    <supportedRuntime version="v2.0.50727"/>
  </startup>
</configuration>
```