

ATM Controller R2

Contents

ATM CONTROLLER: AN INTRODUCTION	1
CHAPTER 1. ATM CONTROLLER DICTIONARIES	2
Customising ATM Controller Dictionaries	2
ATM Types Dictionary	2
ATM Denominations Dictionary	3
Fixed Dictionaries	5
ATM Protocols Dictionary	5
ATM Operations Dictionary	6
ATM Hardware Types Dictionary	7
ATM Message Types Dictionary	8
CHAPTER 2. DESCRIPTION OF A NEW ATM AND ITS CONFIGURATION	11
Setting up an ATM Contract and its Device	11
Setting up the Executable Range of ATM Operations	11
Configuring ATM Hardware Components	12
Specifying Encryption Keys	14
Enabling MAC-signature Mode	16
Additional Device Parameters	16
Configuring the ATM Connection with the WAY4 Host	16
Preparing and Loading ATM Configurations	17
Preparing the Configuration Files	17
Sending a Configuration to an ATM	19
Sending a Configuration to all ATMs	19
CHAPTER 3. CASH DISPENSATION	20
Cash Dispensation: Configuring the ATM	20
Cash Dispensation: Configuring the Receipt	20
CHAPTER 4. ATM CASH ACCEPTANCE	22
ATM Cash Acceptance: Configuring the ATM	22
ATM Cash Acceptance: Configuring the Receipt	23
CHAPTER 5. RECEIVING BALANCES AND MINI-STATEMENTS	24
Configuring Balances on Card Accounts	24
Configuring Mini-statements on Card Accounts	24
Configuring the Screen Display	25
CHAPTER 6. CHANGING THE PIN CODE	26
Changing the PIN Code: Configuring the ATM	26
Changing the PIN Code: Configuring the Receipt Format for PIN Code Changes	26
CHAPTER 7. AUTOMATIC REVERSAL MESSAGE CREATION	28
CHAPTER 8. ADMINISTRATIVE OPERATIONS AND REPLENISHMENT	29
Financial Cycles	29
Configuring the ATM Replenishment Receipt Format	29
Configuring the Replenishment Officer's Receipt for Collecting Client-Deposited Funds	30
CHAPTER 9. SETTLEMENT SCHEME FOR ATM OPERATIONS	31
CHAPTER 10. ATM CONTROLLER CONFIGURATION FILES	34
Controller Configuration File	34
Configuration Files for Controller Interaction with the ATM	35
Principles of working with configuration scripts	36
Request processing configuration file	37
Response processing configuration file	41
Configuring receipt and screen templates	46

ATM Controller: an Introduction

The ATM controller is a software component operating on the WAY4 Transaction Switch Platform.



The ATM controller is used to support interaction between an ATM network and a processing centre. This interaction includes transmitting management commands to ATMs, receiving messages from ATMs and transmitting response codes, as well as other functions.

This document is intended for WAY4 system administrators (banks or processing centre employees) responsible for configuring the ATM network.

In working with this document, we recommend that users refer to the following source material from OpenWay's documentation series:

- DB Manager Manual
- "WAY4™ Transaction Switch. Platform Overview"
- Daily Procedures
- Acquiring Module User Manual
- Issuing Module User Manual
- ATM Network Monitoring
- WAY4™ Service Packages
- Configuration of Client Messages

This document uses the following conventions:

- Field labels in screen forms are shown in *italics*.
- Button labels in screen forms are indicated in square brackets, for example [Approve].
- Sequences for selecting user menu items are shown with arrows, such as "Issuing → Contracts Input & Update";
- Sequences for selecting system menu items are shown with arrows, as in "Database => Change password";
- Key combinations used in DB Manager are shown in angular brackets, such as <Ctrl>+<F3>.
- Warnings indicating the danger of incorrect input are indicated with the symbol .
- Messages marked with  sign contain information about important features, additional facilities or the optimal use of certain functions of the system.

Chapter 1. ATM Controller Dictionaries

Dictionaries are important sources of information used by the ATM controller. Dictionaries are WAY4 database tables containing one type of information; for example, dictionaries for ATM types, for ATM messages, etc.

WAY4 uses two kinds of dictionaries:

- Custom – dictionaries whose content can be changed by users.
- Fixed – dictionaries whose content may only be changed by OpenWay representatives; in some cases dictionary data may be changed by bank or processing center specialists under the supervision of OpenWay representatives.

Customising ATM Controller Dictionaries

ATM Types Dictionary

All types of ATMs interacting with WAY4 must be registered in a special ATM types dictionary.

An ATM type is set by selecting it from a list of options during device configuration (see the section "Configuring Devices" in the document "Acquiring Module User Manual").

The ATM type dictionary (see Fig. 1) is accessed by selecting the following options in the user menu "Full → Configuration Setup → Merchant Device Setup → ATM Types".

Code	Name	Brand	Model	Conf File	Outgoing Console	Cassette Account Cod	Cash Account Code	Remaining Account Cod	EMV Conf File
NCR5070A	NCR 5070 Ange	NCR3G-ASD	5070	ndc3g.cfg	Outgoing Console	ATM Cassette	Cash Dispenser		
NCR5084	NCR 5084	NCR3G	5084	ndc3g.cfg	Outgoing Console	ATM Cassette	Cash Dispenser		
NCR5084A	NCR 5084 Ange	NCR3G-ASD	5084	ndc3g.cfg	Outgoing Console	ATM Cassette	Cash Dispenser		
NCR5305A	NCR 5305 Ange	NCR4G-ASD	5305	ndc.cfg	Outgoing Console	ATM Cassette	Cash Dispenser		
NCR5670	NCR 5670	NCR	5670	ndcplus.cfg	Outgoing Console	ATM Cassette	Cash Dispenser		
NCR5674	NCR 5674	NCR	5674	ndcplus.cfg	Outgoing Console	ATM Cassette	Cash Dispenser		
NCR5684	NCR 5684	NCR	5684	ndcplus.cfg	Outgoing Console	ATM Cassette	Cash Dispenser		
NCR5840	NCR 5840 (Pers)	NCR	5840	ndcplus.cfg	Outgoing Console	ATM Cassette	Cash Dispenser		MPD:10100100000
NCR5870	NCR 5870 (Pers)	NCR	5870	ndcplus.cfg	Outgoing Console	ATM Cassette	Cash Dispenser		MPD:10100100000
NCR5874	NCR 5874 (Pers)	NCR	5874	ndcplus.cfg	Outgoing Console	ATM Cassette	Cash Dispenser		MPD:10100100000

Fig. 1. Form containing ATM types registered in WAY4

The form contains the following fields:

- *Code* – the code identifying the ATM type.
- *Name* – a description of the ATM type.
- *Brand* – brand name of the ATM manufacturer, for example "NCR", or "WINCOR".
- *Model* – the ATM model.
- *Configuration File* – the name of the configuration file with its file path relative to the WEB-INF/inv/atm/conf directory (the bank must

independently create the inv/atm/conf directory in the WAY4 Transaction Switch root directory).

- *Outgoing Console* – an ATM component that assists in ATM management.
- *Cassette Account Code, Cash Account Code, Remaining Account Code* – these fields define the names of account types from an ATM contract's Account Scheme for accounts that show the activity of funds when replenishing the ATM (see "Settlement Scheme for ATM Operations").
- *EMV Conf File* – device parameters; set in a "tag;value" list through semicolons; for example, the MPD and VPD tags can be used to set attributes for outgoing messages (device type, card reading method, etc.) generated by WAY4 Transaction Switch interface services for sending to MasterCard or Visa, respectively.
- *Transaction Attributes* – additional transaction parameters.
- *Protocol Id* – protocol name (see the section "ATM Protocols Dictionary").

To add a record in this form, click [Ins]; to delete a record, click [Del].

When deleting records from the "ATM Types" form that correspond to the type of ATM for which the contract device is registered in WAY4 (see the section "Creating New Device Contracts" in the document "Acquiring Module User Manual"), a warning will appear on the screen.

The [Dflt Oper] button makes it possible to set a list of default operations for the selected ATM type. The list is generated on the basis of the "ATM Operations" dictionary (see "ATM Operations Dictionary").

The [Dflt Hardware] button makes it possible to set a list of default components for the selected ATM type. Components from the "ATM Hardware Types" dictionary can be selected (see "ATM Hardware Types Dictionary").

ATM Denominations Dictionary

Each ATM in the system is assigned a denomination type that defines the type of cassette that can be used by the ATM, the face value and currency of notes/coins stored in the cassette during device configuration, as well as some parameters for cash withdrawal.

The ATM denomination type can be set choosing the appropriate user menu items when configuring the device (see the section "Configuring Devices" from the document "Acquiring Module User Manual").

The ATM denominations dictionary (see Fig. 2) can be accessed by selecting the following: "Full → Configuration Setup → Merchant Device Setup → ATM Denominations".

Fig. 2. Denominations dictionary

To add a record in this form, click the [Ins] button; to delete a record, click [Del].

When attempting to delete a record from the "ATM Denominations" form that corresponds to the type of denomination used for the ATM registered in WAY4 (see the section "Creating New Device Contracts" in the document "Acquiring Module User Manual"), a warning will be displayed.

Every denomination type corresponds to certain parameters that define the way the cassette is loaded with cash/coins and the algorithm for dispensing cash. To access the form that contains the following parameters, select the desired denomination type in the "ATM Denominations" form and click the [Denom] button. The "Denom for <denomination type>" form will be displayed (see Fig. 3).

Direction	Item Type	Name	Mapped to	Currency	Denomination	Limit	Dispense Parameters	Cycle Type Code
Debit	Banknote	1		EUR	500,00	40		
Debit	Banknote	2		EUR	10,00	40		
Debit	Banknote	3		EUR	100,00	40		
Debit	Banknote	4		EUR	50,00	40		

Fig. 3. Parameters for loading a cassette with notes


The form contains the following fields:

- *Direction* – cassette use according to the direction of funds ("Debit" – dispense, "Credit" – accept).
- *Item Type* – cassette type ("Banknote" or "Coin").
- *Name* – cassette name, depending on the ATM type, the numbers "1", "2", "3", "4", "5", "6", "7" are used.
- *Mapped to* – link to the name of a cassette for cash acceptance (for example, for rejected notes from the current cassette).
- *Currency* – list for selecting the abbreviated name of the currency that is loaded in the cassette.
- *Denom* – denomination of notes/coins loaded in the cassette.
- *Limit* – the maximum number of notes/coins (from 0 to 999) that can be dispensed from a cassette during one operation.
- *Dispense Parameters* – rule for compiling a set of notes/coins for dispensing; set with the "PENALTY" and "PENALTY_INCR" tags (separated by a comma), for example:

PENALTY=1/4/3,PENALTY_INCR=0/3/1

Where:

- PENALTY is the penalty for dispensing notes/coins from the cassette; 1/4/3 is the value of the penalty for the first, second, and third algorithms, respectively, for compiling a set of notes/coins.
- PENALTY_INCR – increases the penalty for each subsequent note/coin dispensed from the cassette; 0/3/1 is the value of the increase in the penalty for the first, second and third algorithm for compiling a set of notes/coins.

 The algorithm for compiling notes/coins depending on the menu item selected at the ATM is determined in the controller configuration (see "Request processing configuration file") using the dispenseType parameter.

Various combinations of notes are selected from the ATM's cassette when dispensing cash in such a way as to minimize the valuation function (the penalty) for dispensing some combinations of notes while considering the dispensation limit.

The value of the valuation function P is calculated according to the following formula:

$$P = \sum_{i=1}^N (n_i \cdot CST_FEE_i + \frac{n_i \cdot (n_i - 1)}{2} \cdot CST_FEE_INCREASE_i),$$

where:

- N – the number of cassettes from which notes are selected.
- n_i – the number of notes selected from the i -th cassette.
- CST_FEE_i – the penalty for dispensing notes from the i -th cassette (value of the PENALTY tag for the selected algorithm).
- $CST_FEE_INCREASE_i$ – the increase in the penalty for each subsequent note dispensed from the i -th cassette (value of the PENALTY_INCR tag for the selected algorithm).

Fixed Dictionaries

ATM Protocols Dictionary

ATM protocols regulate the message format and rules for information exchange between ATMs and the processing center.

The ATM protocols dictionary is contained in the form "ATM Protocols" (see Fig. 4).

Name	Code
Diebold 912	MDS912
NDC+	NDC+

Fig. 4. Types of protocols for connecting ATMs to the processing center

The form contains the following fields:

- *Name* – the name of the protocol
- *Code* – the protocol code indicated within the system



The current implementation of the ATM controller on the WAY4 Transaction Switch platform supports the "NDC+" protocol..

ATM Operations Dictionary

Every ATM contract in the system is registered in accordance with an aggregate of operations that can be accomplished by the given device, as well as a set of hardware components necessary for those operations (see "ATM Hardware Types Dictionary").

The ATM operations dictionary is contained in the form "ATM Operations" (see Fig. 5).

Name	Code	Transaction Type	Trans Cond	Request	Category	Service Class	Check	Downgradable	To Requires	Cassette	Is Online	Use Coin	Special P
Bill Payments by Cash	T-Y1	Pay by Cash		Advice	Authorisation	Transaction			No	No	No		P1
CE: Conversion	T-X1_D	CE: 5. Conversion		Advice	Authorisation	Transaction			No	No	No		
CE: Rounding	T-X1_C	CE: 4. Rounding		Advice	Authorisation	Transaction			No	No	No		
CE: Cash Dispense (Che)	T-X1_B	CE: 3. Cash Dispense		Advice	Authorisation	Transaction			Yes	No	Yes		
CE: Cash Dispense	T-X1_A	CE: 2. Cash Dispense		Advice	Authorisation	Transaction			Yes	No	Yes		
CE: Currency Exchange	T-X1	CE: Currency Exchar		Advice	Authorisation	Transaction			No	No	No		OPER
PWC: Rounding	T-W1_C	PWC: 3. Rounding		Advice	Authorisation	Transaction			No	No	No		
PWC: Cash Dispense (CT)	T-W1_B	PWC: 2. Cash Dispense		Advice	Authorisation	Transaction			Yes	No	Yes		

Fig. 5. List of ATM operations

The form contains the following fields:

- *Name* – the name of the operation.
- *Code* – the operation code.
- *Transaction Type* – the type of transaction.
- *Trans Cond* – transaction conditions.
- *Request* – category of document generated for a message about an operation (request/notification).
- *Category* – financial/authorisation message category.
- *Service Class* – the bank's classification of transactions; the value of this parameter determines the way documents are handled by the system. By default, in the absence of other values, the value of the parameter *Service Class* will be "Transaction".

- *Is Checked* – field with list of value options that indicate whether it is necessary to subject the service card to control actions (value "Yes") when fulfilling service operations (Replenishment, End of Day, ATM Service and others).
- *Downgradable To* – field indicating the name of the operation designated to be executed in the event that the given operation cannot be executed.
- *Requires Cassettes* – flag indicating the use of cassettes when performing an operation.
- *Is Online* – indicates if an operation is performed online (whether a request to the issuer is made when performing the operation).
- *Use Coins* – flag indicating whether coins can be used in performing the operation.
- *Special Parameters* – additional parameters for an operation; set in a comma-delimited "tag=value" list.

The list of ATM operations is contained in the section "ATM Operations " in the document "ATM Network Monitoring".

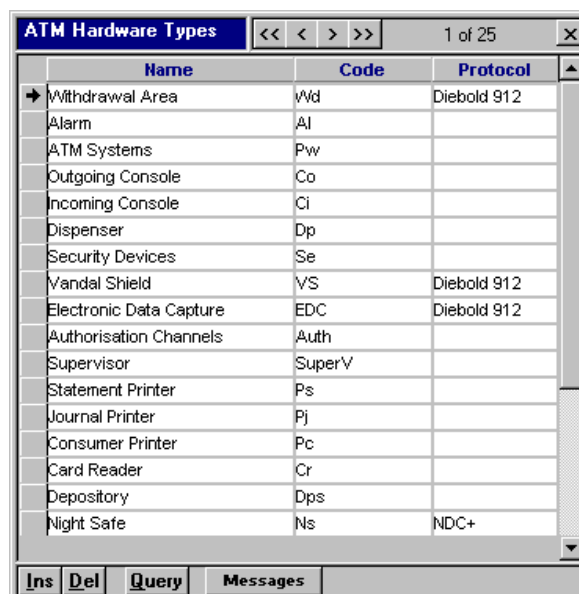
The [Required] button is used to call up the form "Required for <name of operation>", which contains a list of ATM components (see "ATM Hardware Types Dictionary") necessary for fulfilling a given operation.

When the [Fill Default] button is clicked, the "ATM Operations" form is filled with default values.

ATM Hardware Types Dictionary

Every ATM contract in the system is registered in accordance with certain hardware components necessary for the ATM to fulfill its operations (see "ATM Operations Dictionary").

The ATM Hardware Types dictionary is contained in the form "ATM Hardware Types" (see Fig. 6).



Name	Code	Protocol
Withdrawal Area	Wld	Diebold 912
Alarm	Al	
ATM Systems	Pw	
Outgoing Console	Co	
Incoming Console	Ci	
Dispenser	Dp	
Security Devices	Se	
Vandal Shield	VS	Diebold 912
Electronic Data Capture	EDC	Diebold 912
Authorisation Channels	Auth	
Supervisor	SuperV	
Statement Printer	Ps	
Journal Printer	Pj	
Consumer Printer	Pc	
Card Reader	Cr	
Depository	Dps	
Night Safe	Ns	NDC+

Fig. 6. Form showing ATM hardware components

The form contains the following fields:

- *Name* – the name of the component;
- *Code* – the code of the component in the system;
- *Protocol* – the type of protocol allowing for the use of the given component (this field is left blank when using a component that is compatible with all registered protocol types).

The [Messages] button is used to pull up the form "Messages for <name of component>", which contains a list of messages formed by the system when working with the given component (see "ATM Message Types Dictionary").

The list of ATM components may be found in the paragraph "ATM Hardware " in the document "ATM Network Monitoring".

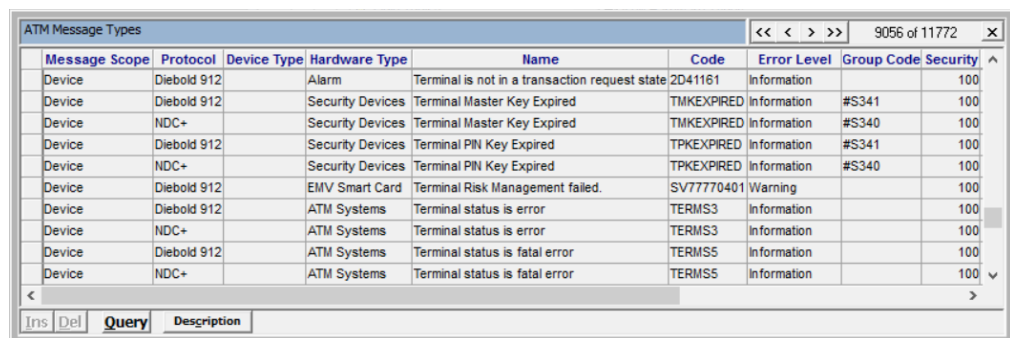
ATM Message Types Dictionary

During the ATM's operation, certain status messages may be generated in WAY4.

The WAY4 host may send management messages (commands) to the controller, ATM or ATM group. For example, through the management console (see the section "Changing ATM Status and Managing ATMs " in the document "ATM Monitoring") it is possible to install an ATM configuration or load controller configuration files.

ATMs, in turn, send messages about the results of executing host commands, and messages with information about the state of their devices. As a result, several messages are recorded for the corresponding ATM in the WAY4 database; each message belonging to a certain ATM component.

The dictionary of possible ATM message types is contained in the form "ATM Message Types" (see Fig. 7).



Message Scope	Protocol	Device Type	Hardware Type	Name	Code	Error Level	Group Code	Security
Device	Diebold 912		Alarm	Terminal is not in a transaction request state	2D41161	Information		100
Device	Diebold 912		Security Devices	Terminal Master Key Expired	TMKEXPIRED	Information	#S341	100
Device	NDC+		Security Devices	Terminal Master Key Expired	TMKEXPIRED	Information	#S340	100
Device	Diebold 912		Security Devices	Terminal PIN Key Expired	TPKEXPIRED	Information	#S341	100
Device	NDC+		Security Devices	Terminal PIN Key Expired	TPKEXPIRED	Information	#S340	100
Device	Diebold 912		EMV Smart Card	Terminal Risk Management failed.	SV77770401	Warning		100
Device	Diebold 912		ATM Systems	Terminal status is error	TERMS3	Information		100
Device	NDC+		ATM Systems	Terminal status is error	TERMS3	Information		100
Device	Diebold 912		ATM Systems	Terminal status is fatal error	TERMS5	Information		100
Device	NDC+		ATM Systems	Terminal status is fatal error	TERMS5	Information		100

Fig. 7. Messages generated during ATM operation

The form contains the following fields:

- *Message Scope* – message recipient/source:
 - "Device" – ATM.
 - "Device Group" – ATM group.
 - "Physical Channel" – controller.
- *Protocol* – the name of the protocol (see the section "ATM Protocols Dictionary").

- *Device Type* – ATM types registered in WAY4, this field is used to provide message details according to the device type; for example, a message with various values in the field *Error Level* may appear for different kinds of ATMs when fulfilling the same operation (with the same value in the *Code* field).
- *Hardware Type* – the name of the ATM hardware component (see "ATM Hardware Types Dictionary") that was running when the message was generated; only specified for *Message Scope* = "Device".
- *Name* – a description of the message.
- *Code* – the message code.
- *Error Level* – the error level to which the ATM hardware component is relegated upon receiving the given message; depending on the error level, the current status of the component and ATM may change as follows:
 - "OK" (code "0") – the component status is "OK"; the ATM status is "OK" if no errors have occurred other components; the ATM status is "Information" if there are no errors in other components, but there was an error in this component; the ATM status is "Information", "Warning", or "Error" depending on the status of other components.
 - "Information" (code "1") – component and ATM status does not change.
 - "Warning" (code "2") – component status is "Warning"; ATM status is "Information".
 - "Error" (code "3") – components status is "Error", ATM status is "Warning".
 - "Not Configured" (code "4") – component status is "Not Configured"; ATM status is "Information".
 - "Unavailable" (code "6") – components status is "Unavailable", ATM status is "Information".
 - "Fatal Error" (code "5") – component status is "Fatal Error"; ATM status is "Error".
- *Group Code* – field used to enter instructions to the ATM that are executed when the given message is received (instruction format: #<operation code> or a simplified name); for example:
 - DI01=1113 – set cassette status, where DI01 is the dispenser code according to the "ATM Hardware Types" dictionary, 1, 1, 1, 3 are cassette status codes (see the codes in the description of the *Error Level* field).
 - #S340 – send keys to the ATM (where S340 is the code of the corresponding operation in the "ATM Operations" dictionary).
 - STARTUP – execute a sequence of commands; take out of service, request configuration and counter information, put into service.
- *Security* – the access level granted to the processing center's operator for administrative control of the ATM controller. The access level for user groups can be queried through the *Security Level* field in the form

"Constants for <name of group>", which can be invoked by clicking on the [Constants] button in the form "User Groups and Users - View" (Full → DB Administrator Utilities → Users & Grants → User Groups and Users – View).

The [Description] button invokes the form "Description for <message description>", which contains further details on the message.

Chapter 2. Description of a New ATM and its Configuration

Setting up an ATM Contract and its Device

A description of how a new ATM contract is entered may be found in the section paragraph "Creating New Device Contracts" in the document "Acquiring Module User Manual".

Setup of devices for ATM contracts is described in the section "Configuring Devices" of the document "Acquiring Module User Manual").

Setting up the Executable Range of ATM Operations

The executable range of ATM operations is set up through the form "Operations for <name of ATM>" (see Fig. 8).

The form can be invoked in two ways:

- After selecting from the user menu "Acquiring → ATM Controller → ATM Device Management", select the desired ATM and click on the [Operations] button in the form "ATM Device Management".
- After selecting from the user menu "Acquiring → Acquiring Contracts", select the desired account contract and click the [Devices] button in the account contract form, then select the desired ATM. Then, click on the [ATM] button in the device contract and click [Operations] in the device configuration form.

Operations for MERCHANT 5				<< < > >>	1 of 38	b	x
	Operation Type	Status	Hardware Problem	Last Changed			
➔	Cash Dispense with ticket	Active		27/05/02 14:26.38	<div></div>		
	Balance Inquire	Active		27/05/02 14:26.38			
	Cash Dispense without ticket	Active		27/05/02 14:26.38			
	Master key Change	Active		22/03/02 13:24.43			
	COMM key under Master Change	Active		22/03/02 13:24.43			
	COMM key under COMM key Change	Active		22/03/02 13:24.43			
	MAC key under Master key Change	Active		22/03/02 13:24.43			
	MAC key under COMM key Change	Active		22/03/02 13:24.43			
	Set B-key as current COMM key	Active		22/03/02 13:24.43			
	Set B-key as current MAC key	Active		22/03/02 13:24.43			
	Send new Configuration Data	Active		22/03/02 13:24.25			
	Send new Screens/Keyboard Data	Active		22/03/02 13:24.25			
	Send new State Tables	Active		22/03/02 13:24.43			
					<div></div>		
Ins	Del	Query	Ch Status	History			

Fig. 8. List of executable ATM operations

An operation can be deleted from the list by selecting the desired row in the form and clicking on the [Del] button.

The execution of any given operation can be suppressed by changing its status after clicking on the [Ch Status] button. Clicking on this button will change the status of executable operations from the value "Active" to value "Closed".

The *Last Changed* field contains the date and time of the last change to the operation's status.

To restore the list of allowed operations (according to the "ATM Operations" dictionary) after rows have been deleted from the table, click the [Setup] button and choose the [Check and Fill] context menu item in the "ATM Device Management" form or the "ATM for <device identifier>" device configuration form.

Configuring ATM Hardware Components

ATM components can be configured through the form "Hardware for <name of ATM>" (see Fig. 9).

This form can be invoked in two ways:

- After selecting from the user menu "Acquiring → ATM Controller → ATM Device Management", select the desired ATM and click on the [Hardware] button in the form "ATM Device Management".
- After selecting from the user menu "Acquiring → Acquiring Contracts", select the desired account contract and click the [Devices] button in the account contract form, then select the desired ATM. Then, click on the [ATM] button in the device contract and click [Hardware] in the device configuration form.

The screenshot shows a window titled "Hardware for 00000007". It contains a table with the following columns: Hardware Type, Status, Amend Date, Last Error Type, and Last Error Details. The table lists various components like Digital Camera System, Envelope Dispenser, Coin Dispenser, Envelope Depository, ATM Controller, Currency Cassettes, Display, Contactless Smart Card Read, Passbook Printer, Statement Printer (highlighted with a mouse cursor), and Sensors. The Status column shows "OK" for most, "Error" for the Statement Printer, and "Warning" for Sensors. The Amend Date for all is 27/01/15 11:41:57, except for the Statement Printer at 11:45:41 and Sensors at 11:46:55. The Last Error Type for the Statement Printer is "Statement Printer Error (U)".


Hardware Type	Status	Amend Date	Last Error Type	Last Error Details
Digital Camera System	OK	27/01/15 11:41:57		
Envelope Dispenser	OK	27/01/15 11:41:57		
Coin Dispenser	OK	27/01/15 11:41:57		
Envelope Depository	OK	27/01/15 11:41:57		
ATM Controller	OK	27/01/15 11:41:57		
Currency Cassettes	OK	27/01/15 11:41:57		
Display	OK	27/01/15 11:41:57		
Contactless Smart Card Read	OK	27/01/15 11:41:57		
Passbook Printer	OK	27/01/15 11:41:57		
Statement Printer	Error	27/01/15 11:45:41	Statement Printer Error (U)	
Sensors	Warning	27/01/15 11:46:55		

At the bottom of the window, there are buttons for "Ins", "Del", "Query", "Console", and "Messages".

Fig. 9. Form for configuration of ATM components

The list of ATM components available for configuration is generated as follows:

- Based on the list of default components for this ATM type (see "ATM Hardware Types Dictionary").
- If default components are not specified for this ATM type, the list is generated on the basis of the entire "ATM Hardware Types" dictionary (see "ATM Hardware Types Dictionary").

 In this case components that are missing in the ATM must be disabled by setting their status to "Not Configured" (the procedure for changing components status is described below).

This list is filled in with records relevant for the current configuration by selecting the [Check and Fill] context menu item of the [Setup] button in the "ATM Device Management" form or the "ATM for <device ID>" form.

The *Amend Date* field of the "Hardware for <ATM name>" grid form contains the date and time the component's status was last changed. In the event of an error, the *Last Error Type* and *Last Error Details* fields contain the error type and text, respectively.

The ATM component is ready to function properly if the status of the component is set to "OK". A component can also function in the "Warning" status, but this state indicates that problems in its operation are possible.

To change the status of a component, select the desired row in the form and click on the [Console] button. By this command the screen will display the form "Console for <name of component>" (see Fig. 10). Select in the *Command* field of the form the desired control command, for example, "<name of component> OK", and click on the [Run] button.

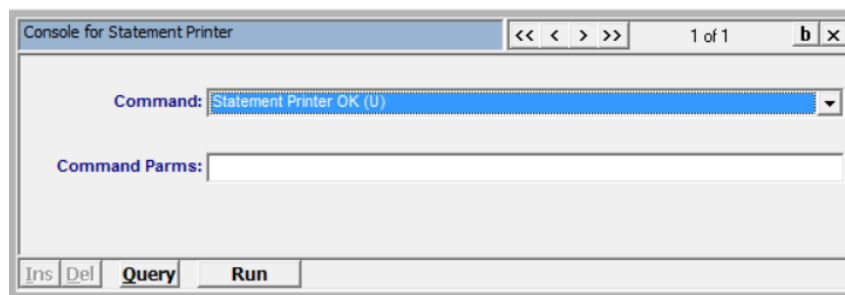


Fig. 10. Form for entering ATM component management commands

The "OK" status can be set for all components at once by selecting the [Set to OK] context menu item of the [Setup] button in the "ATM Device Management" form or the "ATM for <device ID>" form.

To deactivate a component selected in the "Hardware for <ATM name>" form, select the management command "<component name> Not Configured" in the *Command* field of the "Console for <component name>" form (see Fig. 10) and click the [Run] button. For more information about ATM management commands, see the section "ATM Management Commands" in the document "ATM Monitoring".

Information on the history of messages related to this component can be found in the form "Messages for <name of component>" (see Fig. 11). The form is invoked from the table "Hardware for <name of ATM>" after choosing the desired row and clicking on the [Messages] button.

Record ID	Message Time	Message Text	Message Type	Error Level	Code	Status	Hardware
3773	27/01/15 11:45:41		Statement Printer Error (U)	Error	1VE003	Posted	Statement Printer
3771	27/01/15 11:44:54		Statement Printer OK (U)	OK	1VE0	Posted	Statement Printer
3769	27/01/15 11:44:18		Statement Printer Fatal Error (U)	Fatal Error	1VE004	Posted	Statement Printer
1796	18/11/14 14:43:31		Statement Ribbon - Not Configured	Not Configured	2F1S220	Posted	Statement Printer
1795	18/11/14 14:43:31		Statement Paper - Not Configured	Not Configured	2F1S210	Posted	Statement Printer
1773	18/11/14 14:43:31		Statement printer - Routine error	Information	2F1F211	Posted	Statement Printer
1749	18/11/14 14:43:30		Statement Ribbon - Not Configured	Not Configured	2F1S220	Posted	Statement Printer
1748	18/11/14 14:43:30		Statement Paper - Not Configured	Not Configured	2F1S210	Posted	Statement Printer
1726	18/11/14 14:43:30		Statement printer - No error	OK	2F1F210	Posted	Statement Printer

Fig. 11. Form containing system messages generated during changes in ATM component status

Specifying Encryption Keys

Encryption keys are created by a security officer with the help of encryption equipment and include a fixed number of digits.

Encryption keys are only stored in the system and in the ATM in a state where each key is encrypted by other encryption keys. A check value is used for controlling the key's accuracy. This value is defined only by the value of the key and does not depend on how it is encrypted.

Specification of encryption keys is accomplished through the form "Keys for <name of ATM>" (see Fig. 12).

The form can be invoked on the screen in two ways:

- After selecting from the user menu "Acquiring → ATM Controller → ATM Device Management", select the desired ATM and click on the [Keys] button in the form "ATM Device Management";
- After selecting from the user menu "Acquiring → Acquiring Contracts", select the desired account contract and click the [Devices] button in the account contract form, then select the desired ATM. Then, click on the [Keys] button in the device contract.

Key Algorithm	Key Type	Key Name	DES Key	Key Check	Used as MK	Storage MK	Serial Number	Is Active	Date From	Date To	Max Usage
3DES ABA	Terminal PIN Key	Terminal PIN Key	UD387E839870E8FA6A7223F					Active	18/11/14 00:00:00	0	0
3DES ABA	Terminal Authentication	Terminal Authentication						Inactive	20/11/14 00:00:00	0	0

Fig. 12. Form for the specification of ATM encryption keys

The form contains the following fields for entering encryption key values:

- *Key Algorithm* – a selection from a list in order to indicate an encryption algorithm the key will be used for.
- *Key Type* – the type of encryption key indicated through selecting from a list formed on the basis of the "PM Key Types" system dictionary.
- *Key Name* – the name of an encryption key.
- *DES Key* – the field for entering the value of an encryption key encrypted with the Local Master Key of the HSM (Host Security Module).
- *Key Check* – the check value of an encryption key.

- *Used as MK* – this field determines whether the key will be used as the master key.
- *Storage MK* – drop-down list to select the master key used to encrypt this key when transmitting it to the terminal; the list consists of keys with the "Yes" value in the *Used as MK* field.
- *Serial Number* – the ID of a key determining its value among other keys of the same type.
- *Is Active* – this field indicates an encryption key's availability; possible values:
 - "Active" – active key (used in encryption).
 - "Inactive" – inactive key (not used in encryption).
 - "Locked" – key that has been locked because the number of attempts to use it incorrectly was exceeded (not used in encryption).
- *Date From* – the field for entering the initial date of the period of time, within which the key in question remains available for use.
- *Date To* – the field for entering the final date of the period of time, within which the key in question remains available for use.
- *Max Usage* – the field for entering a number determining how many times the encryption key in question may be used.
- *Max Wrong Attempts* – number of attempts to incorrectly use the key after which the key is locked.
- *Wrong Attempts Threshold* – number of wrong attempts to incorrectly use the key after which an alarm goes off.
- *Current Usage* – the field containing the value specifying how many times this encryption key was used.
- *Wrong Attempts* – counter of attempts to incorrectly use the key.
- *Storage Form* – form for storing the key in the database.
- *Key Code* – *Key Type* value shown in the form specified in the *Storage Form* field.
- *Parent Key* – parent key.
- *Add Data* – additional data.

The [Manage] button of the "Keys for <ATM name>" form opens the "DES Management Mode" form used for generating keys.

The [Key Options] button in the "Keys for <ATM name>" form opens the "Key Options for Terminal PIN Key" form, used to manage additional key parameters.

To add a record to this form, click the [Ins] button; to delete a record, click [Delete]

Enabling MAC-signature Mode

To enable MAC (Message Authentication Code) mode, set the value "Mandatory" in the field *Mac Status* in the "ATM Device Management" form or "ATM for <ATM name>" form. If the value in this field is set to "None", this mode is not enabled.

Additional Device Parameters

The list of device configuration parameters can be expanded by specifying additional parameters in the "Enh Parms for <ATM name>" form (see Fig. 13). This form is opened by clicking the [Enh Parms] button in the "ATM for <ATM name>" form or "ATM Device Management" form.

Parameter	Value
Debug Level	ON
EJ Acknowledgement Timer	30
EJ Upload Block Size	350
EJ Retry Threshold	2

Fig. 13. Additional device parameters

The form contains the following fields:

- *Parameter* – name of an available parameter.
- *Value* – parameter value.

The [Send Task] button is not used for ATM parameters.

Configuring the ATM Connection with the WAY4 Host

An ATM interacts with the WAY4 host via TCP connections transmitting messages in both directions. Each TCP connection is unique and is uniquely identified by a pair of sockets (a set of four elements defining the two end points of the connection: local IP address of the terminal, local TCP port, remote IP address of the host and remote TCP port) in a network.

To ensure the TCP connection is unique, the ATM must use dynamically allocated ports; that is, ports with a short lifecycle.

To get up-to-date information about the state of a connection, a keep-alive timer must be set up.

An ATM establishes a connection with the controller based on the IP address and port number defined in the device's software settings. The port must be specified as the one listened on in controller settings (see "Controller Configuration File"). When a connection is established, the controller checks the WAY4 database for a device with an IP address corresponding to this ATM (see the section "Configuring Devices" in the document "Acquiring Module. User Manual").

To work with the ATM, the controller generates a special communication service code consisting WAY4 Transaction Switch node ID in which the corresponding service operates (the "node" parameter of the node.properties

configuration file) and the name of the service with which the connection is established. After connecting, this code is saved in the WAY4 database and is shown in the *Online Service* field of the form with the device's state ("Acquiring → ATM Controller → ATM Device Management", [State] button).

If connection problems occur, the ATM can be dynamically switched between WAY4 Transaction Switch nodes (for example, between main and backup). In this case, a new communication service code is set for the ATM and recorded in the WAY4 database for the corresponding device.

The following figure Fig. 14 shows the scheme for connecting ATMs to WAY4 Transaction Switch over leased lines using the TCP/IP protocol.

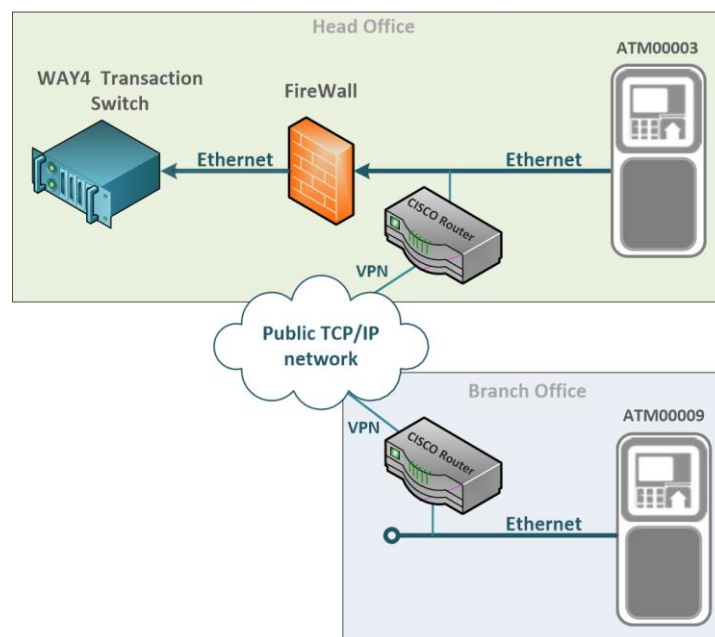


Fig. 14. ATMs connected to WAY4 Transaction Switch on leased lines using the TCP/IP protocol.

Preparing and Loading ATM Configurations

The procedure for preparing and loading ATM configuration files is described below.

Preparing the Configuration Files

ATM configuration files may contain the following information:

- States – components of the configuration file that define the sequence of operations executed by the ATM depending on user actions, in the event of equipment or service failure or other events.
- Screens – components of the configuration file that define the external appearance of the ATM screen at each possible step of an operation being executed (the screen menu, messages to the user, and others).
- FIT form – components of the configuration file that are used to determine the affiliation of the bank card to a certain payment system.

- Configuration parameters, various additional utilities, the ATM's logical number, the values of different time intervals and delays
- Extended configuration parameters, various additional utilities, the ATM's logical number, the values of different time intervals and delays
- Configuration data for reserve screen templates, used during operations
- Configuration data for smart card operations
- Script commands to the ATM controller:
 - #S331 – change the ATM's master key, the new key is sent to the ATM encrypted under the current master key;
 - #S332 – change the ATM's PIN-key, the new key is sent to the ATM encrypted under the current master key;
 - #S333 – change the ATM's PIN-key, the new key is sent to the ATM encrypted under the current PIN-key;
 - #S335 – change the ATM's MAC-key, the new key is sent to the ATM encrypted under the current master key;
 - #S336 – change the ATM's MAC-key, the new key is sent to the ATM encrypted under the current PIN-key;
 - #S340 – change the ATM's MAC-key and PIN-key, the new keys are sent to the ATM encrypted under the current master key;
 - #S341 – change the ATM's MAC-key and PIN-key and sets up the current value of the Configuration ID, the new keys are sent to the ATM encrypted under the current master key;
 - #S334 – setting up the ATM's PIN-key from the cell containing a key, directly placed in the ATM by security officers;
 - #S337 – setting up the ATM's MAC-key from the cell containing a key, directly placed in the ATM by security officers;
 - #S322 – operation for setting the time and date in the ATM;

The configuration file may contain several variants of ATM configuration, which are distinguished from one another by their configuration ID.

The WAY4 Transaction Switch directory containing configuration files for each type of ATM as well as the file names themselves must correspond to the values in the *Configuration File* field in the ATM Types dictionary (see "ATM Types Dictionary").

Presented below is a fragment of the configuration file:

100002001	c f J G T E M P O R A R I L Y O F F - L I N E
210000205	J 0 5 5 0 0 0 0 5 5 0 5 5 0 0 0 0 0 0 0 0 0 0 0 0
300000	0 0 0 0 0 0 0 0 0 3 2 0 3 0 0 0 1 1 6 0 0 0 _ 0 0 0 _ 0 0 6 0 7 5 0 7 0 0 2 0 8 0 7 5 0 9 0 7 2
510000000	0 0 0 1 0 3 0 9 8 0 0 4 2 5 5 2 5 5 0 0 0 0 0 0 1 3 2 0 0 0 0 0 0 0 0 0 0 0 3 1 1 3 8 0 0
@10000001	C 0 4 Z Z Z Z 5 Z Z 9 6 9 9
A00000	0 0 0 _ 0 0 6 0 7 5 0 7 0 0 2 0 8 0 7 5 0 9 0 7 2

```

B10000      00000000000000000000000000000000
G00000D0010 034
*00000#S332

```

ATM configuration files have the following format:

Position	Value	Parameter
1	1	ATM screen template
	2	ATM state
	3	Configuration parameters, value of timing delays
	5	Table of financial institutions
	A	Additional parameters
	B	Parameter defining for which message fields MAC (Message Authentication Code) will be used
	G	Parameter defining the rewriting of system templates for ATM screens
	*	Instructions to the ATM controller
2	1/0	Flag determining whether MAC (Message Authentication Code) is used for the corresponding line in the configuration file; when "1", MAC is used, when "0", MAC is not used
3-6	Numerals from 0000 to 9999	Configuration ID
7-...	Characters according to device specifications	Data on states, screen templates, financial institutions, etc.

Sending a Configuration to an ATM

An ATM configuration is updated through the management console by executing the "Send New Configuration to ATM" management command in the ATm controller (see the section "Commands for Loading Configuration Data" in the document "ATM Monitoring").

Sending a Configuration to all ATMs

The configuration of all ATMs registered in WAY4 can be updated through the ATM group management console using the command "Send new Configuration to GROUP" (see the section "Commands for Loading Configuration Data" in the document "ATM Monitoring"). The configuration of a specific ATM will be loaded according to the Configuration ID value specified in the *Configuration* field for the corresponding ATM (see the section "Configuring Devices" in the document "Acquiring Module User Manual").

Chapter 3. Cash Dispensation

To enable cash dispensing functionality, the following settings are required.

Cash Dispensation: Configuring the ATM

In order to dispense cash, the ATM must support the following options:

- Entering of the PIN code of the cardholder;
- Selection of the language in which the ATM screen will present information; the given selection can be done automatically by the ATM, for example, depending on the card number;
- Filtration by financial institutions;
- Selection of operations;
- Selection of currency for the operation (for multi-currency ATMs);
- Selection of account type by the cardholder (for holders of cards issued by other banks);
- Entering the amount of the operation;
- Request to print a receipt;
- Reaction to response from the processing center (for example, regarding the absence of funds in the account of the cardholder).

Upon executing the operation, the ATM transmits the following information to the processing center:

- Bank card number;
- Account type selected by the cardholder (for holders of cards issued by other banks)
- Amount of the operation;
- Currency of the operation;
- Language, in which the receipt from the executed operation is printed.

Cash Dispensation: Configuring the Receipt

The receipt indicating the result of the executed operation can contain fields that correspond to the data received from the processing center:

- Name and address of the bank;
- ATM name;
- Date and time of the operation;
- Bank card number;
- Unique operation identification number;
- Authorization code;

- Amount and currency of the operation;
- Amount and currency of the acquiring fee (if such is determined);
- Balance on the card account (when present).

The indicated fields should appear in the receipt template (see "Configuring receipt and screen templates").

Chapter 4. ATM Cash Acceptance

The WAY4 system facilitates cash acceptance operations through ATM cash acceptor devices, such as the Bunch Note Acceptor and the Cash Acceptor. This functionality is provided as a separate WAY4 module and is made available through special permissions from OpenWay. While executing this operation, the cash acceptor device checks the authenticity of notes as well as note currency and denomination.

During this operation, the cardholder indicates the currency of the deposit. The cash acceptor device checks the authenticity of the notes, as well as their currency and denomination. The ATM sends an authorization request to the processing center while indicating the type of operation and the sum and currency of the deposit (or information that allows those parameters to be determined in WAY4 Transaction Switch). The system implements the standard procedure for bankcard verification, and checks whether the given operation is authorized for the cardholder. If one of these produces a negative return, the system returns an operation refusal to the ATM. If the necessary parameters are successfully met, the amount, registered by the cash acceptor device upon processing by an authorization document, will be added to the amount of available funds in the card account. The actual account replenishment, or in other words, transfer of monetary funds, is accomplished through processing the financial document of that operation.

ATM Cash Acceptance: Configuring the ATM

To replenish accounts, the ATM configuration should support the following:

- Entering of the PIN code of the cardholder;
- Selection of the language in which the ATM screen will present information; the given selection can be done automatically by the ATM, for example, depending on the card number;
- Filtration by financial institutions;
- Selection of operations;
- Selection of account type by the cardholder;
- Selection of currency for the operation (for multi-currency ATMs);
- Entering the amount of the operation;
- Request to print a receipt;
- Reaction to response from the processing center (for example, that the given operation is not authorized for the cardholder)

Upon executing the operation, the ATM transmits the following information to the processing center:

- Bank card number;
- Account type selected by the cardholder;

- Amount and currency of the operation or information that allows those parameters to be determined in WAY4 Transaction Switch;
- Language, in which the receipt from the executed operation is printed.

ATM Cash Acceptance: Configuring the Receipt

The receipt indicating the result of the executed operation can contain fields that correspond to the data received from the processing center:

- Name and address of the bank;
- ATM name;
- Date and time of the operation;
- Bank card number;
- Unique operation identification number;
- Authorization code;
- Amount and currency of the operation;
- Amount and currency of the acquiring fee (if such is determined);
- Amount of accepted notes presented by denomination with breaks in between (this information can be printed in a receipt by configuring the receipt template, when the necessary information is available).

The indicated fields should appear in the receipt template (see "Configuring receipt and screen templates").

Chapter 5. Receiving Balances and Mini-statements

The chapter describes settings for getting balance information and mini-statements for a client's card account.

Configuring Balances on Card Accounts

The rules for giving out balances according to card accounts are set up when working with the issuing module of the WAY4 system, and are also regulated by the ATM receipt format template.

Restrictions on the number of free balance inquiries of card accounts, as well as the fee factor for exceeding that number, may be specified by configuring the Service Package of the card account (see the document "WAY4™ Service Packages".)

The ATM receipt template file (see "Configuring receipt and screen templates") may be configured to block balance inquiries on card accounts with card numbers within a certain range or those issued by a bank with a certain bank identification number.

Balance inquiries, including balances on card accounts, can be blocked with changing the status of the corresponding operation or the ATM component (see the sections "ATM Operations" in the document "ATM Monitoring").

The receipt for balance inquiries on card accounts can contain fields that correspond to the data received from the processing center:

- Bank name and address;
- ATM number;
- Date and time of the operation;
- Abbreviated bank card number;
- Amount of available funds;
- Credit limit.

The above-mentioned fields should be indicated in the receipt template (see "Configuring receipt and screen templates").

Configuring Mini-statements on Card Accounts

The rules for giving out mini-statements according to card accounts are similar to the rules governing balances and are set up when working with the issuing module of the WAY4 system and regulated by the ATM receipt format template.

Restrictions on the number of free mini-statements on card accounts, as well as the fee factor for exceeding that number, may be queried by configuring the

Service Package of the card account (see WAY4™ Service Packages Administrator Manual).

The ATM receipt template file (see "Configuring receipt and screen templates") may be configured to block requests for mini-statements on card accounts with card numbers within a certain range or those issued by a bank with a certain bank identification number.

Blocks on mini-statements, including mini-statements on card accounts, can be configured by changing the status of the corresponding operation or the ATM component (see the sections "ATM Operations" and "ATM Hardware" in the document "ATM Monitoring").

The mini-statement on card accounts can contain fields that correspond to the data received from the processing center:

- Bank name and address;
- ATM number;
- Date and time of the operation;
- Bank card number;
- Amount of available funds.

The above-mentioned fields should be indicated in the receipt template (see "Configuring receipt and screen templates").

Configuring the Screen Display

The cardholder can be presented with a choice of where the balance request results should be displayed: printed in a receipt, or displayed on the ATM screen.

To do this, create the corresponding templates and links to them in the response message configuration file (see "Configuring receipt and screen templates").

Chapter 6. Changing the PIN Code

WAY4 allows cardholders to change their card PIN code at ATMs. This functionality is not part of the standard setup for WAY4 and is provided through special agreement with OpenWay representatives.

Changing the PIN Code: Configuring the ATM

To change the PIN code, the ATM should support the following:

- Selection of the language in which the ATM screen will present information; this can be done automatically by the ATM, for example, depending on the card number;
- Entering of the old PIN code by the cardholder;
- Filtration by financial institutions;
- Selection of operations;
- Entering of the new PIN code by the cardholder;
- Confirming of the new PIN code by the cardholder by entering it twice;
- Reaction to response from the processing center (for example, that the given operation is not authorized for the cardholder).
- When the PIN code change operation is completed, the ATM transmits the following data to the processing center:
 - Bank card number;
 - Language in which the receipt for the given operation will be printed;
 - Old PIN-block, encrypted under the ATM's PIN-key;
 - New PIN-block, encrypted under the ATM's PIN-key.

Changing the PIN Code: Configuring the Receipt Format for PIN Code Changes

The receipt given out for PIN code changes can contain fields that correspond to the following data, received from the processing center:

- Bank name and address;
- ATM number;
- Date and time of the operation;
- Bank card number;
- Operation's unique identification number.

The above-mentioned fields should be indicated in the receipt template (see "Configuring receipt and screen templates").

Chapter 7. Automatic Reversal Message Creation

The following figure Fig. 15 shows the exchange of messages when an operation is executed.

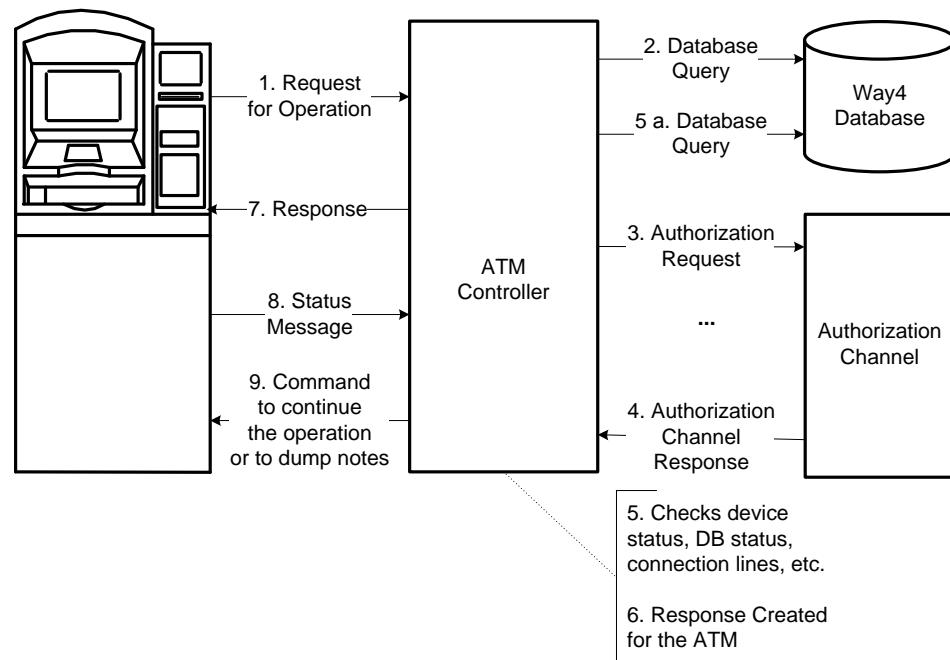


Fig. 15. Message exchange while executing ATM operations

The ATM controller automatically creates and transmits through the authorization channel reversal messages in the following four situations:

In stage 4 of the operation (see Fig. 15): if there is no response from the authorization channel within a specified time (50 sec).

In stage 5 of the operation (see Fig. 15) if one of the of the following conditions is found:

- Connection to the ATM was lost;
- Device status has changed;
- DB status has changed;
- A command is received to immediately remove the device from service;
- Another request for another operation is received from the ATM.
- In stage 6 of the operation (see Fig. 15): if data is insufficient, absent, or corrupted in field #39 of standard ISO message 8583, also if it is not possible to create a MAC message, for example, because of no connection with the hardware security module (HSM).

In stage 8 of the operation (see Fig. 15): if a corresponding status message is received, for example, that the amount selected has not been dispensed to the cardholder.

Chapter 8. Administrative Operations and Replenishment

Replenishment is a set of operations to re-supply the ATM with the cash funds intended for dispense, to collect funds deposited by clients to replenish their accounts, to reconcile the information stored in the ATM and held in the processing center on the results of ATM operations, and to fulfill other functions.

The replenishment procedure contains the following steps:

- Printing of receipts for the cash replenishment officer according to the processing center and the state of the ATM's counters;
- Removal of the cassette from the ATM for its contents to be inventoried at the bank;
- Loading new cassettes into the ATM;
- Removal of retracted and retained cards;
- Closing of the financial cycle in the database (see the section "Closing Financial Cycles" in the document "ATM Monitoring").

During the replenishment process, the cash replenishment officer may enter data on the quantity of loaded and unloaded notes.

ATM replenishment is accomplished through the use of a replenishment officer service card (see chapter "Issuing Service Cards " in the document "Acquiring Module User Manual").

Financial Cycles

Financial cycles are intervals of time between the ATM's replenishment. The system allows for observing the balance for the current financial cycle and past financial cycles, as the difference between the quantity of notes loaded into and dispensed from the ATM (see the section "Financial Cycles" in the document "ATM Monitoring").

Closing the current financial cycle in the database and opening the next one is done automatically after the ATM is replenished and the replenishment officer enters the completed operation in the processing center by a special transaction or manually.

Configuring the ATM Replenishment Receipt Format

The receipt received by the replenishment officer, detailing the results of ATM replenishment, can contain fields corresponding to the following data received from the processing center:

- Bank name and address;

- ATM number;
- Date and time of operation;
- Service card number;
- Unique identification number for the operation;
- Amount of dispensed funds from each cassette;
- Amount of dispensed funds according to currency (for multi-currency ATMs);
- Amount of notes loaded, dispensed, dispensed but ignored by the cardholder and retracted by the ATM, and diverted by ATM during dispense;
- Number of the financial cycle.

The above-mentioned fields should be indicated in the receipt template (see "Configuring receipt and screen templates").

Configuring the Replenishment Officer's Receipt for Collecting Client-Deposited Funds

The replenishment officer's receipt can contain fields corresponding to data received from the processing center:



- Bank name and address;
- ATM number;
- Date and time of operation;
- Replenishment officer's service card number;
- Unique operation ID;
- Amount of deposited funds in each currency;
- Amount of deposited notes in each denomination;
- Financial cycle number.

The fields should be configured in the receipt template (see "Configuring receipt and screen templates").

Chapter 9. Settlement Scheme for ATM Operations

During an ATM's operation (dispensing/accepting cash, replenishment) accounting entries are generated in WAY4 that reflect fund activity.

An ATM contract's Accounting Scheme determines the types of accounts between which entries are made. The acquiring module contains the standard Accounting Scheme "001-Default ATM Scheme" (see acquiring Product Accounting Schemes in the menu item "Full → Configuration Setup → Products → Acquiring Products → Acquiring Account Schemes") which establishes the relation between the following types of account:

- "ATM Cassette" – type of account that shows fund activity:
 - In cassettes issued to replenishment officers from the bank till for loading into the ATM.
 - In cassettes taken from the ATM by replenishment officers, to be given to the bank till.
- "Cash Dispenser" – type of account that shows fund activity in the ATM in the period between loading and unloading.
 -  If ATM cash acceptance and dispensing operations are supported, two separate types of account must be used in the Accounting Scheme; for example, "Cash Dispenser In" and "Cash Dispenser Out" to show funds that have been accepted and funds that are available for dispensing, respectively.
- "Merchant Receivable" – type of account that shows the amount of funds for operations made by clients at the ATM during the business day.
 -  If ATM cash acceptance and dispensing operations are supported, two separate types of account must be used in the Accounting Scheme; for example, "Merchant Receivable In" and "Merchant Receivable Out" to show funds that have been accepted from and dispensed to clients during the business day.

Due normalization between "Cash Dispenser" and "Merchant Receivable" account types is set up that determines whether the accumulated amount of dispensed/accepted funds must be shown in the "Cash Dispenser" account when opening a new business day.

Entries showing the movement of funds between accounts in the process of ATM operation and service are generated in two ways:

- Manually – when posting financial documents created by the operator (for example, through the "Doc – General" form for working with documents).
- Automatically – when posting financial documents created as a result of:
 - Due normalization set up in the ATM contract's Accounting Scheme.
 - Replenishment (loading/unloading the ATM).

The settlement scheme for ATM operations is shown below. This scheme is based on acquiring module standard settings and includes the following entries:

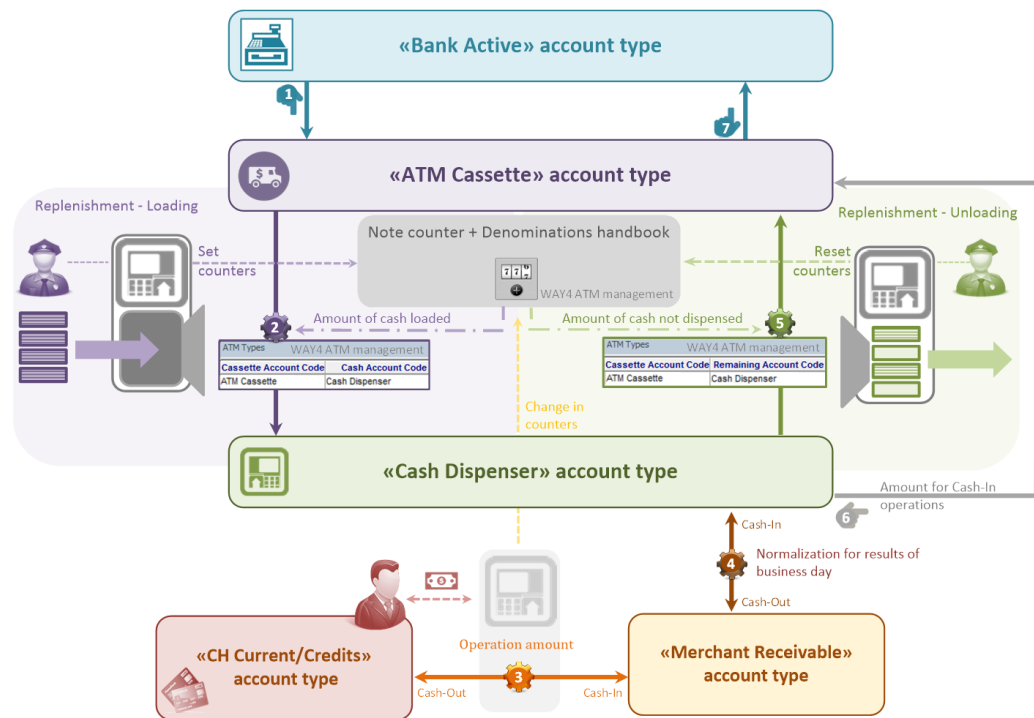


Fig. 16. Settlement scheme for ATM operations

1. The operator manually creates a document for an "ATM cash replenishment" transaction (see the transaction type dictionary in the menu item "Full → Configuration Setup → Transaction Types → Transactions - All"). When this document is posted, an entry is generated between accounts of the following types:

- Dt "ATM Cassette" Ct "Bank Active"

For the amount of funds issued to replenishment officers to be loaded into the ATM.

i Here and further – the document posting process can be run several times a day and is not linked to the process for opening the banking day.

2. After loading cassettes into the ATM, the replenishment officer uses a service card and performs the ATM_SERVICE operation. Information about the number of notes loaded in each cassette is entered in the ATM menu. This data is used to set cassette note counters in WAY4.

For each cassette, a separate document is generated whose amount is determined based on the specified number of notes and the denominations dictionary (see "ATM Denominations Dictionary"). When posting these documents, entries between accounts are generated. Account types are set in the *Cassette Account Code* and *Cash Account Code* fields for the corresponding ATM type (see "ATM Types Dictionary"):

- Dt "Cash Dispenser" Ct "ATM Cassette"

3. If a debit/credit operation is successful, a financial document is created in the WAY4 database whose posting results in the following entries:

- Dt "CH Current/Credits" Ct "Merchant Receivable" – for cash dispensing settlement schemes.

- Dt "Merchant Receivable" Ct "CH Current/Credits" – for cash acceptance settlement schemes.



The account type "CH Current/Credits" used for settlements with the bank's "own" clients was chosen as an example.

4. The procedure for opening a new banking day performs due normalization for the amount of operations made during the previous business day.
 - Dt "Merchant Receivable" Ct "Cash Dispenser" – for cash dispensing settlement schemes.
 - Dt "Cash Dispenser" Ct "Merchant Receivable" – for cash acceptance settlement schemes.
5. After removing cassettes from the ATM, the replenishment officer uses a service card and performs the REPLENISHMENT operation. As a result of this operation, for each cassette a separate document is generated in WAY4 for the amount of funds not dispensed (determined on the basis of current values for note counters and the denominations dictionary in the WAY4 database). When posting these documents, entries between accounts are generated. Account types are set in the *Cassette Account Code* and *Remaining Account Code* fields for the corresponding ATM type (see "ATM Types Dictionary").
 - Dt "ATM Cassette" Ct "Cash Dispenser"
6. For cash acceptance settlement schemes, the operator "manually" creates financial documents for cassettes that replenishment officers removed with accepted notes. When posting these documents, entries between accounts are created:
 - Dt "ATM Cassette" Ct "Cash Dispenser"
7. When funds removed by replenishment officers in unloading the ATM are received, the operator "manually" creates a document for an "ATM residual cash collection" transaction (see the dictionary of transaction types in the menu item "Full → Configuration Setup → Products → Acquiring Products → Acquiring Account Schemes"). When this document is posted, an entry is created between the following types of account:
 - Dt "Bank Active" Ct "ATM Cassette"

If discrepancies are found, adjustments (for the excess/insufficient amount) are manually generated by the operator.

Chapter 10. ATM Controller Configuration Files

ATM controller setup includes:

- Configuration of the ATMController service on the WAY4 Transaction Switch platform.
- Preparation of scripts for processing requests and response messages for the controller's interaction with ATMs.
- Preparation of templates for information output in receipts and on the ATM screen.

For the ATM controller to operate correctly, the following WAY4 Transaction Switch services may also have to be configured:

- AcqDBService – service for interacting with the WAY4 acquiring module (AcqDB.s.xml configuration file).
- AuthService – WAY4 authorisation service (AuthService.s.xml configuration file).
- HSMThalesAdapter – service for interacting with the encryption device (HSMAdapter.s.xml configuration file).
- IntranetAdapter – service for interacting with WAY4 intranet infrastructure software components (Intranet.s.xml configuration file).
- IssDBService – service for interacting with the WAY4 issuing module (IssDB.s.xml configuration file).
- AppRoutingService – service for routing messages between WAY4 Transaction Switch services (Routing.s.xml configuration file).

Controller Configuration File

The ATM controller configuration file is located in the WEB-INF/conf/application directory relative to the WAY4 Transaction Switch working directory and by default is named ATMController.s.xml.

Main parameters:

- port – determines the number of the free port listened on for establishing a TCP connection with the ATM.
- tcp_keepAlive – determines the need (true/false) to enable a keep-alive mechanism.
- recTimeoutSec – determines the maximum time interval in seconds to get a message in a TCP connection.
- sendTimeoutSec – determines the maximum time interval in seconds to send a message in a TCP connection.

Sample ATMController.s.xml configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<service xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  singleton="true"
  autoStarted="true"
>

  <appComponent xsi:type="ATMController">
    <transport
      xsi:type="ExternalServer"
      port="5262"
      tcp_keepAlive="true"
      recTimeoutSec="5"
      sendTimeoutSec="50">
      <filter
        xsi:type="BinLen"
        headerLength="2"/>
      </transport>
    </appComponent>
    <dependency service="AcqDB"/>
    <dependency service="HSMAdapter"/>
  </service>

```

⚠ The configuration file should only be changed under the supervision of OpenWay representatives.

Configuration Files for Controller Interaction with the ATM

Setup of controller interaction with ATMs (see Fig. 17) includes:

- Defining rules for processing requests from ATMs, for their further conversion to ISO messages and transmission to WAY4 Transaction Switch interface services (with bank, payment system or payment acceptance system) (see "Request processing configuration file").
- Defining rules for processing responses received in ISO messages from interface channels to generate the corresponding commands to the ATM (see "Response processing configuration file").

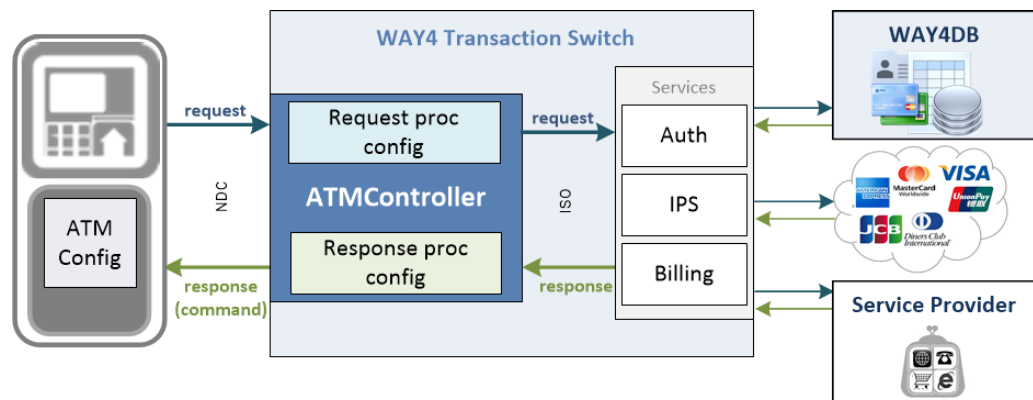


Fig. 17. ATM interaction with the controller

These rules are defined in the corresponding configuration files implemented in Java Groovy. If it is necessary to edit groovy scripts or to add new ones, this language's syntax must be observed (see <http://www.groovy-lang.org/>).

Principles of working with configuration scripts

All groovy scripts used in a configuration must be placed in the WEB-INF/inv/atm/scripts/ directory. The controller constantly monitors the content of groovy scripts, applying the changed configuration "on the fly", excluding the need to restart the service.

Each groovy script can be called repeatedly during one transaction. Data accumulated in the transaction's context are saved between script calls, making it possible to implement complex logic for processing transactions and to use these data in receipt and screen templates (through the "params" structure, see "Configuring receipt and screen templates").

A special structure must be imported to each groovy script:

```
com.openwaygroup.ts.atm.controller.handler.algorithms.transaction.TransactionContext
```

This structure stores all parameters for the current transaction and a special object – a link to processes that can be executed in the script in the context of the transaction. Each script, resulting from its operation, must return this structure.

The following processes can be run through the object-link:

- startOperation – initiates execution of a specified type of operation in WAY4 (generation of an acquiring document); no parameters.
- cassettesReport – initiates generation of a cash dispensing cassette report in WAY4; no parameters.
- bnaReport – initiates generation of a cash acceptance cassette report; no parameters.
- closeCycle – initiates closing an ATM financial cycle in WAY4; there may be no parameters or a value like:

```
com.openwaygroup.ts.fp.shared.atm.replenishment.ReplenishmentCycleType
```

- logOperationName – logs the name of the current operation to the WAY4 database; no parameters.
- setInReplenishment – transfers a device to replenishment status in the WAY4 database; no parameters.
- setPostProcess – sets the scenario used after the transaction; the following type of value is used as a parameter:

```
com.openwaygroup.ts.fp.shared.atm.scenario.ScenarioType
```

Processes in the body of groovy scripts are called as follows:

```
proc.startOperation.exec() // process without parameters
proc.closeCycle.exec { // process with parameters
    ReplenishmentCycleType.CASH_IN
}
```

Request processing configuration file

Rules for processing ATM requests are defined in the RequestMapper.groovy configuration script. This file's name can be redefined in the ATMController service's configuration.

Configuration script example:

```
package com.openwaygroup.ts.atm.controller.handler.scripts // script
package

//Import of objects used in the script.
import
com.openwaygroup.ts.atm.controller.handler.algorithms.transaction.TransactionContext
import com.openwaygroup.ts.fp.shared.Currency
import
com.openwaygroup.ts.atm.controller.handler.scripts.model.ScreenReference
import com.openwaygroup.ts.fp.shared.OperStatus
import com.openwaygroup.ts.fp.shared.atm.command.KeyPosition
import com.openwaygroup.ts.fp.shared.atm.command.KeyStatus
import com.openwaygroup.ts.fp.shared.atm.scripts.AccountType

//Static import.
import static com.openwaygroup.ts.fp.shared.atm.scripts.OperationType.*
import static
com.openwaygroup.ts.fp.shared.atm.scripts.DispenseAlgorithmType.*

logger.debug("executing RequestMapper v 3.0")
//Get context from the enviroment (the next string is not mandatory but
//this declaration is convenient for automatic addition to IDE)
def ctx = (TransactionContext) binding.ctx
//Definition of the Key Buffer value from Transaction Request
def keys = ctx.getMessage().getOperationKeyBuffer();
//Account type definition -
com.openwaygroup.ts.fp.shared.atm.scripts.AccountType
ctx.accountFrom = AccountType.DEFAULT
//Operation currency definition
ctx.currency = Currency.byName("RUR")
//Language definition (language - any string identifier that can also
//be used in receipt and screen templates). For example, in the template
//described below, this is the language: <#if lang == "RU">
ctx.language = (keys =~ /.....A/) ? "RU" : "EN"
//By default, receipt printing is allowed
ctx.printReceipt = true
//Operation type definition
```

```
switch (keys) {
    case ~/C...../: //This groovy construction makes it posible to
                        //match strings with a template - regular
                        //expression
                        //Operation type definition (enum
                        //com.openwaygroup.ts.fp.shared.atm.scripts.OperationType )
                        ctx.operation = CASH_WITHDRAWAL
                        //Definition of the cash dispensing calculation type (enum
                        //com.openwaygroup.ts.fp.shared.atm.scripts.DispenseAlgorithmType)
                        ctx.dispenseType = PENALTY_BASED
                        //Definition of the receipt printing flag
                        ctx.printReceipt = (keys =~ /..A...../)
                        break
    case ~/BAAAB.../:
                        //Operation type definition( enum
                        //com.openwaygroup.ts.fp.shared.atm.scripts.OperationType )
                        ctx.operation = MINI_STATEMENT
                        break
    case ~/BAAAA.../:
                        //Operation type definition( enum
com.openwaygroup.ts.fp.shared.atm.scripts.OperationType )
                        ctx.operation = BALANCE_INQ
                        //Account type definition -
                        //com.openwaygroup.ts.fp.shared.atm.scripts.AccountType
                        ctx.accountFrom = AccountType.SAVINGS
                        //Set the tag to call start operation
                        ctx.setDocTag("LANG", "ru")
                        break
    case ~/A...../:
                        //Operation type definition ( enum
                        //com.openwaygroup.ts.fp.shared.atm.scripts.OperationType )
                        ctx.operation = NOTE_ACCEPTANCE
                        ctx.printReceipt = !(keys =~ /..C...../)
                        break
    case ~/D...../:
                        //Operation type definition ( enum
                        //com.openwaygroup.ts.fp.shared.atm.scripts.OperationType )
                        ctx.operation = PIN_CHANGE
                        //Set new PIN value ( from buffer b )
                        // ctx.setNewPin(ctx.getMessage().getBufferB())
                        //или из customer selected pin buffer
```



```
        ctx.setNewPin(ctx.getMessage().getCspData())
        break
    case ~/I.AA..../:
        //Operation type definition ( enum
        //com.openwaygroup.ts.fp.shared.atm.scripts.OperationType )
        ctx.operation = REPLENISHMENT
        break
    case ~/I.AB..../:
        //Operation type definition ( enum
        //com.openwaygroup.ts.fp.shared.atm.scripts.OperationType )
        ctx.operation = END_OF_DAY
        break
    case ~/I.AC..../:
        //Operation type definition ( enum
        //com.openwaygroup.ts.fp.shared.atm.scripts.OperationType )
        ctx.operation = COLLECTION
        break
    default: // UNDEFINED - does not have to be defined (this operation
type is set
        //by default)
        ctx.operation = _UNDEFINED
        break
}
//Use of an object-link to the process
//for calling startOperation.
proc.startOperation.exec()

if(ctx.getOperation() in [REPLENISHMENT, COLLECTION, END_OF_DAY]){
    //If necessary, the operation name is logged.
    proc.logOperationName.exec()
}
//For cash acceptance, the block is determined for the interactive
session with
//the customer.
if(ctx.operation == NOTE_ACCEPTANCE && ctx.getRc().isSuccessful())
{
    //Customer timeout is 120 seconds
    ctx.setConsumerTimeout(120)
    //Determines the need to execute an asynchronous interactive
    //session with the ATM after the full (!) completion of this script.
    ctx.interactiveTransactionRequest()
    //to do so, this closure will be performed before the session.
```

```

//IMPORTANT! In the current implementation, the closure is performed in
a context
//separate from the main script.
//Use of variables from an external (free) context
//STRICTLY PROHIBITED!!!!
//This behaviour may change in future versions.
ctx.beforeProcess { TransactionContext context, def proc ->
    //clear screen buffer in transaction context
    context.clearScreens()
    //Generation of a block of active keys for sending to the ATM
    ctx.setActiveKey(KeyPosition.FDK_A_AND_ENTER,
KeyStatus.ACTIVATE)
    ctx.setActiveKey(KeyPosition.FDK_D, KeyStatus.ACTIVATE)
    ctx.setActiveKey(KeyPosition.CANCEL, KeyStatus.ACTIVATE)
    //Add a screen from the template note_acceptance_scr.ftl
context.addScreen(ScreenReference.screenReference("note_acceptance_scr"))
    //Return context
    context
}
//with subsequent closure after getting a response from the ATM.
ctx.afterProcess { TransactionContext context, def proc ->
    //Analysis of received bufferB.
    def bufferB = context.getMessage().getBufferB()
    //If the cancel button was pressed on the screen ("D")
    // cancel на pin-pad ("E")
    //or the timeout expired ("T"),
    if(!bufferB || (bufferB.substring(0, 1) in ["D", "E", "T"] ))
    {
        //the operation is declined with the code CUSTOMER_CANCELLATION
(17)
        context.setRc(OperStatus.CUSTOMER_CANCELLATION)
    }
    context
}
}
//Return the object TransactionContext
Ctx

```

This example uses a link to a screen template (scr.ftl). For more information about setting up screen and receipt templates, see the section "Configuring receipt and screen templates".

Since the name of the script that must be used to process the results of the operations listed is not defined in the example shown, the RCMapper.groovy script will be used (see "Response processing configuration file").

Response processing configuration file

Rules for processing responses (operation results) are defined in the RCMapper.groovy configuration script. This is the default name and can be redefined through the ctx.rcMapperName field in the request processing configuration script (see "Request processing configuration file"). To process operation results, several scripts can be used describing the procedure for performing additional actions, generating receipts, screens, etc.

Configuration script example:

```
package com.openwaygroup.ts.atm.controller.handler.scripts

import
com.openwaygroup.ts.atm.controller.handler.algorithms.transaction.TransactionContext
import
com.openwaygroup.ts.atm.controller.handler.algorithms.transaction.TransactionPhase
import com.openwaygroup.ts.fp.shared.atm.command.PrintType
import
com.openwaygroup.ts.fp.shared.atm.replenishment.ReplenishmentCycleType
import com.openwaygroup.ts.fp.shared.atm.scenario.ScenarioType

import static
com.openwaygroup.ts.atm.controller.handler.scripts.model.PrintReference.printReference
import static
com.openwaygroup.ts.atm.controller.handler.scripts.model.ScreenReference.screen
import static com.openwaygroup.ts.fp.shared.atm.scripts.OperationType.*

logger.debug("rc script ver. 2.1")

def ctx = (TransactionContext) binding.ctx
//Get RC (response code is stored as an object
//com.openwaygroup.ts.fp.shared.OperStatus, but here it's more convenient
to get
//standard numeric)
def rcCode = ctx.getRc().getResponseCode()

//Define default state
ctx.nextStateId = "396" // default next state id
//Clear screen buffer
ctx.clearScreens()
```

```
//Clear receipt buffer
ctx.clearPrints()

switch(rcCode)
{
    case 0:
        if (ctx.operation == CASH_WITHDRAWAL) {
            ctx.nextStateId = "701"
            //Add screens to the list for generation
            //Transaction Reply
            ctx.addScreen (screen("070"))
            ctx.addScreen (screen("071"))
            ctx.addScreen (screen("072"))
            //A list of receipts is generated depending on the flag
            if (ctx.printReceipt) {
                ctx.addPrint (printReference(PrintType.RECEIPT,
"consumer1")) //Add a receipt with the RECEIPT type and consumer1.ftl
template
                ctx.addPrint (printReference(PrintType.JOURNAL,
"journal1")) //Add a receipt with the JOURNAL type and journal1.ftl
template
            } else {
                ctx.addPrint (printReference(PrintType.JOURNAL,
"journal1"))
            }
        }

        if (ctx.operation == BALANCE_INQ) {
            ctx.nextStateId = "701"
            ctx.addScreen(screen("072"))
            ctx.addPrint (printReference(PrintType.RECEIPT, "consumer1"))
            ctx.addPrint (printReference(PrintType.JOURNAL, "journal1"))
        }

        if(ctx.operation == MINI_STATEMENT)
        {
            ctx.nextStateId = "701"
            ctx.addScreen(screen("072"))
            ctx.addPrint( printReference(PrintType.RECEIPT, "statement1"))
            ctx.addPrint( printReference(PrintType.JOURNAL, "journal1"))
        }

        if(ctx.operation == NOTE_ACCEPTANCE)
```

```

    {
        ctx.nextStateId = "701"
        ctx.addScreen(screen("072"))
        ctx.addPrint( printReference(PrintType.RECEIPT,
"note_acceptancel"))
        ctx.addPrint( printReference(PrintType.JOURNAL, "journal1"))
    }

    if(ctx.operation == REPLENISHMENT)
    {
        //Use an object-link to processes
        //to execute cassettesReport in the WAY4 DB
        proc.cassettesReport.exec()
        ctx.nextStateId = "701"
        ctx.addScreen(screen("072"))
        ctx.addPrint (printReference(PrintType.JOURNAL_RECEIPT,
"admin1"))
        //If APPROVE, a script is set that will be
        //executed after the transaction has been completed
        //the script can be specified as a closure or lambda, or
        //as a link with the name of the script
        if(ctx.getPhase() == TransactionPhase.APPROVE)
        {
            //in this case, it is expected that the script directory
will
            //contain the groovy script Replenishment.groovy
            ctx.postScriptWith ("Replenishment")
        }
    }
    if(ctx.operation == COLLECTION)
    {
        proc.bnaReport.exec()
        ctx.nextStateId = "701"
        ctx.addScreen(screen("072"))
        ctx.addPrint(printReference(PrintType.JOURNAL_RECEIPT,
"collection1"))
        if(ctx.getPhase() == TransactionPhase.APPROVE)
        {
            //Use an object-link to processes
            //for closing an ATM financial cycle in the WAY4 DB
            //the cycle type is specified in the the closure (by
default:

```

```
//com.openwaygroup.ts.fp.shared.atm.replenishment.ReplenishmentCycleType.CA
SH_OUT )

        proc.closeCycle.exec {
            ReplenishmentCycleType.CASH_IN
        }
        //Use an object-link to processes
        //for setting the scenario ScenarioType.RETRIEVE_NOTE_DEFS
as
        //the post process
        proc.setPostProcess.exec {
            ScenarioType.RETRIEVE_NOTE_DEFS
        }
    }
}
break

case 1:
case 2:
case 5:
    ctx.nextStateId = "203"
    break
case 3:
case 15:
    ctx.nextStateId = "202"
    break
case 4:
case 7:
    ctx.nextStateId = "205"
    ctx.addPrint(printReference(PrintType.JOURNAL, "journal1"))
    ctx.retainCard = true
    break
case 13:
    ctx.nextStateId = "209"
    break
case 14:
    ctx.nextStateId = "214"
    break
default:
    ctx.nextStateId = "396"
    break
}
```

```
ctx
```

This example uses links to receipt templates (consumer1.ftl, journal1.ftl). For more information about setting up screen and receipt templates, see the section "Configuring receipt and screen templates".

In the example, execution of the Replenishment.groovy script is planned if a REPLENISHMENT transaction is completed successfully. Replenishment.groovy may have, for example, the following content:

```
Package com.openwaygroup.ts.atm.controller.handler.scripts

import
com.openwaygroup.ts.atm.controller.handler.algorithms.transaction.TransactionContext
import com.openwaygroup.ts.fp.shared.atm.scenario.ScenarioType

logger.debug("replenishment script part 1. version 1.0")

def ctx= (TransactionContext) binding.ctx

//Planned asynchronous process for sending commands to ATM with subsequent
execution of the ReplenishmentAfter.groovy script
ctx.sendAtmCommand().withCommand(ScenarioType.OP_OUT_OF_SERVICE).afterProcess("ReplenishmentAfter")

ctx
```

ReplenishmentAfter.groovy sample content:

```
package com.openwaygroup.ts.atm.controller.handler.scripts

import
com.openwaygroup.ts.atm.controller.handler.algorithms.transaction.TransactionContext
import
com.openwaygroup.ts.fp.shared.atm.replenishment.ReplenishmentCycleType

logger.debug("replenishment script part 2. version 1.0")
def ctx = (TransactionContext) binding.ctx

//Use of an object-link to processes
//for closing an ATM financial cycle in the WAY4 DB.
proc.closeCycle.exec()
//Use of an object-link to processes
// for closing a different type of financial cycle in the WAY4 DB.
proc.closeCycle.exec {
    ReplenishmentCycleType.COINS_OUT
}
```

```
//Use of an object-link to the process
//setting device replenishment status in the WAY4 DB.
proc.setInReplenishment.exec()
ctx
```

Configuring receipt and screen templates

Templates for information output in receipts and to an ATM screen are generated using the Java FreeMarker template engine (for more information, see <http://freemarker.org/>). The jEdit editor can be used to work with templates.

The following ASCII codes can be used in templates:

Code	ASCII value	Code	ASCII value	Code	ASCII value
ACK	Acknowledgment	EM	End of Medium	LF	Line Feed
BS	Back Space	ETX	End of Text	NAK	Negative Acknowledgement
BEL	Bell	EOT	End of Transmission	NUL	Null char
CAN	Cancel	ETB	End of Transmit Block	RS	Record Separator
CR	Carriage Return	ENQ	Enquiry	SI	Shift In / X-Off
DLE	Data Line Escape	ESC	Escape	SO	Shift Out / X-On
DC1	Device Control 1 (oft. XON)	FS	File Separator	SOH	Start of Heading
DC2	Device Control 2	FF	Form Feed	STX	Start of Text
DC3	Device Control 3 (oft. XOFF)	GS	Group Separator	SUB	Substitute
DC4	Device Control 4	HT	Horizontal Tab	SYN	Synchronous Idle
US	Unit Separator	VT	Vertical Tab		

The following fields may be used in templates:

Name	Type	Description
date	java.util.Date	Current date and time.
time	java.util.Date	Current date and time.
rrn	java.lang.String	RRN
stan	java.lang.String	STAN
authCode	java.lang.String	Authorisation code.
lang	java.lang.String	Transaction language.
operation	com.openwaygroup.ts.fp.shared.atm.scripts.OperationType	Operation type.
rc	java.lang.String	Response code
lun0	java.lang.String	Terminal identifier.
terminalId	java.lang.String	Terminal identifier.
card	java.lang.String	PAN
applicationId	java.lang.String	EMV AID
applicationLabel	java.lang.String	EMV Application label
cardName	java.lang.String	Bankcard type.
amount	com.openwaygroup.ts.fp.shared.Amount	Cash withdrawal amount and currency.

This information is confidential and may be used exclusively to run OpenWay programs. It may not be duplicated, published or disclosed without written permission of OpenWay.

Name	Type	Description
request	com.openwaygroup.ts.fp.shared.Amount	Amount and currency requested by the cardholder.
fee	com.openwaygroup.ts.fp.shared.Amount	Acquirer fee amount and currency.
amounts	java.util.List<com.openwaygroup.ts.fp.shared.ExtendedAmount>	List of accounts, amounts and currencies.
statement	java.util.List<com.openwaygroup.ts.fp.shared.MinistatementRecord>	List with ministatement records.
dispense	java.util.List<com.openwaygroup.ts.atm.controller.handler.dispense.DispenseValue>	Number of issued notes, by cassette.
atm	java.lang.String	ATM type.
vendor	java.lang.String	ATM type.
location	com.openwaygroup.ts.fp.shared.Location	Structure describing ATM address data.
info	java.util.Map<java.lang.String, java.lang.String>	Associative array with tags exported from the WAY4 DB.
params	java.util.Map<java.lang.String, java.lang.Object>	Associative array with parameters defined in groovy scripts (see "Configuration Files for Controller Interaction with the ATM").
cardsRetained	java.lang.Long	Number of cards retained.
cassettes	java.util.List<com.openwaygroup.ts.fp.shared.atm.device.Cassette>	List of cassettes (denomination types, counters).
cassetteAmounts	java.util.List<com.openwaygroup.ts.fp.shared.Amount>	Cassette amounts.

Dot notation must be used to contact a structure field or associative array element. For example, the currency code for any Amount object can be obtained as follows:

```

${amount.currency.name} - for a three-letter currency code
${amount.currency.code} - for a numeric ISO code

```

Example of receipt format:

```

<#setting number_format="0.00" />
<#setting datetime_format="dd.MM.yy HH:mm" />
<#if vendor == "NCR" || vendor == "NCR3G">
    <#if lang = "EN">
    ---- ${ESC}(2BANK -----
    Tel.(322) 32-23-34

    Atm number: ${terminalId}
    Date:      Time:  Card number:
    ${time?datetime}  ${card[0...*6]}..${card[card?length - 1..]}

```

```

    <#if operation == "CASH_WITHDRAWAL" && (amount.amount)??>

Dispense amount: ${amount.amount?left_pad(12)} <#if
(amount.currency)??>${amount.currency.name}</#if>

    </#if>
    <#if amounts??>

        <#list amounts as amt>
            <#switch amt.amountType>
                <#case "01">

Ledger:          ${amt.amount?left_pad(12)} ${amt.currency.name}
                    <#break>

                <#case "02">

Available:       ${amt.amount?left_pad(12)} ${amt.currency.name}
                    <#break>

                <#case "91">

Credit limit:   ${amt.amount?left_pad(12)} ${amt.currency.name}
                    <#break>

                <#default>
            </#switch>
        </#list>
    </#if>

${authCode}/${rrn}
${FF}
</#if>
<#if lang == "RU">
---- ${ESC}(2BANK -----
Tel.(322) 32-23-34

Банкомат: ${lun0}
Дата:      Время:      карта:
${time?datetime}  ${card[0...*6]}..${card[card?length - 1..]}

${ESC}(2
    <#if operation == "CASH_WITHDRAWAL" && (amount.amount)??>

```

```

ВЫДАНА СУММА: ${amount.amount?left_pad(12)} <#if
(amount.currency)?>${amount.currency.name}</#if>

</#if>

<#if amounts??>

    <#list amounts as amt>
        <#switch amt.amountType>
            <#case "01">
текущий    ${ESC}(2: ${amt.amount?left_pad(12)} ${amt.currency.name}
            <#break>

            <#case "02">
доступный  ${ESC}(2: ${amt.amount?left_pad(12)} ${amt.currency.name}
            <#break>

            <#case "91">
кредитный  ${ESC}(2: ${amt.amount?left_pad(12)} ${amt.currency.name}
            <#break>

            <#default>
            </#switch>
        </#list>
    </#if>

${ESC}(2${authCode}/${rrn}
${FF}
</#if>
</#if>

```

Prepared .ftl template files must be put in the WEB-INF/inv/atm/templates directory. Links to these files can be used in configuration scripts for processing operation requests and results (see "Configuration Files for Controller Interaction with the ATM").

The controller constantly monitors the content of the /templates directory, applying changes "on the fly", excluding the need to restart the service.

For screens and receipts to be generated correctly based on created templates in the WEB-INF/inv/atm/templates directory, it is necessary to enable the Encoding.groovy script determining the table for re-encoding depending on the languages used. This script is only loaded when starting the ATMController. Encoding.groovy sample content:

```

package com.openwaygroup.ts.atm.controller.handler.scripts

import com.openwaygroup.ts.atm.controller.handler.scripts.encoding.Encoder
import com.openwaygroup.ts.atm.controller.handler.scripts.encoding.EncoderT
ype

```

```
import com.openwaygroup.ts.atm.controller.handler.scripts.encoding.StringEncoder
import groovy.transform.BaseScript
@BaseScript com.openwaygroup.ts.platform.ext.script.BaseScript thisScript
encoderScript {

    def rus = new StringEncoder(controlSequence: "\u001B(1", charMap: [
        "A": "A",
        "B": "B",
        "C": "C",
        "D": "D",
        "E": "E",
        "F": "F",
        "G": "G",
        "H": "H",
        "I": "I",
        "J": "J",
        "K": "K",
        "L": "L",
        "M": "M",
        "N": "N",
        "O": "O",
        "P": "P",
        "Q": "Q",
        "R": "R",
        "S": "S",
        "T": "T",
        "U": "U",
        "V": "V",
        "W": "W",
        "X": "X",
        "Y": "Y",
        "Z": "Z",
        "a": "A",
        "b": "B",
        "c": "C",
        "d": "D",
        "e": "E",
        "f": "F",
        "g": "G",
        "h": "H",
```

```
"i": "I",  
"j": "J",  
"k": "K",  
"l": "L",  
"m": "M",  
"n": "N",  
"o": "O",  
"p": "P",  
"q": "Q",  
"r": "R",  
"s": "S",  
"t": "T",  
"u": "U",  
"v": "V",  
"w": "W",  
"x": "X",  
"y": "Y",  
"z": "Z",  
"Ё": "t",  
"ё": "t",  
"А": "f",  
"Б": "~",  
"В": "d",  
"Г": "u",  
"Д": "l",  
"Е": "t",  
"Ж": "|",  
"З": "p",  
"И": "b",  
"Й": "q",  
"К": "r",  
"Л": "k",  
"М": "v",  
"Н": "y",  
"О": "j",  
"П": "g",  
"Р": "h",  
"С": "c",  
"Т": "n",  
"У": "e",  
"Ф": "a",
```

```
"X": "{",  
"Ц": "w",  
"Ч": "x",  
"Ш": "i",  
"Щ": "o",  
"Ъ": "}",  
"Ы": "s",  
"Ь": "m",  
"Э": "`",  
"Ю": "\\u007F",  
"Я": "z",  
"а": "f",  
"б": "~",  
"в": "d",  
"г": "u",  
"д": "l",  
"е": "t",  
"ж": "|",  
"з": "p",  
"и": "b",  
"й": "q",  
"к": "r",  
"л": "k",  
"м": "v",  
"н": "y",  
"о": "j",  
"п": "g",  
"р": "h",  
"с": "c",  
"т": "n",  
"у": "e",  
"ф": "a",  
"х": "{",  
"ц": "w",  
"ч": "x",  
"ш": "i",  
"щ": "o",  
"ъ": "}",  
"ы": "s",  
"ь": "m",  
"э": "`",
```

```
"ю": "\u007F",
"я": "z"

])
def en = new StringEncoder(controlSequence: "\u001B(2", charMap: [
    "A": "A",
    "B": "B",
    "C": "C",
    "D": "D",
    "E": "E",
    "F": "F",
    "G": "G",
    "H": "H",
    "I": "I",
    "J": "J",
    "K": "K",
    "L": "L",
    "M": "M",
    "N": "N",
    "O": "O",
    "P": "P",
    "Q": "Q",
    "R": "R",
    "S": "S",
    "T": "T",
    "U": "U",
    "V": "V",
    "W": "W",
    "X": "X",
    "Y": "Y",
    "Z": "Z",
    "a": "A",
    "b": "B",
    "c": "C",
    "d": "D",
    "e": "E",
    "f": "F",
    "g": "G",
    "h": "H",
    "i": "I",
    "j": "J",
    "k": "K",
```

```
"l": "L",  
"m": "M",  
"n": "N",  
"o": "O",  
"p": "P",  
"q": "Q",  
"r": "R",  
"s": "S",  
"t": "T",  
"u": "U",  
"v": "V",  
"w": "W",  
"x": "X",  
"y": "Y",  
"z": "Z",  
"Ė": "t",  
"ė": "t",  
"A": "f",  
"Б": "~",  
"B": "d",  
"Г": "u",  
"Д": "l",  
"E": "t",  
"Ж": "|",  
"З": "p",  
"И": "b",  
"Й": "q",  
"K": "r",  
"Л": "k",  
"M": "v",  
"H": "y",  
"O": "j",  
"П": "g",  
"P": "h",  
"C": "c",  
"T": "n",  
"Y": "e",  
"Φ": "a",  
"X": "{",  
"Ц": "w",  
"Ч": "x",
```



```
"Ш": "i",  
"Щ": "o",  
"Ъ": "}",  
"Ы": "s",  
"Ь": "m",  
"Э": "`",  
"Ю": "\\u007F",  
"Я": "z",  
"а": "f",  
"б": "~",  
"в": "d",  
"г": "u",  
"д": "l",  
"е": "t",  
"ж": "|",  
"з": "p",  
"и": "b",  
"й": "q",  
"к": "r",  
"л": "k",  
"м": "v",  
"н": "y",  
"о": "j",  
"п": "g",  
"р": "h",  
"с": "c",  
"т": "n",  
"у": "e",  
"ф": "a",  
"х": "{",  
"ц": "w",  
"ч": "x",  
"ш": "i",  
"щ": "o",  
"ъ": "}",  
"ы": "s",  
"ь": "m",  
"э": "`",  
"ю": "\\u007F",  
"я": "z"
```

])

```
def en2 = new StringEncoder(controlSequence: "\u001B(I", charMap: [  
    "A": "A",  
    "B": "B",  
    "C": "C",  
    "D": "D",  
    "E": "E",  
    "F": "F",  
    "G": "G",  
    "H": "H",  
    "I": "I",  
    "J": "J",  
    "K": "K",  
    "L": "L",  
    "M": "M",  
    "N": "N",  
    "O": "O",  
    "P": "P",  
    "Q": "Q",  
    "R": "R",  
    "S": "S",  
    "T": "T",  
    "U": "U",  
    "V": "V",  
    "W": "W",  
    "X": "X",  
    "Y": "Y",  
    "Z": "Z",  
    "a": "A",  
    "b": "B",  
    "c": "C",  
    "d": "D",  
    "e": "E",  
    "f": "F",  
    "g": "G",  
    "h": "H",  
    "i": "I",  
    "j": "J",  
    "k": "K",  
    "l": "L",  
    "m": "M",  
    "n": "N",
```

```
"o": "O",  
"p": "P",  
"q": "Q",  
"r": "R",  
"s": "S",  
"t": "T",  
"u": "U",  
"v": "V",  
"w": "W",  
"x": "X",  
"y": "Y",  
"z": "Z",  
"Ё": "t",  
"ё": "t",  
"А": "f",  
"Б": "~",  
"В": "d",  
"Г": "u",  
"Д": "l",  
"Е": "t",  
"Ж": "|",  
"З": "p",  
"И": "b",  
"Й": "q",  
"К": "r",  
"Л": "k",  
"М": "v",  
"Н": "y",  
"О": "j",  
"П": "g",  
"Р": "h",  
"С": "c",  
"Т": "n",  
"У": "e",  
"Ф": "a",  
"Х": "{",  
"Ц": "w",  
"Ч": "x",  
"Ш": "i",  
"Щ": "o",  
"Ъ": "}"
```

```
"Ы": "s",
"Ь": "m",
"Э": "`",
"Ю": "\u007F",
"Я": "z",
"a": "f",
"б": "~",
"в": "d",
"г": "u",
"д": "l",
"е": "t",
"ж": "|",
"з": "p",
"и": "b",
"й": "q",
"к": "r",
"л": "k",
"м": "v",
"н": "y",
"о": "j",
"п": "g",
"р": "h",
"с": "c",
"т": "n",
"у": "e",
"ф": "a",
"х": "{",
"ц": "w",
"ч": "x",
"ш": "i",
"щ": "o",
"ъ": "}",
"ы": "s",
"ь": "m",
"э": "`",
"ю": "\u007F",
"я": "z"

])

def all = new StringEncoder(charMap: [
    "A": "A",
    "B": "B",
```

```
"C": "C",  
"D": "D",  
"E": "E",  
"F": "F",  
"G": "G",  
"H": "H",  
"I": "I",  
"J": "J",  
"K": "K",  
"L": "L",  
"M": "M",  
"N": "N",  
"O": "O",  
"P": "P",  
"Q": "Q",  
"R": "R",  
"S": "S",  
"T": "T",  
"U": "U",  
"V": "V",  
"W": "W",  
"X": "X",  
"Y": "Y",  
"Z": "Z",  
"a": "A",  
"b": "B",  
"c": "C",  
"d": "D",  
"e": "E",  
"f": "F",  
"g": "G",  
"h": "H",  
"i": "I",  
"j": "J",  
"k": "K",  
"l": "L",  
"m": "M",  
"n": "N",  
"o": "O",  
"p": "P",  
"q": "Q",
```

```
"r": "R",  
"s": "S",  
"t": "T",  
"u": "U",  
"v": "V",  
"w": "W",  
"x": "X",  
"y": "Y",  
"z": "Z",  
"Ё": "t",  
"А": "f",  
"Б": "~",  
"В": "d",  
"Г": "u",  
"Д": "l",  
"Е": "t",  
"Ж": "|",  
"З": "p",  
"И": "b",  
"Й": "q",  
"К": "r",  
"Л": "k",  
"М": "v",  
"Н": "y",  
"О": "j",  
"П": "g",  
"Р": "h",  
"С": "c",  
"Т": "n",  
"У": "e",  
"Ф": "a",  
"Х": "{",  
"Ц": "w",  
"Ч": "x",  
"Ш": "i",  
"Щ": "o",  
"Ъ": "}",  
"Ы": "s",  
"Ь": "m",  
"Э": "`",  
"Ю": "\\u007F",
```

```
"Я": "z",
"a": "f",
"б": "~",
"в": "d",
"г": "u",
"д": "l",
"е": "t",
"ж": "|",
"з": "p",
"и": "b",
"й": "q",
"к": "r",
"л": "k",
"м": "v",
"н": "y",
"о": "j",
"п": "g",
"р": "h",
"с": "c",
"т": "n",
"у": "e",
"ф": "a",
"х": "{",
"ц": "w",
"ч": "x",
"ш": "i",
"щ": "o",
"ъ": "}",
"ы": "s",
"ь": "m",
"э": "`",
"ю": "\u007F",
"я": "z",
"ё": "t",

])

def encoder1 = new Encoder(type: EncoderType.PRINTER,
vendor: "NCR").add(en).add(rus).add(en2).add(all)

def encoder2 = new Encoder(type: EncoderType.SCREEN,
vendor: "NCR").add(all)

def encoder3 = new Encoder(type: EncoderType.PRINTER,
vendor: "NCR3G").add(en).add(rus).add(en2).add(all)

// load to encoder store
```

```
addEncoder (encoder1)
addEncoder (encoder2)
addEncoder (encoder3)
}
```