



Installation and Configuration Manual

ATM Controller R2

03.52.30

30.07.2021

Contents

1	ATM controller dictionaries	5
1.1	Customizing ATM Controller dictionaries	5
1.1.1	ATM types dictionary	5
1.1.2	ATM denominations dictionary	7
1.2	Fixed dictionaries	10
1.2.1	ATM protocols dictionary	10
1.2.2	ATM operations dictionary	10
1.2.3	ATM hardware types dictionary	11
1.2.4	ATM message types dictionary	12
2	Description and configuration of a new ATM	15
2.1	Setting up an ATM contract and its device	15
2.1.1	Setting up the executable range of ATM operations	15
2.1.2	Configuring ATM hardware components	16
2.1.3	Managing ATM state	18
2.1.4	Specifying encryption keys	18
2.1.5	Enabling MAC signature mode	20
2.1.6	Additional device parameters	20
2.2	Configuring the ATM connection with the Way4 host	20
2.3	Preparing and loading ATM configurations	22
2.3.1	Preparing configuration files	22
2.3.2	Sending a configuration to an ATM	24
2.3.3	Sending a configuration to all ATMs	24
3	ATM controller configuration files	25
3.1	Controller configuration file	25
3.2	Configuration files for controller interaction with the ATM	26
3.2.1	Principles of working with configuration scripts	27
3.2.2	Request processing configuration file	29
3.2.3	Response processing configuration file	32
3.3	Configuring receipt and screen templates	35
3.3.1	Encoding configuration	44

3.3.2	Templates testing	54
4	Settlement scheme for ATM operations	61
5	Automatic reversal message creation	65

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 center. 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 center employees) responsible for configuring the ATM network.

When working with this document, it is recommended to use the following resources from the OpenWay documentation series:

- DB Manager Manual
- Way4 Transaction Switch. Platform Overview
- Daily Procedures
- Acquiring Module
- Issuing Module
- ATM Monitoring
- Way4 Service Packages
- Configuration of Client Messages

The following notation is used in the document:

- Screen form field labels are shown in *italics*.
- Screen form button labels are shown in square brackets, such as [Approve].
- Sequences for selecting user menu items are shown using arrows as follows: "Issuing → Contracts Input & Update".
- Sequences for selecting system menu items are shown using arrows as follows: "Database => Change password".
- Key combinations in DB Manager are shown in angular brackets, for example <Ctrl>+<F3>.



Warnings about potentially hazardous situations or actions are marked with a special icon and highlighted.



Information about important features, additional options, or the best use of certain system functions is marked with a special icon and highlighted.

1 ATM controller dictionaries

Dictionaries are an important source of information used during ATM controller operation. 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 can only be changed by OpenWay representatives; in some cases, these dictionaries can be modified by bank or processing center specialists under the supervision of OpenWay specialists.

1.1 Customizing ATM Controller dictionaries

1.1.1 ATM types dictionary

All types of ATMs interacting with Way4 must be registered in a special ATM types dictionary.

An ATM type is set during device configuration (see the section "Device Information" of the document "Acquiring Module").

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

ATM Types											<< >>		38 of 49	X
Code	Name	Brand	Model	Conf File	Outgoing Console	Cash Account	Cash Account	Cash Account	Cash Account	EMV Conf File	Transaction Attributes	Protocol Id		
NCR6638	NCR SelfSer	NCR	6638	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser			MPD:10100100		NDC+		
NCR6634	NCR SelfSer	NCR	6634	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser			MPD:10100100		NDC+		
NCR6632	NCR SelfSer	NCR	6632	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser			MPD:10100100		NDC+		
NCR6626	NCR SelfSer	NCR	6626	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser			MPD:10100100		NDC+		
NCR6625	NCR SelfSer	NCR	6625	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser			MPD:10100100		NDC+		
NCR6622	NCR SelfSer	NCR	6622	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser			MPD:10100100		NDC+		
NCR5884	NCR 5884 (P	NCR	5884	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser			MPD:10100100		NDC+		
NCR5874	NCR 5874 (P	NCR	5874	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser			MPD:10100100		NDC+		
NCR5870	NCR 5870 (P	NCR	5870	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser			MPD:10100100		NDC+		
NCR5840	NCR 5840 (P	NCR	5840	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser					NDC+		
NCR5684	NCR 5684	NCR	5684	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser					NDC+		
NCR5674	NCR 5674	NCR	5674	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser					NDC+		
NCR5670	NCR 5670	NCR	5670	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser					NDC+		
NCR5305	NCR 5305 A	NCR4G-ASC	5305	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser					NDC+		
NCR5084	NCR 5084 A	NCR3G-ASC	5084	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser					NDC+		
NCR5084	NCR 5084	NCR3G	5084	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser				ATM_TERM_TRACK,TERM_CC	NDC+		
NCR5070	NCR 5070 A	NCR3G-ASC	5070	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser					NDC+		
NCR5070	NCR 5070	NCR3G	5070	conf/atm/ndc	Outgoing Console	ATM Cassette	Cash Dispenser					NDC+		

Table containing ATM types registered in Way4

This table contains the following fields:

- Code – the code identifying the ATM type.
- Name – a description of the ATM type.

- *Brand* – nominal ATM brand, which in combination with the Protocol defines the dialect used by the controller:
 - Aptra NDC – brands: NCR, NCR4G, NCR5G, NCR4G-ASD, NCR5G-ASD, PERSONAS, GRG, WINCOR-APTRA, NAUTILUS-APTRA, DIEBOLD.
 - ProCash NDC – brands: WINCOR, BANQIT.
 - ProCash MDS912 – brands: WINCOR, BANQIT, GRG, NAUTILUS.
 - Diebold MDS912 – brands: DIEBOLD, IBM.
- *Model* – the ATM model.
- *Configuration File* – name of the configuration file with its path relative to the directory WEB-INF/inv/atm/conf (the inv/atm/conf directory must be independently created 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 the section "[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 to the grid form, click the [Ins] button; 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 "Information about Device Contracts" of the document "Acquiring Module"), 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 the section "[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 the section "[ATM hardware types dictionary](#)").

The [DevAttrs] button is used to specify a list of device attributes that will be recorded in transaction documents for this type of ATM (and can further be included in exported clearing information). The list is generated using the values selected in the "DevAttrs for <ATM type>" form:

DevAttrs for NCR 5084
<< < > >>
2 of 21
b x

Attribute Name	Attribute Presence or Value
Terminal Type	ATM
Card data input capability: Magnetic stripe	Yes
Card data input capability: Bar code	[None]
Card data input capability: OCR	No
Card data input capability: Chip	Yes
Card data input capability: Key entry	
Card data input capability: Contactless Chip	
Card data input capability: Contactless Magnetic stripe	
Card data input capability: 3-D Secure	No
Card data input capability: Wallet	No
Cardholder authentication capability: Manual signature verification	No
Cardholder authentication capability: Electronic signature analysis	No
Cardholder authentication capability: Online PIN (default for simple PIN)	No
Cardholder authentication capability: Offline PIN (default for offline PIN)	No
Cardholder authentication capability: Soft PIN-Pad	No
Cardholder authentication capability: No capability	No
Card capture capability	Capture
Location indicator	
Card data output capability	
Terminal output capability	
PIN capture capability	

Ins Del Query Set

To approve the list of selected values, click [Set]. The list of codes for the corresponding attributes will be shown in the *Transaction Attributes* field of the "ATM Types" form.

1.1.2 ATM denominations dictionary

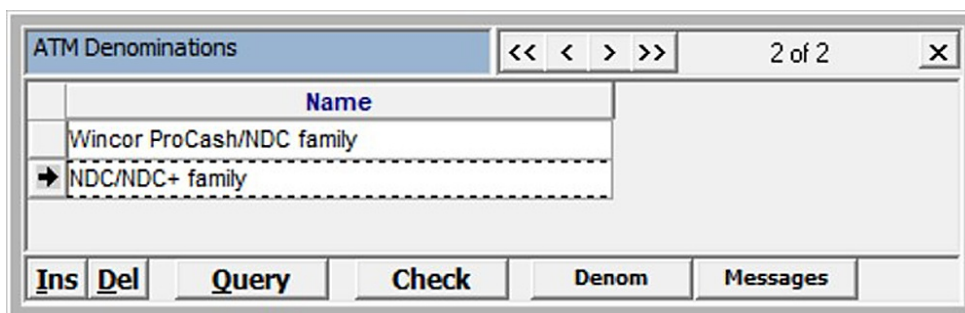
Each ATM in the system is assigned a denomination type that defines the type of cassette that can be used by the ATM, the face value and currency of notes stored in the cassette during device configuration, as well as some parameters for cash withdrawal.

Banknote dispensing rules can be defined by ctx.dispenseType parameter (see the section "[Request processing configuration file](#)"):

- MINIMAL_NOTES – minimizing number of banknotes (if possible, the highest denomination banknotes are selected).
- MINIMAL_VALUES – minimizing denomination of banknotes (if possible, the lowest denomination banknotes are selected).
- PENALTY_BASED – in accordance with the specified compiling algorithm (see the description of the Dispense Parameters field below).

An ATM's denomination type is selected from a list during device setup (see the section "Device Information" of the document "Acquiring Module").

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

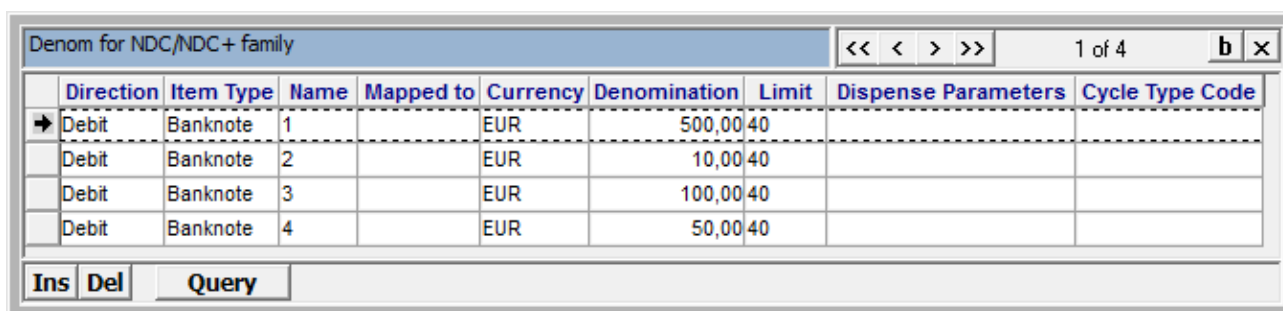


Denominations dictionary

To add a record to the grid 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 "Information about Device Contracts" of the document "Acquiring Module"), a warning will be displayed.

Every denomination type corresponds to certain parameters that define the way the cassette is loaded with cash and the algorithm for dispensing cash. To access the table 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>" grid form will be displayed.



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		

Parameters for loading a cassette with notes

This table contains the following fields:

- *Direction* – cassette use according to the direction of funds ("Debit" – dispense, "Credit" – accept).
- *Item Type* – cassette type ("Banknote" or "Coin" (working with coins is not supported in the current version)).
- *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 the cassette for cash acceptance (for example, rejected notes from the current cassette).
- *Currency* – drop-down list for selecting the name of the currency for notes loaded to the cassette.
- *Denom* – denomination of notes loaded in the cassette.
- *Limit* – the maximum number of notes (from 0 to 999) that can be dispensed from a cassette during one operation.
- *Dispense Parameters* – rule for compiling a set of notes 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 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.
- PENALTY_INCR – increases the penalty for each subsequent note 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.

Integers from 0 to 99 (inclusively) can be specified as penalty values. For efficiency, it is recommended to use values in the range from 0 to 30.



The algorithm for compiling notes depending on the menu item selected at the ATM is determined in the controller configuration (see the section "[Request processing configuration file](#)") using the `ctx.dispenseAlgorithmId` parameter (by default – 1 (the first algorithm is used)).

Notes are selected from ATM cassettes when dispensing cash in such a way as to minimize the conditional penalty for dispensing a certain set of notes with consideration of limits.

The value of the conditional penalty 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 is the number of cassettes from which notes are selected.
- n_i is the number of notes selected from the i -th cassette.
- CST_FEE_i is the penalty for dispensing notes from the i -th cassette (value of the PENALTY tag for the selected algorithm).
- $CST_FEE_INCREASE_i$ is 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).

To check whether the data is filled in correctly when adding or editing entries in the "Denom for <denomination type>" form, select the denomination type in the "ATM Denominations" form and click [Check]. Entries that meet the following conditions are considered valid:

- the *Name*, *Direction*, *Item Type*, *Currency*, and *Denomination* fields are not empty.
- the *Dispense Parameters* field does not contain the ";" character (semicolon).

The check ends with the corresponding message. To get more information about the verification results, use the [Messages] button.

1.2 Fixed dictionaries

1.2.1 ATM protocols dictionary

ATM protocols regulate message format and rules for transmitting information between an ATM and a processing center.

The ATM protocols dictionary is contained in the "ATM Protocols" table.

Name	Code	Is Adjusting Totals
NDC+	NDC+	

Types of protocols for connecting ATMs to the processing center

This table 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+" and "MDS912" protocol.

1.2.2 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 (see the section "[ATM hardware types dictionary](#)") necessary for those operations.

The ATM operations dictionary is contained in the "ATM Operations" table.

Name	Code	Transaction Type	Ans Col	Request	Category	Service Class	Checked	Downgradable To	Requires Cassette	Is Online	Use Coin	Is P
Bill Payments by Cash	T-Y1	Pay by Cash		Advice	Authorisation				No	No	No	P1_A
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 (C	T-W1_B	PWC: 2.Cash Dispens		Advice	Authorisation	Transaction			Yes	No	Yes	

List of ATM operations

This table contains the following fields:

- *Name* – transaction name

- *Code* – transaction 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/authorization message category
- *Service Class* – the bank's transaction classification; the value of this parameter determines the way a document will be processed in Way4; if this field is empty, it means the default value *Service Class* = "Transaction" is used
- *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 (this functionality is not supported in the current version)
- *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 [Monitoring ATM Networks]" of the document "ATM Monitoring R2".

The [Required] button is used to call up the "Required for <name of operation>" grid form, which contains a list of ATM components (see the section "[ATM hardware types dictionary](#)") necessary for fulfilling a given operation.

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

1.2.3 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 the section "[ATM operations dictionary](#)").

The ATM Hardware Types dictionary is contained in the "ATM Hardware Types" table.

ATM Hardware Types			
<< < > >>		1 of 25	
Name	Code	Protocol	
Withdrawal Area	Wd	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+	

Table showing ATM hardware components

This table 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 opens the "Messages for <component name>" grid form, containing a list of messages generated in the system when working with this component (see the section "[ATM message types dictionary](#)").

The list of ATM components is contained in the section "ATM Hardware [Monitoring ATM Networks]" of the document "ATM Monitoring R2".

1.2.4 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 "ATM Status [Monitoring ATM Networks]" of the document "ATM Monitoring R2") 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 for the corresponding ATM are recorded in the Way4 database; each message belonging to a certain ATM component.

The dictionary of possible ATM message types is contained in the "ATM Message Types" table.

ATM Message Types								<< < > >>		9056 of 11772	X
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			

Messages generated during ATM operation

This table contains the following fields:

- *Message Scope* – message recipient/source:
- "Device" – ATM
- "Device Group" – ATM group
- "Physical Channel" – controller
- *Protocol* – protocol name (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 the section "[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* – 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") – hardware component status is "OK"; ATM status is "OK" if there no errors on other hardware , and there haven't been any; ATM status is "Information" if there are no errors on other hardware , but there was an error on this hardware; ATM status is "Information", "Warning" or "Error" depending on the status of other hardware.
- "Information" (code "1") – hardware status and ATM status does not change.
- "Warning" (code "2") – hardware status is "Warning"; ATM status is "Information".
- "Error" (code "3") – hardware status is "Error"; ATM status is "Warning".
- "Not Configured" (code "4") – hardware status is "Not Configured"; ATM status is "Information".

- "Unavailable" (code "6") – hardware status is "Unavailable"; ATM status is "Information".
- "Fatal Error" (code "5") – hardware 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 "User Groups and Users – View" grid form (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.

2 Description and configuration of a new ATM

2.1 Setting up an ATM contract and its device

A description of how a new ATM contract is entered may be found in the section "Information about Device Contracts" of the document "Acquiring Module".

Setup of devices for ATM contracts is described in the section "Device Information" of the document "Acquiring Module").

2.1.1 Setting up the executable range of ATM operations

The executable range of ATM operations is set up through the grid form "Operations for <name of ATM>".

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 [Operations] button in the form "ATM Device Management".
- After selecting from the user menu "Acquiring → Acquiring Contracts", select the desired account contact 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				
<div> <div><< < > >></div> <div>1 of 38</div> <div>b x</div> </div>				
Operation Type	Status	Hardware Problem	Last Changed	
→ Cash Dispense with ticket	Active		27/05/02 14:26.38	
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 key 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	

Ins

Del

Query

Ch Status

History

List of executable ATM operations

An operation can be deleted from the list by selecting the desired row in the table 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.

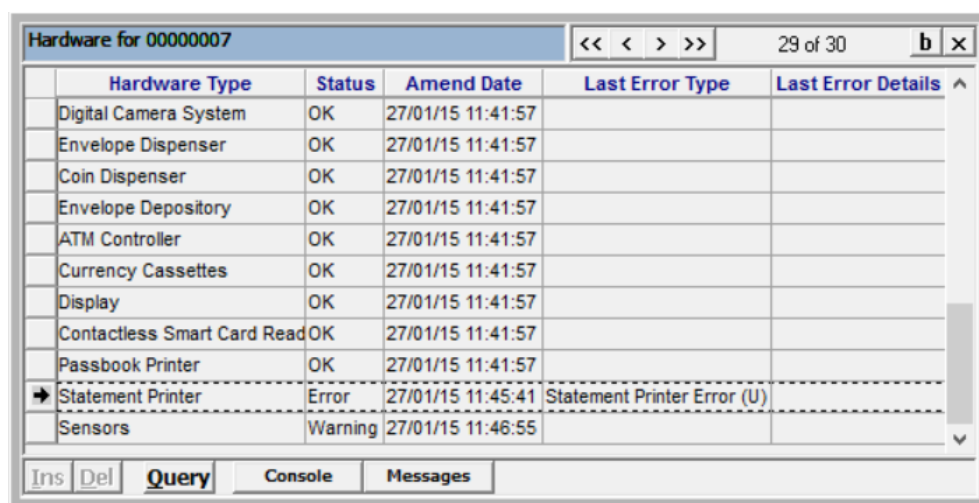
To restore the list of allowed operations (according to the "ATM Operations" dictionary) after rows have been deleted from the table, click the [Setup] button and choose the [Check and Fill] context menu item in the "ATM Device Management" form or the "ATM for <device identifier>" device configuration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

2.1.2 Configuring ATM hardware components

ATM components can be configured through the grid form "Hardware for <name of ATM>".

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 [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.



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		

Table 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 the section "[ATM 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 the section "[ATM hardware types dictionary](#)").



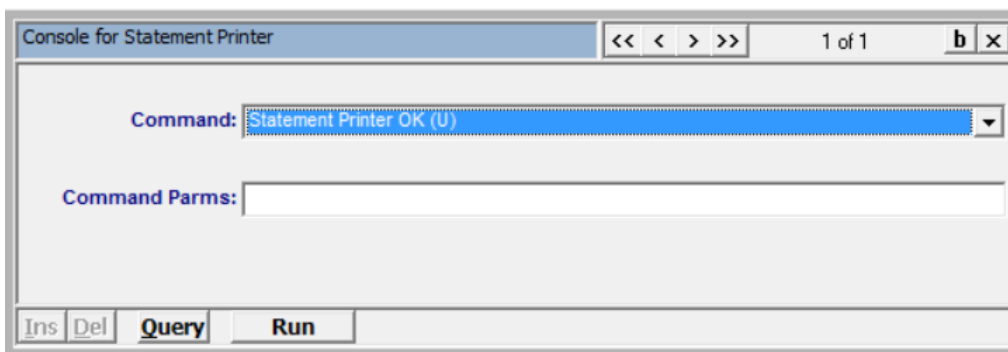
In this case components that are missing in the ATM must be disabled by setting their status to "Not Configured" (the procedure for changing components status is described below).

- 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 table and click on the [Console] button. By this command the screen will display the form "Console for <name of component>". Select in the *Command* field of the form the desired control command, for example, "<name of component> OK", and click on the [Run] button.



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 and click the [Run] button. For more information about ATM issuer scripts, see the section "ATM Status [Monitoring ATM Networks]" of the document "ATM Monitoring R2".

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

Messages for Statement Printer						<< < > >>		3 of 93	b x
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		

Grid form containing system messages generated during changes in ATM component status

2.1.3 Managing ATM state

To show information about an ATM's current state, click on the [State] button in the "ATM Device Management" form or in the "ATM for <device ID>" device configuration form. The following information will be shown:

- **Status** – current status of the ATM ("OK", "Information", "Warning", "Error", "Not Configured", and "Fatal Error").
- **Online** – indicates if an ATM is available online ("Yes"/"No").
- **Online Service** – communication service code identifying the connection between the controller and the ATM.

To switch an ATM to an operational state ("OK" status), select the [Set to OK] context menu item of the [Setup] button in the "ATM Device Management" form or in the "ATM for <device ID>" device configuration form. All components of this device will be switched to the "OK" status.

To temporarily suspend operation of the ATM, when Way4 ignores all requests and messages received from a device, select the [Set to Repair] context menu item of the [Setup] button. The ATM status will be changed to "Not Configured". Operation of the ATM will be resumed when the status is changed to "OK" (as described above).

2.1.4 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 only determined by the encryption key value and does not depend on how it was encrypted.

Specification of encryption keys is accomplished through the form "Keys for <name of ATM>".

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 contact 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.

Keys for TEST ACQ										<< < > >>		1 of 3		b x	
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	M			
3DES ABA	Terminal Authentication	Terminal Authentication	U9CB60C31A96C742	5AE541				Active	18/11/14 00:00/00/00	0	0	0			
3DES ABA	Terminal PIN Key	Terminal PIN Key	UD387E838B70E8FA8A7223F					Active	18/11/14 00:00/00/00	0	0	0			
3DES ABA	Terminal Authentication	Terminal Authentication						Inactive	20/11/14 00:00/00/00	0	0	0			
<div>< [] ></div>															
<div>Ins Del Query Manage Key Options</div>															

Form for the specification of ATM encryption keys

This form contains the following fields:

- **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.

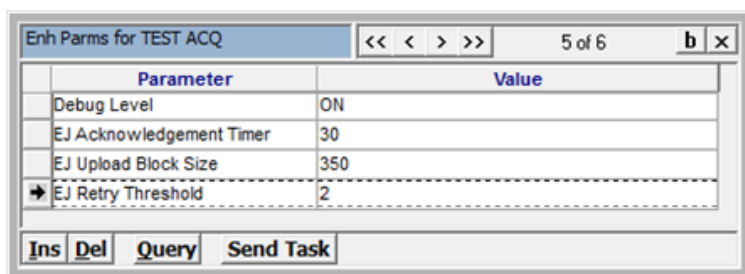
To add a record to this table, click the [Ins] button; to delete a record, click [Delete]

2.1.5 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.

2.1.6 Additional device parameters

The list of device configuration parameters can be expanded by specifying additional parameters in the "Enh Parms for <ATM name>" grid form. 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

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.

2.2 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.

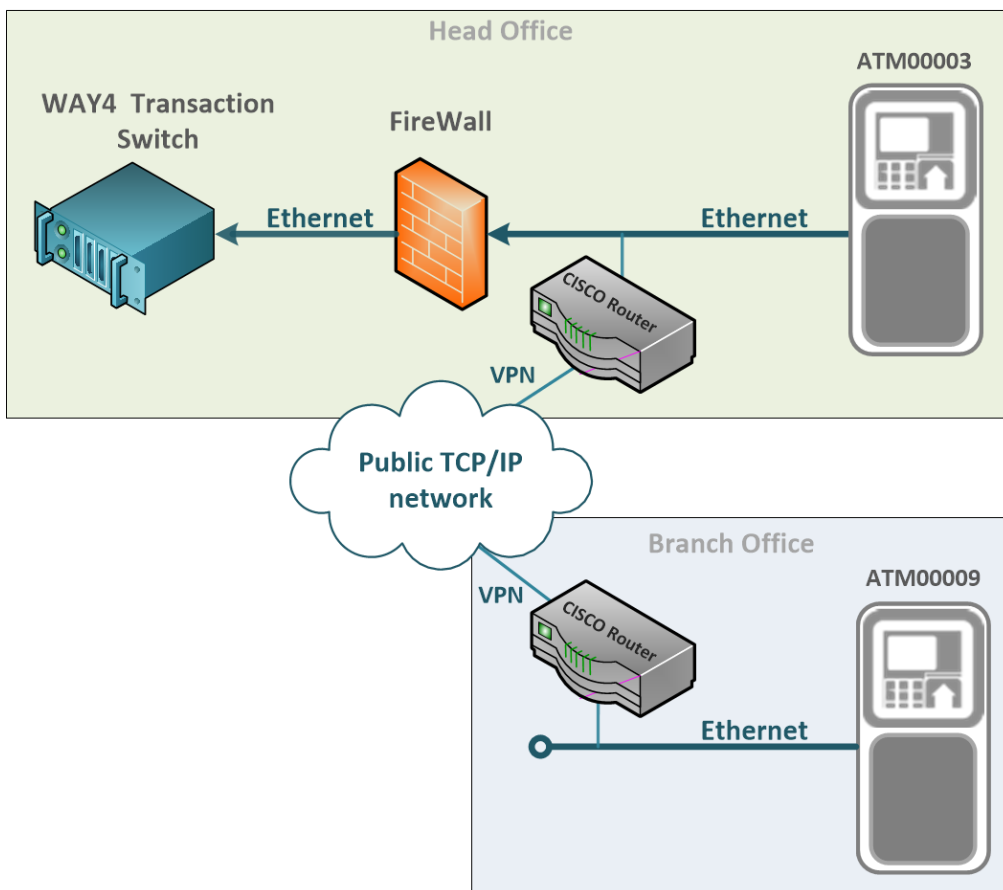
To get up-to-date information about the state of a connection, a keep-alive timer must be set up (see `tcp_keepAlive` parameter description in the section "[Controller configuration file](#)").

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 the section "[ATM controller configuration files](#)"). 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 "Device Information" of the document "Acquiring Module").

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 dynamically switch 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 shows the scheme for connecting ATMs to Way4 Transaction Switch over dedicated lines using the TCP/IP protocol.



ATMs connected to Way4 Transaction Switch on dedicated lines using the TCP/IP protocol

2.3 Preparing and loading ATM configurations

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

2.3.1 Preparing 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 the section "[ATM types dictionary](#)").

Fragment of the configuration file:

```
100002001 cfJGTEMPORARILY OFF-LINE
210000205 J0550000550550000000000000000
300000 0000000003203000116000_000_006075070020807509072
510000000 00010309800425525500000013200000000003113800
@100000001 C04ZZZZ5ZZ9699
A00000 000_006075070020807509072
B10000 0000000000000000000000000000
G00000D0010 034
*00000#S332
```

ATM configuration files have the following format:

Position	Value	Parameter
1	1	ATM screen template.
	2	ATM basic state.
	3	Configuration parameters, timing delay values.
	5	Financial institution tables.
	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.

Position	Value	Parameter
3-6	Number from 0000 to 9999	Configuration ID.
7-...	Characters according to the device specification	Data on basic states, screen templates, financial institutions, etc.

2.3.2 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 "ATM Status [Monitoring ATM Networks]" of the document "ATM Monitoring R2").

2.3.3 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 "ATM Status [Monitoring ATM Networks]" of the document "ATM Monitoring R2"). 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 "Device Information" of the document "Acquiring Module").

3 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 authorization 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).

3.1 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 (recommended value – "true").
- 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.
- allowedOperations - list of operations (separated by spaces) that are supported in addition to the set of basic "ATM Management" operations. Functionality implemented by the operations in the list is provided according to an additional agreement with the OpenWay.

Basic "ATM Management" operations (supported by default):

- CASH_WITHDRAWAL – cash withdrawal
- BALANCE_INQ – balance inquiry
- MINI_STATEMENT – mini-statement request
- PIN_CHANGE – PIN change
- REPLENISHMENT – ATM cassette replenishment
- END_OF_DAY – getting a statement on the state of cassettes
- ATM_SERVICE – ATM technical service

When the ATMController service is started, the full list of available operations is recorded in the Transaction Switch log and in the Way4 database (can be viewed in the form "Full\Process Log\Process Log").

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"
    allowedOperations="OPER_NAME1 OPER_NAME2">
    <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>
```

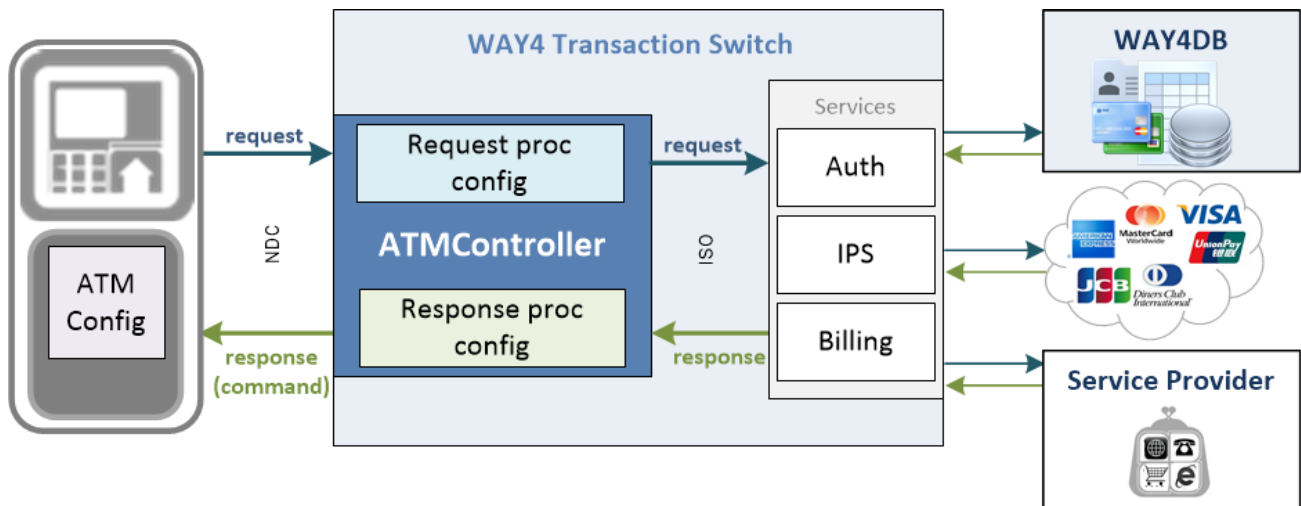


The configuration file should only be changed under the supervision of OpenWay representatives.

3.2 Configuration files for controller interaction with the ATM

Setup of controller interaction with ATMs includes:

- Defining rules for processing requests from ATMs, for their further conversion to required format and transmission to Way4 Transaction Switch interface services (with bank, payment system or payment acceptance system) (see the section "[Request processing configuration file](#)").
- Defining rules for processing responses received from interface channels to generate the corresponding commands to the ATM (see the section "[Response processing configuration file](#)").



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/>).

3.2.1 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 uses the /inv/<adm_conf>/resources/configuration.prd file to work with the current configuration.

If you change the configuration scripts, use the following console commands to apply the new settings:

- compileScripts – to compile of new scripts based on changes made to WEB-INF/inf/adm/scripts/ and create a new configuration.prd file.
- installScripts – to install of new configuration.pdf file for ATM controller.

If compilation of new scripts is not successful, the corresponding errors are logged in a log file (displayed in the console window), a new configuration.prd is not created.



One of possible compilation errors is a "Method too large" exception. This error occurs when the compiled code of one of groovy script methods exceeds 64 Kbytes (this limitation is determined by JVM). To solve this problem, it is necessary to optimize the code of the configuration groovy scripts. For example, if possible, to describe some logical blocks (especially duplicated ones) as separate called methods.

After successful compilation, the old version of the configuration is archived in a configuration.prd.yyyyMMddHHmmss file. "yyyymmddHHmmss" – the date and time when this version was compiled. For example, if the configuration was created on February 6, 2019 at 15:20:00, the archive file will have the name configuration.prd.20190206152000.

These compilation steps do not apply to the Encoding.groovy file located in the WEB-INF/inf/adm/scripts/directory.

When the controller starts, a check is made for availability of scripts and the configuration.prd file. If there are no scripts or configuration.prd, the service does not start. If scripts are absent, and the configuration.prd file is not valid, the controller ceases to start. If the configuration.prd file is absent (or invalid), it is created automatically based on scripts in the WEB-INF/inv/atm/scripts/ directory.

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 the section "[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 value like com.openwaygroup.ts.fp.shared.atm.replenishment.ReplenishmentCycleType:

```
proc.closeCycle.exec { // process of cash withdrawal cycle closing
    ReplenishmentCycleType.CASH_OUT
}

proc.closeCycle.exec { // process of note acceptance cycle closing
    ReplenishmentCycleType.CASH_IN
}
```

- 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; there may be value like com.openwaygroup.ts.fp.shared.atm.scenario.ScenarioType:

```
proc.setPostProcess.exec { // BNA configuration request
    ScenarioType.RETRIEVE_NOTE_DEFS
}
```

3.2.2 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.

Sample configuration script:

```

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 environment (the next string is not mandatory but
//this declaration is convenient for autocompletion in 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 possible 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 ~/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())
//or from 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
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, 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.
//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 on the 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 the section "[Response processing configuration file](#)").

3.2.3 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 the section "[Request processing configuration file](#)"). To process operation results, several scripts can be used describing the procedure for performing additional actions, generating receipts, screens, etc.

Sample configuration script:


```

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.printReferen
ce
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 == 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")
        }
    }
    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 an ATM financial cycle in the Way4 DB.
proc.closeCycle.exec {
    ReplenishmentCycleType.COINS_OUT
}
//Use of an object-link to processes
// for closing a different type of financial cycle in the Way4 DB.
proc.setInReplenishment.exec()
ctx
```

3.3 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 objects may be used in templates:

Name	Type	Fields	Comments
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		Authorization code.

Name	Type	Fields	Comments
lang	java.lang.String		Transaction language.
operation	java.lang.String		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	<ul style="list-style-type: none"> currency externalCurrency amount 	Cash withdrawal amount and currency.
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>	<ul style="list-style-type: none"> amountType accountType 	List of accounts, amounts and currencies.
statement	java.util.List<com.openwaygroup.ts.fp.shared.MinistatementRecord>	<ul style="list-style-type: none"> date operationType amount 	List with ministatement records.

Name	Type	Fields	Comments
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	<ul style="list-style-type: none"> country city address state postalCode street 	Structure describing ATM address data.
merchant	com.openwaygroup.ts.fp.shared.Merchant	<ul style="list-style-type: none"> name id SIC SICGroupCode url 	Information about the merchant.
acquirer	com.openwaygroup.ts.fp.shared.acquirer	<ul style="list-style-type: none"> bin location acqBankCode 	Information about the acquirer.
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 the section " Configuration files for controller interaction with the ATM ").
cardsRetained	java.lang.Long		Number of retained cards.

Name	Type	Fields	Comments
cassettes	java.util.List<com.openwaygroup.ts.fp.shared.atm.device.Cassette>	<ul style="list-style-type: none"> • hardware • status • position • denomination • deviceId • name • denominationValue • cycleNumber • dispenseLimit • remaining • dispensed • deposited • lastTransactionDispensed • lastTransactionDeposited • loaded • diverted1 • diverted2 • retracted • checkDispensed • checkRetracted • checkLoaded • reactivated 	List of cassettes (denomination types, counters).
cassetteAmounts	java.util.List<com.openwaygroup.ts.fp.shared.Amount>		Cassette amounts.
protocol	com.openwaygroup.ts.fp.shared.atm.protocol		Protocol type ("NDC+", "MDS912").
collectionCycle	java.lang.Long		ID of the current collection cycle (for inclusion in the replenishment officer's receipt).
cashinData	java.util.List<com.openwaygroup.ts.fp.shared.atm.message.BNARec>	<ul style="list-style-type: none"> • amount • notes <ul style="list-style-type: none"> • noteType • denomination • count 	Amount of accepted notes with a list of data by type of notes (note type (NDC), denomination (DC), quantity).
biller	java.lang.String		Service Provider name.

Name	Type	Fields	Comments
numRC	java.lang.String		Numeric representation of response code.
srn	java.lang.String		Internal transaction ID (SRN).
transactionDescription	java.lang.String		Transaction data (for example, phone number for payment).
textDetails	java.lang.String		Data string (used in billing payments).
minDenomination	com.openwaygroup.ts.fp.shared.Amount		Minimal face value of notes in the ATM.
maxDenomination	com.openwaygroup.ts.fp.shared.Amount		Maximal face value of notes in the ATM.
dispenserId	java.lang.Int		Dispenser Identifier.
maxAmounts	java.util.List<com.openwaygroup.ts.fp.shared.Amount>		Maximum withdrawal amount by currency.
settlement	com.openwaygroup.ts.fp.shared.Amount		Settlement currency and amount.
settlement2	com.openwaygroup.ts.fp.shared.Amount		Settlement currency and amount (for second dispenser).
fee2	com.openwaygroup.ts.fp.shared.Amount		Fee amount and currency (for second dispenser).
additionalBalances	java.util.List<com.openwaygroup.ts.atm.controller.handler.support.AdditionalBalance>	<ul style="list-style-type: none"> accountName amount 	List of additional account balances (amounts and currencies)
cashOut	java.lang.Boolean		Indicator of cassettes data present.
cashOut2	java.lang.Boolean		Indicator of cassettes data present (for second dispenser).

Name	Type	Fields	Comments
replCycle	java.lang.Int		ID of the current closed cycle (for inclusion in the replenishment officer's receipt).
replCycle2	java.lang.Int		ID of the current closed cycle (for inclusion in the replenishment officer's receipt for second dispenser).
cassettes2	java.util.List<com.openwaygroup.ts.fp.shared.atm.device.Cassette>	Same as "cassettes"	List of cassettes (denomination types, counters for second dispenser).
cassetteAmounts2	java.util.List<com.openwaygroup.ts.fp.shared.Amount>		Cassette amounts.
cassetteTotals	java.util.List<com.openwaygroup.ts.fp.shared.Amount>		Total amount by cassettes in the context of currency.
cassetteTotals2	java.util.List<com.openwaygroup.ts.fp.shared.Amount>		Total amount by cassettes in the context of currency (for second dispenser).
otpList	java.util.List<String>		List of one-time passwords.
userID	java.lang.String		Login to access remote banking services.
password	java.lang.String		Password to access remote banking services.
prePaid	java.util.List<com.openwaygroup.ts.atm.controller.handler.template.model.PrePaidData>	<ul style="list-style-type: none"> code serialNumber count expiry 	Scratch card data.
pickList	java.util.Map<java.lang.String, java.lang.String>		Associative array for selecting an account from the list of accounts.
paymentCode	java.lang.String		Service Code.

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
```

Sample receipt template:

```

<#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">
Ledge:      ${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
        ATM: ${lun0}
        Date:      Time:      card:
        ${time?datetime} ${card[0..*6]}..${card[card?length - 1..]}
        ${ESC}(2
        <#if operation == "CASH_WITHDRAWAL" && (amount.amount)?>
        AMOUNT WITHDRAWN: ${amount.amount?left_pad(12)} <#if (amount.currency)?>${
        {amount.currency.name}</#if>
        </#if>
        <#if amounts??>
            <#list amounts as amt>
                <#switch amt.amountType>
                    <#case "01">
current      ${ESC}(2: ${amt.amount?left_pad(12)} ${amt.currency.name}
                    <#break>
                    <#case "02">
available    ${ESC}(2: ${amt.amount?left_pad(12)} ${amt.currency.name}
                    <#break>
                    <#case "91">
credit      ${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 the section "[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.

3.3.1 Encoding configuration

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.EncoderType
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",
"Ě": "t",
"ě": "t",
"A": "f",
"Б": "~",
"B": "d",
"Г": "u",
"Д": "l",
"E": "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 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",
"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",
"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",
"Ė": "t",
"ė": "t",
"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",
        "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 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": "j",
        "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": "j",
        "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).a
dd(rus).add(en2).add(all)
    // load to encoder store
    addEncoder(encoder1)
    addEncoder(encoder2)
    addEncoder(encoder3)
}

```

3.3.2 Templates testing

It is recommended to test new templates before applying. It can be made by calling the `testTemplate` command in the management console (see the document "Web Console for Managing Way4 Applications"). The path to xml-file with data for substituting into the template should be specified as a command parameter. As a result of executing the command, if the template is working correctly, the text generated from the data file is output; otherwise, an error message is displayed.

Sample of xml-file with data for template testing (the name can be arbitrary, for example, `template_test_data.xml`):

```

<TemplateData _sys_type_id="atm.TemplateData" _ver="0" applicationId="app_id"
applicationLabel="VISA" authCode="987654" biller="biller" card="1000040163547882"
card2="1000040163547883" cardName="CARD NAME" cardsRetained="11" cashOut="true"
cashOut2="true" collectionCycle="11" date="2019-02-26 13:04:00.0" dispenserId="1"
encoderType="PRINTER" lang="RU" maxSize="0" numRC="0" operation="CASH_WITHDRAWAL"
password="password" protocol="NDC" rc="00" replCycle="9" replCycle2="11" rrn="111222
333" srn="J05732HN0TD2" stan="012345" templateName="admin_dual" terminalId="ATM00000
1" textDetails="text details" transactionDescription="tran description" userId="user
_id" vendor="NCR">
  <additionalBal _sys_type_id="_al">
    <item _sys_type_id="atm.AdditionalBalance" _ver="0" accountName="001-
P-977320">
      <amount _sys_type_id="ext_amnt" _ver="0" accountType="00" amount="1000.00"
amountType="01">
        <currency>EUR</currency>
      </amount>
    </item>
    <item _sys_type_id="atm.AdditionalBalance" _ver="0" accountName="001-
P-977320">
      <amount _sys_type_id="ext_amnt" _ver="0" accountType="00" amount="1000.00"
amountType="02">
        <currency>EUR</currency>
      </amount>
    </item>
    <item _sys_type_id="atm.AdditionalBalance" _ver="0" accountName="001-
P-848060">
      <amount _sys_type_id="ext_amnt" _ver="0" accountType="00" amount="1000.00"
amountType="01">
        <currency>USD</currency>
      </amount>
    </item>
    <item _sys_type_id="atm.AdditionalBalance" _ver="0" accountName="001-
P-848060">
      <amount _sys_type_id="ext_amnt" _ver="0" accountType="00" amount="1000.00"
amountType="02">
        <currency>USD</currency>
      </amount>
    </item>
    <item _sys_type_id="atm.AdditionalBalance" _ver="0" accountName="001-
P-411950">
      <amount _sys_type_id="ext_amnt" _ver="0" accountType="00" amount="1000.00"
amountType="01">
        <currency>RUR</currency>
      </amount>
    </item>
    <item _sys_type_id="atm.AdditionalBalance" _ver="0" accountName="001-
P-411950">
      <amount _sys_type_id="ext_amnt" _ver="0" accountType="00" amount="-2000.00"
amountType="02">
        <currency>RUR</currency>
      </amount>
    </item>
  </additionalBal>
</amounts _sys_type_id="_al">

```

```

    <item _sys_type_id="ext_amnt" _ver="0" accountType="01" amount="100.00"
amountType="02">
    <currency>RUR</currency>
    </item>
</amounts>
<cashinData _sys_type_id="_al">
    <item _sys_type_id="atm.BNARec" _ver="0">
    <amount _ver="0" amount="150.00">
    <currency>RUR</currency>
    </amount>
    <notes _sys_type_id="_al">
    <item _sys_type_id="atm.BNANote" _ver="0" count="1" denomination="50"/>
    <item _sys_type_id="atm.BNANote" _ver="0" count="1" denomination="100"/>
    </notes>
    </item>
    <item _sys_type_id="atm.BNARec" _ver="0">
    <amount _ver="0" amount="2000.00">
    <currency>RUR</currency>
    </amount>
    <notes _sys_type_id="_al">
    <item _sys_type_id="atm.BNANote" _ver="0" count="1" denomination="1000"/
>
    <item _sys_type_id="atm.BNANote" _ver="0" count="2" denomination="500"/>
    </notes>
    </item>
</cashinData>
<cassetteAmounts _sys_type_id="_al">
    <item _ver="0" amount="750.00">
    <currency>RUR</currency>
    </item>
    <item _ver="0" amount="2000.00">
    <currency>RUR</currency>
    </item>
    <item _ver="0" amount="250000.00">
    <currency>RUR</currency>
    </item>
    <item _ver="0" amount="100000.00">
    <currency>RUR</currency>
    </item>
</cassetteAmounts>
<cassetteAmounts2 _sys_type_id="_al">
    <item _ver="0" amount="5.00">
    <currency>USD</currency>
    </item>
    <item _ver="0" amount="20.00">
    <currency>USD</currency>
    </item>
    <item _ver="0" amount="150.00">
    <currency>USD</currency>
    </item>
    <item _ver="0" amount="400.00">
    <currency>USD</currency>
    </item>
</cassetteAmounts2>
<cassetteTotals _sys_type_id="_al">
    <item _ver="0" amount="352750.00">

```



```

        <currency>RUR</currency>
    </item>
</cassetteTotals>
<cassetteTotals2 _sys_type_id="_al">
    <item _ver="0" amount="575.00">
        <currency>USD</currency>
    </item>
</cassetteTotals2>
<cassettes _sys_type_id="_al">
    <item _sys_type_id="atm.Cassette" _ver="0" checkDispensed="0" checkLoaded="0"
checkRetracted="0" cycleNumber="9" denomination="1"
        deposited="0" dispensed="15" diverted1="0" diverted2="0"
lastTransactionDeposited="0" lastTransactionDispensed="0"
        loaded="1000" name="DBP1" position="1" remaining="0" retracted="0">
        <denominationValue _ver="0" amount="50.00">
            <currency>RUR</currency>
        </denominationValue>
        <dispenseLimit _sys_type_id="atm.DispenseLimit" _ver="0" limit="40">
            <penalty _sys_type_id="_hm"/>
        </dispenseLimit>
    </item>
    <item _sys_type_id="atm.Cassette" _ver="0" checkDispensed="0" checkLoaded="0"
checkRetracted="0" cycleNumber="9" denomination="2"
        deposited="0" dispensed="20" diverted1="0" diverted2="0"
lastTransactionDeposited="0" lastTransactionDispensed="0"
        loaded="1000" name="DBP2" position="2" remaining="0" retracted="0">
        <denominationValue _ver="0" amount="100.00">
            <currency>RUR</currency>
        </denominationValue>
        <dispenseLimit _sys_type_id="atm.DispenseLimit" _ver="0" limit="40">
            <penalty _sys_type_id="_hm"/>
        </dispenseLimit>
    </item>
    <item _sys_type_id="atm.Cassette" _ver="0" checkDispensed="0" checkLoaded="0"
checkRetracted="0" cycleNumber="9" denomination="3"
        deposited="0" dispensed="50" diverted1="0" diverted2="0"
lastTransactionDeposited="0" lastTransactionDispensed="0"
        loaded="1000" name="DBP3" position="3" remaining="0" retracted="0">
        <denominationValue _ver="0" amount="5000.00">
            <currency>RUR</currency>
        </denominationValue>
        <dispenseLimit _sys_type_id="atm.DispenseLimit" _ver="0" limit="40">
            <penalty _sys_type_id="_hm"/>
        </dispenseLimit>
    </item>
    <item _sys_type_id="atm.Cassette" _ver="0" checkDispensed="0" checkLoaded="0"
checkRetracted="0" cycleNumber="9" denomination="4"
        deposited="0" dispensed="100" diverted1="0" diverted2="0"
lastTransactionDeposited="0" lastTransactionDispensed="0"
        loaded="2000" name="DBP4" position="4" remaining="0" retracted="0">
        <denominationValue _ver="0" amount="1000.00">
            <currency>RUR</currency>
        </denominationValue>
        <dispenseLimit _sys_type_id="atm.DispenseLimit" _ver="0" limit="40">
            <penalty _sys_type_id="_hm"/>
        </dispenseLimit>
    </item>
</cassettes>

```

```

    </item>
  </cassettes>
  <cassettes2 _sys_type_id="_al">
    <item _sys_type_id="atm.Cassette" _ver="0" checkDispensed="0" checkLoaded="0"
checkRetracted="0" cycleNumber="11"
      denomination="1" deposited="0" dispensed="1" diverted1="0" diverted2="0"
      lastTransactionDeposited="0"
      lastTransactionDispensed="0" loaded="1000" name="DI02DBP1" position="1"
      remaining="0" retracted="0">
      <denominationValue _ver="0" amount="5.00">
        <currency>USD</currency>
      </denominationValue>
      <dispenseLimit _sys_type_id="atm.DispenseLimit" _ver="0" limit="40">
        <penalty _sys_type_id="_hm"/>
      </dispenseLimit>
    </item>
    <item _sys_type_id="atm.Cassette" _ver="0" checkDispensed="0" checkLoaded="0"
checkRetracted="0" cycleNumber="11"
      denomination="2" deposited="0" dispensed="2" diverted1="0" diverted2="0"
      lastTransactionDeposited="0"
      lastTransactionDispensed="0" loaded="1000" name="DI02DBP2" position="2"
      remaining="0" retracted="0">
      <denominationValue _ver="0" amount="10.00">
        <currency>USD</currency>
      </denominationValue>
      <dispenseLimit _sys_type_id="atm.DispenseLimit" _ver="0" limit="40">
        <penalty _sys_type_id="_hm"/>
      </dispenseLimit>
    </item>
    <item _sys_type_id="atm.Cassette" _ver="0" checkDispensed="0" checkLoaded="0"
checkRetracted="0" cycleNumber="11"
      denomination="3" deposited="0" dispensed="3" diverted1="0" diverted2="0"
      lastTransactionDeposited="0"
      lastTransactionDispensed="0" loaded="1000" name="DI02DBP3" position="3"
      remaining="0" retracted="0">
      <denominationValue _ver="0" amount="50.00">
        <currency>USD</currency>
      </denominationValue>
      <dispenseLimit _sys_type_id="atm.DispenseLimit" _ver="0" limit="40">
        <penalty _sys_type_id="_hm"/>
      </dispenseLimit>
    </item>
    <item _sys_type_id="atm.Cassette" _ver="0" checkDispensed="0" checkLoaded="0"
checkRetracted="0" cycleNumber="11"
      denomination="4" deposited="0" dispensed="4" diverted1="0" diverted2="0"
      lastTransactionDeposited="0"
      lastTransactionDispensed="0" loaded="2000" name="DI02DBP4" position="4"
      remaining="0" retracted="0">
      <denominationValue _ver="0" amount="100.00">
        <currency>USD</currency>
      </denominationValue>
      <dispenseLimit _sys_type_id="atm.DispenseLimit" _ver="0" limit="40">
        <penalty _sys_type_id="_hm"/>
      </dispenseLimit>
    </item>
  </cassettes2>

```

```

<dispense _sys_type_id="_al">
  <item _sys_type_id="atm.DispenseValue" _ver="0" amount="10000" count="1"
denomination="1" denominationValue="10000"
  dispenserId="DBP1" limit="0" maxCount="0" penalty="0" penaltyIncrease="0
" positionId="1"/>
</dispense>
<fee _ver="0" amount="1000">
  <currency>RUR</currency>
</fee>
<fee2 _ver="0" amount="1000">
  <currency>RUR</currency>
</fee2>
<info _sys_type_id="_hm">
  <entry key="info1">test</entry>
</info>
<location _ver="0" city="City">
  <country>RUS</country>
</location>
<maxAvailAmounts _sys_type_id="_al">
  <item _ver="0" amount="200000.00">
    <currency>RUR</currency>
  </item>
  <item _ver="0" amount="4000.00">
    <currency>USD</currency>
  </item>
</maxAvailAmounts>
<maxDenomination _ver="0" amount="5000.00">
  <currency>RUR</currency>
</maxDenomination>
<minDenomination _ver="0" amount="50.00">
  <currency>RUR</currency>
</minDenomination>
<otplList _sys_type_id="_al">
  <item>111111</item>
  <item>222222</item>
</otplList>
<params _sys_type_id="_hm">
  <entry key="param1">value1</entry>
</params>
<request _ver="0" amount="1000">
  <currency>RUR</currency>
</request>
<settlement _ver="0" amount="1000">
  <currency>RUR</currency>
</settlement>
<settlement2 _ver="0" amount="1000">
  <currency>RUR</currency>
</settlement2>
<statement _sys_type_id="_al">
  <item _sys_type_id="atm.miniStatementProxy" _ver="0" date="2013-10-08
00:00:00.0" operationType="A">
    <amount _ver="0" amount="-13.00">
      <currency>RUR</currency>
    </amount>
  </item>

```

```

    <item _sys_type_id="atm.miniStatementProxy" _ver="0" date="2013-10-08
00:00:00.0" operationType="A">
      <amount _ver="0" amount="-34.00">
        <currency>RUR</currency>
      </amount>
    </item>
    <item _sys_type_id="atm.miniStatementProxy" _ver="0" date="2013-10-08
00:00:00.0" operationType="A">
      <amount _ver="0" amount="-25.00">
        <currency>RUR</currency>
      </amount>
    </item>
    <item _sys_type_id="atm.miniStatementProxy" _ver="0" date="2013-10-08
00:00:00.0" operationType="A">
      <amount _ver="0" amount="-17.00">
        <currency>RUR</currency>
      </amount>
    </item>
  </statement>
</TemplateData>

```

Most of the fields that may be present in templates are filled in this example. For testing, some of the fields may not be filled.

Service fields in the test xml-file (not used in templates):

- **templateName** – name of the template to compile.
- **encoderType** – (PRINTER/SCREEN) encoding type for compiling template. If the field is empty, the encoding of the output text is not performed.
- **maxSize** – maximum size (in characters) of the output text after compilation and encoding. If the field is empty or contains 0, the text is not cropped.
- **protocol** – (NDC/DDC) encoding type for compiling template. If the field is empty, the encoding of the output text is not performed.
- **vendor** – ATM type (for example NCR3G or NCR). The field determines encoding type when template compiling. If the field is empty, the encoding of the output text is not performed.

Also, an example of the test xml-file content can be obtained from the ATM controller's log when logging with level 60.

4 Settlement scheme for ATM operations

During ATM operation (cash dispense/acceptance, collection operations) accounting entries are generated in Way4 that reflect cash flows.

An ATM contract's Account 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: "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 Account 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.

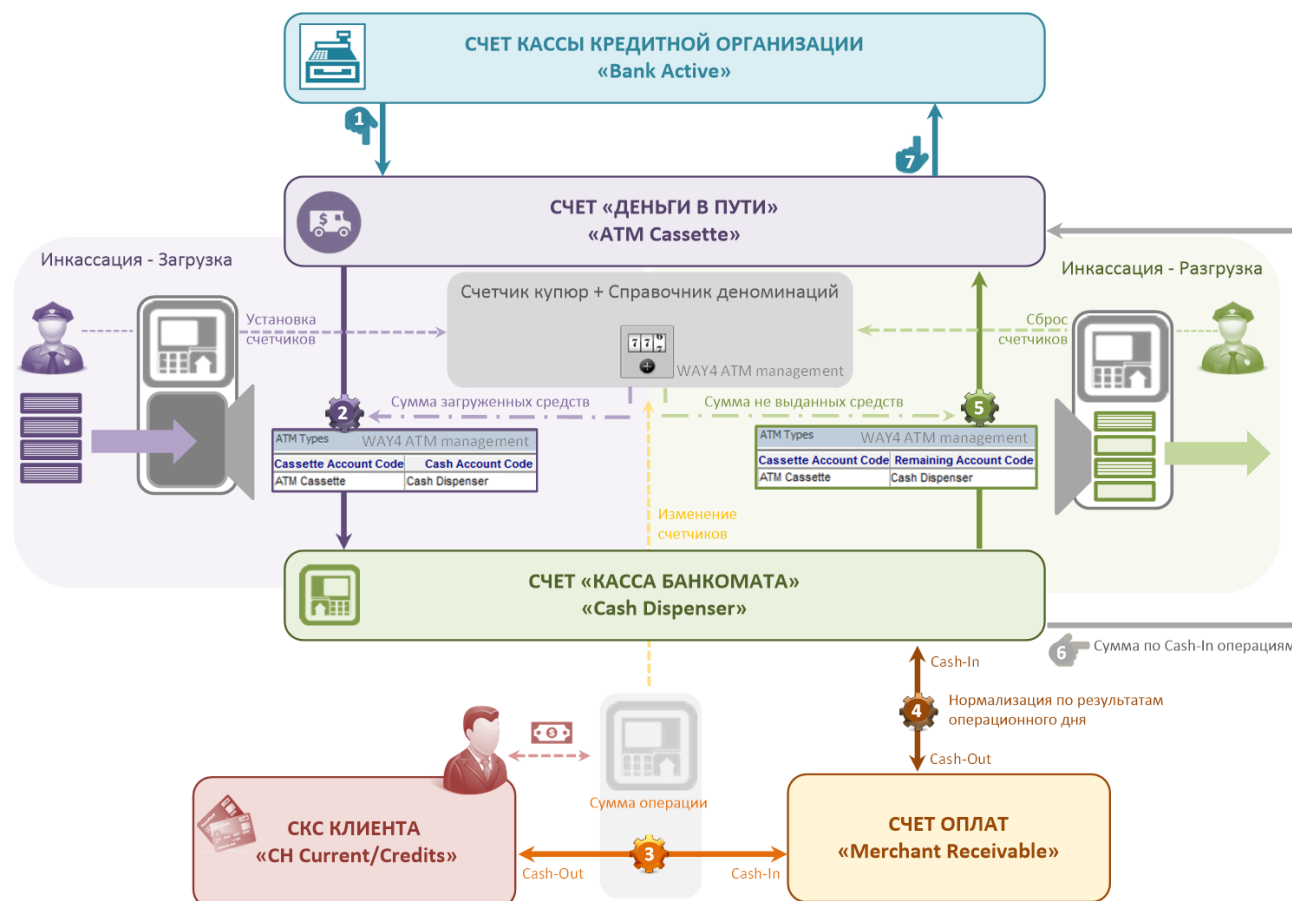
If ATM cash acceptance and dispensing operations are supported, two separate types of account must be used in the Account Scheme; for example, "Merchant Receivable In" and "Merchant Receivable Out" to show funds that have been accepted from and dispensed to clients during the business day.

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 Account 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:



Settlement scheme for ATM operations

Step 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 created between the following types of account:

- Dt "ATM Cassette" Ct "Bank Active"

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



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.

Step 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 the section "[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 the section "[ATM types dictionary](#)"):

- Dt "Cash Dispenser" Ct "ATM Cassette"

Step 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.

Step 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.



The "ENTRY_GROUPING" tag can be used to show cash dispensing and cash acceptance turnover in a "Merchant Receivable" account for a financial cycle. For this, in the appropriate Account Scheme (menu item "Full → Configuration Setup → Products → Acquiring Products → Acquiring Account Schemes") specify the value "ENTRY_GROUPING=BY_BATCH;" in the *Template Details* field for the "Merchant Receivable" account.

Step 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 Code* and *Code Account Remaining Account* fields for the corresponding ATM type (see the section "[ATM types dictionary](#)"):

- Dt "ATM Cassette" Ct "Cash Dispenser"

Step 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"

Step 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.

5 Automatic reversal message creation

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



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: if there is no response from the authorization channel within a specified time (50 sec).
- In stage 5 of the operation: if the operation was not performed for technical reasons.
- In stage 6 of the operation: 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: if a corresponding status message is received, for example, that the amount selected has not been dispensed to the cardholder.

The ATM controller enables to configure the custom response codes (RC) for use in reversal messages. Rules that define these codes are set in the CustomReversals.groovy script, which should be placed in the WEB-INF/inv/atm/scripts directory. To configure rules, use the CustomReversal object, which contains the following parameters:

- ReversalType – reversal type. Possible values:
 - TIMEOUT_REVERSAL
 - CANCELLATION

- UNEXPECTED_ERROR_REVERSAL
- REVERSAL_WITH_ADJUST
- CANCELLATION_WITH_ADJUST
- FULL_REVERSAL
- PARTIAL_REVERSAL
- OperationType – type of operation to reverse (optional parameter).
- Channel – code of the channel for card payment system (optional parameter).
- OperStatus – response code. Possible OperStatus values used by the ATM controller and corresponding RC values:

OperStatus	RC	Description
SUCCESS	0	Successfully completed
SUCCESS_BY_SAF	0	Successfully completed by SAF
REFER_TO_CARD_ISSUER	1	Refer to card issuer
REFER_TO_CARD_ISSUERS_SPECIAL_CONDITION	2	Refer to card issuer's special condition
INVALID_MERCHANT_SOURCE	3	Invalid merchant / source
PICK_UP	4	PICK UP
DO_NOT_HONOUR	5	Do not Honour
OP_ERROR	6	Error
PICK_UP_CARD_SPECIAL_CONDITION	7	Pick-up card, special condition
HONOUR_WITH_IDENTIFICATION	8	Honour with identification
REQUEST_IN_PROGRESS	9	Request in progress
APPROVED_FOR_PARTIAL_AMOUNT	10	Approved for partial amount
APPROVED_VIP	11	Approved (VIP)
INVALID_TRANSACTION	12	Invalid transaction

OperStatus	RC	Description
INVALID_AMOUNT	13	Invalid amount
NO_SUCH_CARD	14	No such card
NO_SUCH_ISSUER	15	No such issuer
APPROVED_UPDATE_TRACK_3	16	Approved, update track 3
CUSTOMER_CANCELLATION	17	Customer cancellation
CUSTOMER_DISPUTE	18	Customer dispute
RE_ENTER_TRANSACTION	19	Re-enter transaction
INVALID_RESPONSE	20	Invalid response
NO_ACTION_TAKEN	21	No action taken
SUSPECTED_MALFUNCTION	22	Suspected malfunction
UNACCEPTABLE_TRANSACTION_FEE	23	Unacceptable transaction fee
FILE_UPDATE_NOT_SUPPORTED_BY_RECEIVER	24	File update not supported by receiver
NO_SUCH_RECORD	25	No such record
DUPLICATE_RECORD_UPDATE_OLD_RECORD_REPLACED	26	Duplicate record update, old record replaced
FILE_UPDATE_FIELD_EDIT_ERROR	27	File update field edit error
FILE_LOCKED_OUT_WHILE_UPDATE	28	File locked out while update
FILE_UPDATE_ERROR_CONTACT_ACQUIRER	29	File update error, contact acquirer
FORMAT_ERROR	30	Format error
ISSUER_SIGNED_OFF	31	Issuer signed-off

OperStatus	RC	Description
COMPLETED_PARTIALLY	32	Completed partially
PICK_UP_EXPIRED_CARD	33	Pick-up, expired card
SUSPECT_FRAUD	34	Suspect Fraud
PICK_UP_CARD_ACCEPTOR_CONTACT_ACQUIRER	35	Pick-up, card acceptor contact acquirer
PICK_UP_CARD_RESTRICTED	36	Pick up, card restricted
PICK_UP_CALL_ACQUIRER_SECURITY	37	Pick up, call acquirer security
PICK_UP_ALLOWABLE_PIN_TRIES_EXCEEDED	38	Pick up, Allowable PIN tries exceeded
NO_CREDIT_ACCOUNT	39	No credit account
REQUESTED_FUNCTION_NOT_SUPPORTED	40	Requested function not supported
PICK_UP_LOST_CARD	41	Pick up, lost card
NO_UNIVERSAL_ACCOUNT	42	No universal account
PICK_UP_STOLEN_CARD	43	Pick up, stolen card
NO_INVESTMENT_ACCOUNT	44	No investment account
DO_NOT_RENEW	50	Do not renew
NOT_SUFFICIENT_FUNDS	51	Not sufficient funds
NO_CHEQUING_ACCOUNT	52	No checking account
NO_SAVINGS_ACCOUNT	53	No savings account
EXPIRED_CARD	54	Expired card / target
INCORRECT_PIN	55	Incorrect PIN

OperStatus	RC	Description
NO_CARD_RECORD	56	No card record
TRANSACTION_NOT_PERMITTED	57	Transaction Not Permitted
TRANSACTION_NOT_PERMITTED_TO_TERMINAL	58	Transaction not permitted to terminal
SUSPECTED_FRAUD	59	Suspected fraud
CARD_ACCEPTOR_CONTACT_ACQUIRER	60	Card acceptor contact acquirer
EXCEEDS_WITHDRAWAL_AMOUNT_LIMIT	61	Exceeds withdrawal amount limit
RESTRICTED_CARD	62	Restricted card
SECURITY_VIOLATION	63	Security violation
WRONG_ORIGINAL_AMOUNT	64	Wrong original amount
EXCEEDS_WITHDRAWAL_FREQUENCY_LIMIT	65	Exceeds withdrawal frequency limit
CALL_ACQUIRERS_SECURITY_DEPARTMENT	66	Call acquirers security department
CARD_TO_BE_PICKED_UP_AT_ATM	67	Card to be picked up at ATM
RESPONSE_RECEIVED_TOO_LATE	68	Response received too late
INVALID_TRANSACTION_CONTACT_CARD_ISSUER	70	Invalid transaction; contact card issuer
DECLINE_PIN_NOT_CHANGED	71	Decline PIN not changed
ALLOWABLE_NUMBER_OF_PIN_TRIES_EXCEEDED	75	Allowed number of PIN tries exceeded
WRONG_PIN_NUMBER_OF_PIN_TRIES_EXCEEDED	76	Wrong PIN, number of PIN tries exceeded
WRONG_REFERENCE_NO	77	Wrong Reference No.
RECORD_NOT_FOUND	78	Record Not Found

OperStatus	RC	Description
ALREADY_REVERSED	79	Already reversed
NETWORK_ERROR	80	Network error
FOREIGN_NETWORK_ERROR_PIN_CRYPTOGRAPHIC_ERROR	81	Foreign network error / PIN cryptographic error
TIMEOUT	82	Time-out at issuer system
TRANSACTION_FAILED	83	Transaction failed
PRE_AUTHORIZATION_TIMED_OUT	84	Pre-authorization timed out
NO_REASON_TO_DECLINE	85	No Reason To Decline
UNABLE_TO_VALIDATE_PIN	86	Unable to validate PIN
PURCHASE_APPROVAL_ONLY	87	Purchase Approval Only
CRYPTOGRAPHIC_FAILURE	88	Cryptographic failure
AUTHENTICATION_FAILURE	89	Authentication failure
CUTOFF_IS_IN_PROGRESS	90	Cutoff is in progress
ISSUER_OR_SWITCH_IS_INOPERATIVE	91	Issuer or switch is inoperative
UNABLE_TO_ROUTE_AT_ACQUIRER_MODULE	92	Unable to route at acquirer module
CANNOT_BE_COMPLETED_VIOLATION_OF_LAW	93	Cannot be completed, violation of law
DUPLICATE_TRANSMISSION	94	Duplicate Transmission
RECONCILE_ERROR_AUTH_NOT_FOUND	95	Reconcile error / Auth Not found
SYSTEM_MALFUNCTION	96	System Malfunction
CREDIT_FAILED	145	Credit authorization declined

OperStatus	RC	Description
ROUTING_UNAVAILABLE	2015	HA: no available DB routes

One can also define an additional custom response code using the OperStatus object, for example, as follows:

```
new OperStatus(OperStatus.ResponseClass.ERROR, 34L, "Fraud message")
```

Example of CustomReversals.groovy:

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

import com.openwaygroup.ts.atm.controller.handler.operations.CustomReversal
import com.openwaygroup.ts.atm.controller.handler.operations.ReversalType
import com.openwaygroup.ts.fp.shared.OperStatus
import com.openwaygroup.ts.fp.shared.atm.scripts.OperationType

[
    new CustomReversal(ReversalType.CANCELLATION_WITH_ADJUST,
        OperationType.PIN_CHANGE, "E", OperStatus.OP_ERROR),
    new CustomReversal(ReversalType.CANCELLATION_WITH_ADJUST, null, null,
        OperStatus.OP_ERROR)
    new CustomReversal(ReversalType.TIMEOUT_REVERSAL,
        OperationType.CARD_CTRL_REQUEST, "E", new OperStatus(OperStatus.ResponseClass.ERROR,
            168L, "Response received too late"))
]
```

The list is processed in the order of the rules, so the first rule that matches the criteria for the current reversal will be applied.