Cycle API

Integration Guide

V 2.1.0

History of changes

Version	Data	Description
2.1.0	06.02.2025	General improvements

Table of contents

Disclaimer

The developers of the following payment module (SDK) and service Cycle provide the following method for the deferred authorization with an explicit disclaimer of their responsibility for the results of deferred authorization of bank card payment transactions. By saving the data package for further authorization on the side of payment solution (software) developed by the partner or client using this payment module (SDK), the developer assumes the responsibility for informing the end customer about the possible refusal of deferred authorization by the issuing bank for various reasons, such as (included, but not limited to): insufficient funds, card blocking, expired cryptogram in the data package and other reasons for the authorization refusal.

The responsibility for the goods issue or service render through deferred authorization and not receiving reimbursement for such transactions due to deferred authorization which ended with the refusal of the issuing bank, is entirely on the end user's side. The method of obtaining a hashed card number for maintaining cards' stop-lists (backlists) just provides an opportunity to prevent the second and subsequent card sales which were previously declined. Maintenance of such stop-lists (blacklists) should be implemented by the developer of the payment solution (software) independently. The developer should not rely on the fact that such functionality will be provided by the Cycle service.

Android Permissions

Android API 19 is the minimum level supported. Before working with a library, the following strings should be added to the file **AndroidManifest.xml** :

```
<uses-permission android:name=”android.permission.INTERNET” />
<uses-permission android:name=”android.permission.ACCESS_NETWORK_STATE” />
<uses-permission android:name=”android.permission.READ_PHONE_STATE” />
<uses-permission android:name=”android.permission.ACCESS_COARSE_LOCATION” />
<uses-permission android:name=”android.permission.ACCESS_FINE_LOCATION” />
<uses-permission android:name=”android.permission.BLUETOOTH” />
<uses-permission android:name=”android.permission.BLUETOOTH_ADMIN” />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
```

SDK Integration

To integrate SDK into an Android project, it is necessary to add a reference to maven repository in build.gradle of the project:

```
allprojects {
   repositories {
      google()
      jcenter()
      maven { url 'https://maven.cyclebitgroup.com' }
   }
}
```

And to add a dependency in build.gradle of the app that uses SDK:

```
dependencies {
   implementation 'com.mpos:sdk:2.1.0'
   ....
}
```

Package com.mpos.sdk

Class PaymentController

The following class is the central in the library. It contains methods for creating transactions and passing additional parameters to them, as well as encapsulating the operation with card readers. The class contains sets of various parameters in the form of **enum** that are required to make a payment.

Before conducting transactions, it is required to set the user email and password that are necessary for authentication using the **setCredentials** method and set the reader type using the **setReaderType** type. It is also recommended to call **initPaymentSession** before each payment session to initialize the internal counters.

Before working with card readers, it is necessary to call a method **enable** and call a method **disable** after you finish. Method **disable** or disabling the card reader will interrupt the processing of the ongoing transaction. It is required to wait for event **ReaderEvent.INIT_SUCCESSFULLY** before starting the transaction.

It is also necessary to call identically titled methods of the class instance when the following methods are called in the parent **Activity**: **onCreate**, **onDestroy**, **onSaveInstanceState, onNewIntent**. The timeout for card retrieval in case of transaction failure is 5 seconds.

To process card reader events and/or transaction execution, **PaymentControllerListener** may be passed to an instance of the class using the method **setPaymentControllerListener**.

When specifying a payment amount with a greater decimal place than the decimal place of the currency, the number of digits after the decimal point will be reduced **without rounding**.

Sets of parameters:

Static properties

Configuration PaymentController

Property	**Description**
DEBUG	The indicator of operation in debugging mode
VERSIONCODE	The version of the library
MAX_EMV_RETRIES	The number of failed attempts of conducting the transaction with a chip required to authorize the use of magnetic tape

CONNECTION_LOST_RETIRES	The number of attempts to confirm a payment when the Internet connection is lost. Only in case CONNECTION_LOST_TIMEOUT = 0
CONNECTION_LOST_TIMEOUT	Waiting time of payment confirmation in case the connection is lost

ReaderType

A set of supported card reader types

Type	Description
DSPREAD D60	Device Dspread D60

ReaderEvent

A set of possible events that can be transmitted by the card reader

Type	Description
CONNECTED	The card reader is connected
DISCONNECTED	The card reader is disconnected
START_INIT	The beginning of initialization
INIT_SUCCESSFULLY	Successful initialization
INIT_FAILED	Initialization error
EJECT_CARD_TIMEOUT	The device is not used
SWIPE_CARD	The magnetic stripe conduction has been detected
EMV_TRANSACTION_STARTED	The chip transaction has been initiated
NFC_TRANSACTION_STARTED	The NFC transaction has been initiated
WAITING_FOR_CARD	Waiting for magnetic stripe swipe or chip card insertion
PAYMENT_CANCELED	The transaction is canceled by the user
EJECT_CARD	The user may eject the card (when a transaction error occurs)
BAD_SWIPE	The attempt to encode or read the magnetic stripe on the card has failed
LOW_BATTERY	The battery charge of the card reader is lower than 10%
CARD_TIMEOUT	The card waiting timeout exceeded

CARD_INFO_RECEIVED	The card information has been received
PIN_TIMEOUT	PIN entry waiting timeout exceeded

PaymentInputType

A set of possible payment types

Type	Description
SWIPE	Magnetic stripe card payment
CHIP	Card chip payment
NFC	NFC payments
Type	**Description**
PREPAID	Prepayment
CREDIT	Credit payment
CASH	Cash payment
OTHER	Payment via link
OUTER_CARD	POS-terminal payment

PaymentError

A set of possible errors that may occur during transaction execution

Type	Description
CONNECTION_ERROR	Error occurred while trying to connect to servers
SERVER_ERROR	Error occurred while trying to make a transaction
TRANSACTION_NULL_OR_EMPTY	Error occurred while trying to create a transaction
NO_SUCH_TRANSACTION	The transaction could not be found or was not unique
EMV_ERROR	EMV error
EMV_TERMINATED	The transaction was terminated
EMV_DECLINED	The transaction was declined
EMV_CANCEL	The transaction was canceled
EMV_CARD_ERROR	Card error
EMV_DEVICE_ERROR	Reader error
EMV_CARD_NOT_SUPPORTED	The card type is not supported
EMV_NOT_ALLOWED	Chip transaction is not allowed
EMV_ZERO_TRAN_EMV	The attempt to make a zero-value transaction was made
NFC_NOT_ALLOWED	NFC transaction is not allowed
INVALID_AMOUNT	The refund/cancellation payment amount exceeds the balance of the transaction
STANDALONE_FAILED	An error occurred while making a payment via TTK protocol
NFC_LIMIT_EXCEEDED	The contactless payment limit was exceeded
TTK_FAILED	The payment error via external application has occurred
EXT_APP_FAILED	The payment error via external application has occurred
SWIPE_NOT_ALLOWED	The magnetic stripe card payment is not supported
RESUBMIT_FAILED	An error occurred while trying to confirm a transaction
DEFERRED_FAILED	An error occurred while trying to generate data for a deferred transaction
INTERNAL_ERROR	An unforeseen error occurred while trying to execute a transaction

RegularRepeatType

A set of possible types of regular payments

Type	Description
Never	The payment will be executed once
Weekly	The payment will be executed weekly
Monthly	The payment will be executed monthly
Quarterly	The payment will be executed quarterly
Type	**Description**
Annual	The payment will be executed annually
ArbitraryDays	The payment will be executed on specified days

RegularEndType

A set of possible ways to stop recurring payments

Type	Description
BY_QUANTITY	By a number of repetitions
BY_DAY	On a specific date

ReverseAction

A set of possible ways to cancel recurring payments

Type	Description
CANCEL	To cancel a payment
RETURN	To make a return

Currency

Currencies available for making a transaction

See Appendix 4

PrintResult

A set of possible outcomes of a print run

Type	Description
SUCCESS	The print job is completed successfully

NO_PAPER	There is no paper in a printer
WRONG_CMD	The command error has occurred
OVERHEAT	The overheating of a printing head
TIMEOUT	Response timeout is exceeded
PRINTER_ERROR	The printer error has occurred

PaymentMethod

A set of possible methods for payment execution

Type	Description
CARD	With a credit card
CREDIT	On credit
CASH	With cash
OTHER	Via payment link
LINKED_CARD	With a linked card
OUTER_CARD	With a POS-terminal
PREPAID	Prepayment

Class methods:

getInstance

Signature	PaymentController getInstance()
Input parameters	No
Return value	Instance
Description	Method for getting an instance of the class

onCreate

Signature	void onCreate(Context context, Bundle savedInstanceState)
Input parameters	context – application context savedInstanceState – is transfered from the method of the parent Activity
Return value	No
Description	Must be called when the identically titled method of the parent Activity is called

onDestroy

Signature	void onDestroy()
Input parameters	No
Return value	No
Description	Must be called when the identically titled method of the parent Activity is called

onSaveInstanceState

Signature	void onSaveInstanceState(Context context, Bundle savedInstanceState)
Input parameters	savedInstanceState – is being transfered from the method of the parent Activity
Return value	No

Description	Must be called when the identically titled method of the parent Activity is called

enable

Signature	void enable()
Input parameters	No
Return value	No
Description	Starts working with a card reader

disable

Signature	void disable()
Input parameters	No
Return value	No
Description	Stops working with a card reader

isConnected

Signature	boolean isConnected()
Input parameters	No
Return value	**true**, if the card reader is connected
Description	Ensures the card reader is ready

getBluetoothDevices

Signature	ArrayList<BluetoothDevice> getBluetoothDevices(Context context)
Input parameters	context – application context
Return value	ArrayList of the paired devices
Description	Is used to obtain a set of Bluetooth devices available for a connection

setPaymentControllerListener

Signature	void setPaymentControllerListener(PaymentControllerListener listener)
Input parameters	listener – event processor
Return value	No
Description	Is used to set a new event processor of making a transaction

setCredentials

Signature	void setCredentials(String email, String password) throws PaymentControllerException
Input parameters	email – user's email password – user's password
Return value	No
Description	Sets the user data required to make transactions. An exception PaymentControllerException will be generated during an attempt to call a method while conducting a transaction.

auth

Signature	APIAuthResult auth(Context context)
Input parameters	context – context of the application
Return value	APIAuthResult
Description	Verifies the accuracy of the user's credentials and returns user's account information

setReaderType

Signature	void setReaderType(Context context, ReaderType readerType, String address, String config) throws PaymentControllerException

Input parameters	context – context of the application. If *readerType == NFC* it is necessary to pass activity readerType – type of the card reader address – MAC-address of the Bluetooth card reader. For USB connection, it is required to pass the constant PaymentController.USB_MODE_KEY config – configuration parameters of the card reader
Return value	No
Description	Changes the type of the current card reader. An exception PaymentControllerException will be generated during an attempt to change the reader type while conducting a transaction.

getReaderType

Signature	ReaderType getReaderType()
Input parameters	No
Return value	Current card reader type
Description	Returns the current card reader type

startPayment

Signature	void startPayment(Context context, PaymentContext paymentContext) throws PaymentException
Input parameters	context – application context paymentContext – transaction information
Return value	No
Description	Starts a transaction. An exception PaymentException will be generated if the payment parameters are incorrect or in case of an attempt to make a new payment/cancel the payment before its completion

reversePayment

Signature	void reversePayment(Context context, String transactionID, ReverseAction action, Double amountToReverse, Currency currency, String receiptPhone, String receiptEmail) throws PaymentException
Input parameters	context – application context transactionID – Transaction ID of the canceled payment action – Cancellation type amountToReverse – cancellation payment amount. For a complete payment cancellation, pass **null** currency – currency that is used for a cancellation/refund receiptPhone – phone number for sending an SMS receipt receiptEmail – email for sending an email receipt
Return value	No
Description	Outdated. See reversePayment(Context context, ReversePaymentContext reversePaymentContext)

reversePayment

Signature	void reversePayment(Context context, ReversePaymentContext reversePaymentContext) throws PaymentException
Input parameters	context – application context reversePaymentContext – data for making a transaction / refund
Return value	No
Description	Initiates the transaction cancellation. An exception PaymentException will be generated in case of an attempt to make a new payment/cancel the payment before its completion

adjust

Signature	APIResult adjust(Context context, String transactionID, String receiptPhone, String receiptEmail, byte [] signature)

Input parameters	context – application context transactionID – transaction ID that requires extra data receiptPhone – phone number for sending an SMS receipt receiptEmail – email for sending an email receipt signature – a picture with a payer's signature
Return value	The result of a data transfer
Description	Is used to send a signature and a receipt for a single payment transaction

adjust

Signature	APIResult adjust(Context context, int regularID, byte [] signature)
Input parameters	context – application context regularID – transaction ID that requires extra data signature – a picture with a payer's signature
Return value	The result of a data transfer
Description	Is used to send a signature and a receipt for recurring payment transactions

adjustReverse

Signature	APIResult adjustReverse(Context String transactionID, String receiptPhone, String receiptEmail, byte [] signature)
Input parameters	context – application context transactionID – transactionID – transaction ID that requires extra data receiptPhone – phone number for sending an SMS receipt receiptEmail – email for sending an email receipt signature – a picture with a payer's signature
Return value	The result of a data transfer
Description	Is used to send a signature and a receipt for a recurring payment transaction

isPaymentInProgress

Signature	boolean isPaymentInProgress()
Input parameters	No
Return value	**true**, if payment processing is not completed
Description	Is used to check the status of the controller

getHistory

Signature	APIGetHistoryResult getHistory(Context context, int page)
Input parameters	context – application context page – page number
Return value	Object APIGetHistoryResult that contains a set of transactions
Description	Enables to get a history of transactions in a paginal format

getTransactionByID

Signature	APIGetHistoryResult getTransactionByID(Context context, String transactionID)
Input parameters	context – application context transactionID – ID of the requested transaction
Return value	Object APIGetHistoryResult that contains the requested transaction
Description	Enables to get a transaction data by its ID

getTransactionByExtID

Signature	APIGetHistoryResult getTransactionByExtID(Context context, String extID)
Input parameters	context – application context extID – extID of the requested transaction
Return value	Object APIGetHistoryResult that contains the requested transaction

Description	Enables to get a transaction data by its extID

getTransactionByRRN

Signature	APIGetHistoryResult getTransactionByRRN(Context context, String rrn)
Input parameters	context – application context rrn – Retrieval Reference Number
Return value	Object APIGetHistoryResult that contains the requested transaction
Description	Enables to get a transaction data by its RRN

getTransactionByInvoice

Signature	APIGetHistoryResult getTransactionByInvoice(Context context, String invoice)
Input parameters	context – application context invoice – the receipt number
Return value	Object APIGetHistoryResult that contains the requested transaction
Description	Enables to get a transaction data by the receipt number

setSingleStepEMV

Signature	void setSingleStepEMV(boolean singleStepEMV)
Input parameters	singleStepEMV – single-factor authorization feature
Return value	No
Description	Enables payments with single-factor authentication

getSingleStepEMV

Signature	boolean isSingleStepEMV()
Input parameters	No

Return value	Sign of a single-factor authentication
Description	Returns the sign of a single-factor authentication

submitFiscal

Signature	APIResult submitFiscal(Context context, String transactionID, int printerID, int docID, int CVC, int shift)
Input parameters	context – application context transactionID – transaction ID that requires the transfer of the fiscal data printerID – fiscal register ID docID – continuous numbering of the document CVC – CVC of the document shift – number of the operation shift
Return value	The result of the data transfer
Description	Performs the transfer of the fiscal data

submitFiscal

Signature	APIResult submitFiscal(Context context, String transactionID, String printerID, int shift, int docSN, String fdn, String fdm, String fs, Date fdt)
Input parameters	context – application context transactionID – transaction ID that requires the transfer of the fiscal data printerID – fiscal register ID shift – number of the operation shift docID – continuous numbering of the document fdn – fiscal number of the document fdm – fiscal attribute of the document fs – fiscal storage fdt – date and time of a fiscal operation
Return value	The result of the data transfer
Description	Transfers fiscalization data according to the standards of the Federal Law 54

printText

Signature	PrintResult printText(String text, Layout.Alignment alignment) throws PaymentControllerException
Input parameters	text – Text for printing alignment – Text alignment
Return value	Printing results
Description	Command operates with WISEPAD2_PLUS readers only, otherwise an exception PaymentControllerExceptioni will be generated

setClientProductCode

Signature	void setClientProductCode(String clientProductCode)
Input parameters	Client application code
Return value	No
Description	Sets the client application code

getClientProductCode

Signature	String getClientProductCode()
Input parameters	No
Return value	Code that is set for a client application
Description	Returns the set code of a client application

addLinkedCard

Signature	void addLinkedCard(Context context, AttachCardContext attachCardContext) throws PaymentException
Input parameters	context – application context attachCardContext – card linking parameters
Return value	No

Description	Outdated. Embarks on a card linking procedure. An exception PaymentException will be generated in an attempt to link the card before the payment (reversal) completion. Asynchronous, see **PaymentControllerListener**

addLinkedCard

Signature	void addLinkedCard(Context context, PaymentController.Currency currency, String acquirerCode) throws PaymentException
Input parameters	context – application context currency – card currency acquirerCode – bank code
Return value	No
Description	Outdated. Embarks on a card linking procedure. An exception PaymentException will be generated in an attempt to link the card before the payment (reversal) completion. Asynchronous, see **PaymentControllerListener**

addLinkedCard

Signature	void addLinkedCard(Context context, PaymentController.Currency currency) throws PaymentException
Input parameters	
Return value	
Description	Outdated. See addLinkedCard (Context context, PaymentController.Currency currency, null)

removeLinkedCard

Signature	APIResult removeLinkedCard(Context context, int cardID)

Input parameters	context – application context cardID – ID of the deleted card
Return value	Operation result
Description	Deletes the linked card for the active account. Synchronous

getLinkedCards

Signature	APIReadLinkedCardsResult getLinkedCards(Context context)
Input parameters	context – application context
Return value	ObjectAPIReadLinkedCardsResult that contains a set of linked cards
Description	Requests a set of linked cards for the active account. Synchronous.

startAutoConfig

Signature	String startAutoConfig() throws PaymentControllerException
Input parameters	No
Return value	Card reader configuration string
Description	Enables to get the configuration of a card reader for a further operation. An exception **PaymentControllerException** will be generated with an attempt to call it while making a transaction or change the configuration.

balanceInquiry

Signature	void balanceInquiry(Context context, PaymentController.Currency currency, String acquirerCode) throws PaymentException
Input parameters	context – application context currency – card currency acquirerCode – bank code
Return value	No

Description	Embarks on a procedure of requesting a card balance. An exception PaymentException will be generated with an attempt to check the balance before a transaction / return completion. Asynchronous, see **PaymentControllerListener.**

balanceInquiry

Signature	void balanceInquiry(Context context, PaymentController.Currency currency) throws PaymentException
Input parameters	context – application context currency – card currency
Return value	No
Description	Outdated. See balanceInquiry(Context context, PaymentController.Currency currency, null)

setRepeatOnError

Signature	void setRepeatOnError(boolean repeatOnError)
Input parameters	repeatOnError – an indicator of the need to request the card again when an error of the transaction processing occurs
Return value	No
Description	Allows to enable/disable the mode of repeated card request by the card reader in case of transaction error

getRepeatOnError

Signature	boolean getRepeatOnError ()
Input parameters	No
Return value	An indicator of the need to request the card again when an error of the transaction processing occurs
Description	Returns an indication of the need to re-request the card in case of a transaction error

submitEmailNotification

Signature	APIResult submitEmailNotification(Context context, String email, String subj, String body, Map<String, byte[]> images)
Input parameters	context – application context email – forwarding address subj – email subject body – html email content images – set of images
Return value	Result of sending a request
Description	Enables to send a notification to a user’s email. Images in an email should be placed according to their code in scr as an attribute cis:image_code (e.g. <img src=”cid:test_image”).

readerInjectKeys

Signature	void readerInjectKeys(String host) throws PaymentControllerException
Input parameters	host – host url for a key firmware
Return value	No
Description	Embarks on a procedure of reader’s keys firmwaring. Available for Urovo devices only. An exception PaymentControllerException will be generated with an attempt to make a payment or change a configuration.

readerConfigEmv

Signature	void readerConfigEmv(Hastable<String, Object> data) throws PaymentControllerException
Input parameters	data – configuration parameters
Return value	No

Description	Outdated, use readerSetConfig. Embarks on a procedure of EMV-configuration change. Available for Urovo devices only. An exception PaymentControllerException will be generated with an attempt to call during a transaction processing or changing a configuration.

readerConfigCapk

Signature	void readerConfigCapk(Hastable<String, Object> data) throws PaymentControllerException
Input parameters	data – configuration parameters.
Return value	No
Description	Outdated, use readerSetConfig. Embarks on a procedure of CAPK-configuration changing. Available for Urovo devices only. An exception PaymentControllerException will be generated with an attempt to call during a transaction processing or changing a configuration.

getKeysHosts

Signature	Map<String, String> getKeysHosts(Context context, String pin)
Input parameters	context – application context pin – access key
Return value	The list of pairs of a type <Host name, host url >
Description	Enables to get a list of hosts for flashing a reader.

tryGetFiscalInfo

Signature	APITryGetPaymentStatusResult tryGetFiscalInfo(Context context, String transactionID)
Input parameters	context – application context transactionID – transaction ID

Return value	Request result that contains **TransactionItem** with updated data.
Description	Enables to perform a request on a fiscal status and fiscal data of the transaction with a request timeout of 60 seconds

tryGetPaymentStatus

Signature	APITryGetPaymentStatusResult tryGetPaymentStatus(Context context, String transactionID)
Input parameters	context – application context transactionID – transaction ID
Return value	Request result that contains **TransactionItem** with an updated data
Description	Enables to request a payment status with a request timeout of 60 seconds. It is used for payments via “payment link”.

settlement

Signature	SettlementResult settlement() throws PaymentControllerException
Input parameters	No
Return value	Operation result
Description	TTK Protocol settlement An exception PaymentControllerException will be generated with an attempt to call it while making a transaction.

prepare

Signature	APIPrepareResult prepare(Context context, String productCode, Map<String, String> fields)
Input parameters	context – application context productCode – product code that requires flling out the fields fields – values of product preparable-fields, according to which fields should be filled out
Return value	Operation result

Description	Fills out the fields for a custom product that has a *preparable* property according to data in its *preparable*-fields. It is recommended to fill only *preparable* fields with values.

readerSetConfig

Signature	void readerSetConfig(String config) throws PaymentControllerException
Input parameters	config – configuration string
Return value	No
Description	Initiates the reader configuration procedure. An exception PaymentControllerException will be generated with an attempt to call it while making a transaction or changing a configuration.

setCustomReaderParams

Signature	void setCustomReaderParams(Hashtable<String, Object> data) throws PaymentControllerException
Input parameters	data – reader operation parameters
Return value	No

Description	Sets the parameters for a reader operation. An exception PaymentControllerException will be generated with an attempt to call it while making a transaction or changing a configuration.
	
	Possible parameters:
	NOTUP (true/false) – automatic activation of NFC on the reader D60 while making a transaction.
	Example:
	`Hashtable<String, Object> param = new Hashtable<>();`
	`param.put(“NOTUP”, true);`
	`PaymentController.getInstance().setCustomReaderParams(param);`

setSoundEnabled

Signature	void setSoundEnabled(boolean enabled)

Input parameters	enabled - sound enabling / disabling
Return value	No
Description	Enables or disables sounds of the reader while making a transaction. Available for a reader D60 only.

readerBeep

Signature	void readerBeep(int count)
Input parameters	count – the number of signals
Return value	No
Description	Initializes the set number of the reader sound signals in case it is connected. Available for a reader D60 only.

fiscalize

Signature	APITryGetPaymentStatusResult fiscalize(Context context, String transcationID)
Input parameters	context – application context transactionID – transaction ID
Return value	The request result that contains **TransactionItem** with updated information
Description	Initializes repeated request of a server fiscalization with a request timeout of 60 seconds

initPaymentSession

Signature	void initPaymentSession() throws PaymentControllerException
Input parameters	No
Return value	No
Description	Initializes internal counters. An exception PaymentControllerException will be generated with an attempt to call it while making a transaction.

submitCash

Signature	PaymentResultContext submitCash(Context context, PaymentContext paymentContext) throws ProcessingException
Input parameters	context – application context paymentContext – payment information
Return value	The result of making a transaction
Description	Synchronous. Conducts a cash payment. An exception ProcessingException will be generated if an error occurs during the request operation

submitDeferred

Signature	PaymentResultContext submitDeferred(Context context, String data) throws PaymentException, ProcessingException
Input parameters	context – application context data – data for a deferred authorization
Return value	Operation result
Description	Synchronous. Sends the authorization data received previously. An exception PaymentException will be generated if a deferred authorization error occurs. An exception ProcessingException will be generated if an error during the request operation occurs.

StartSoftposRegistration

Signature	void startSoftposRegistration(RegistrationCallback callback) throws PaymentControllerException
Input parameters	callback – request result processor
Return value	No
Description	Requests to sign up in an application Tap2Go. An exception PaymentControllerException will be generated with an attempt to call it while making a transaction.

Interface PaymentControllerListener

Callback interface for a class **PaymentController.**

Interface methods:

onTransactionStarted

Signature	void onTransactionStarted(String transactionID)
Input parameters	transactionID – active transaction ID for a simple payment. Canceled transaction ID for a reversal / refund.
Return value	No
Description	Method will be called before conducting a transaction.

onFinished

Signature	void onFinished(PaymentResultContext result)
Input parameters	result – information about the conducted transaction.
Return value	No
Description	The method will be called if the transaction, card linking, or payment reversal are successful.

onReaderEvent

Signature	void onReaderEvent(PaymentController.ReaderEvent event, Map<String,String> params)
Input parameters	event – an event called by the card reader params – linked parameters
Return value	No
Description	The method will be called when an event is received from the card reader. The set of possible linked data see in Appendix 5.

onError

Signature	void onError(PaymentError error, String errorMessage)

Input parameters	error – error type errorMessage – error message. Only used when error == SERVER_ERROR
Return value	No
Description	The method will be called if an error occurs while making a transaction

onSelectApplication

Signature	int onSelectApplication(List<String> apps)
Input parameters	apps – the list of application names
Return value	The sequence number of the selected application (starting from 0)
Description	The method will be called when a chip transaction is executed if the chip card contains more than 1 application. The method call occurs not in the parent thread.

onConfirmSchedule

Signature	boolean onConfirmSchedule(List<Map.Entry<Date, Double>> steps, double totalAmount)
Input parameters	steps – a list of schedule execution steps consisting of pairs of <Date to write-off, Amount to write-off> type totalAmount – total amount for all days
Return value	An indicator that a payer confirms the accuracy of the schedule
Description	The method will be called when a recurring payment is created. The method call occurs not in the parent thread.

onScheduleCreationFailed

Signature	boolean onScheduleCreationFailed(PaymentError error, String errorMessage)
Input parameters	error – error type errorMessage – error message. Used only if error == SERVER_ERROR

Return value	**true** if it is required to repeat the attempt to create a schedule
Description	The method will be called if an error occurs while creating a schedule for a recurring payment

onCancellationTimeout

Signature	boolean onCancellationTimeout()
Input parameters	No
Return value	**true** for a payment return processing
Description	The method will be called in case of attempt to make a payment return when the timeout available for a reversal has expired

onPinRequest

Signature	void onPinRequest()
Input parameters	No
Return value	No
Description	The method will be called when the card PIN code is requested by the card reader

onPinEntered

Signature	void onPinEntered()
Input parameters	No
Return value	No
Description	The method will be called after entering the card PIN code

onBatteryState

Signature	void onBatteryState(double percent)
Input parameters	percent – card reader charge level (in %)
Return value	No

Description	The method will be called after a reader initialization

onSelectInputType

Signature	PaymentController.PaymentInputType onSelectInputType(List<PaymentController.PaymentInputType> allowedInputTypes)
Input parameters	allowedInputTypes – a list of available input types for a payment reversal / return processing
Return value	Selected input type
Description	The method will be called while conducting a reversal/return if a transaction method has to be selected

onAutoconfigUpdate

Signature	void onAutoConfigUpdate(double percent)
Input parameters	percent – a progress indicator (in %)
Return value	No
Description	The method will be called when the progress is updated while the card reader autoconfiguration is in progress

onAutoconfigFinished

Signature	void onAutoConfigFinished(boolean success, String config, boolean isDefault)
Input parameters	success – **true** if the autoconfiguration is successful config – configuration string isDefault – **true** if all default settings were used
Return value	No
Description	The method will be called when the card reader autoconfiguration is complete

onSwitchedToCNP

Signature	void onSwitchedToCNP()
Input parameters	No
Return value	No
Description	The method will be called when a transaction cancellation is performed in CNP mode

onInjectFinished

Signature	void onInjectFinished(boolean success)
Input parameters	success – **true**, if flashing is successful
Return value	No
Description	The method will be called when the card reader keys flashing is complete

onEmvConfigFinished

Signature	void onEmvConfigFinished(boolean success)
Input parameters	success – **true**, if EMV configuration is completed successfully
Return value	No
Description	The method will be called when the EMV configuration of the card reader is complete

onCapkConfigFinished

Signature	void onCapkConfigFinished(boolean success)
Input parameters	success – **true**, if CAPK configuration is completed successfully
Return value	No

Description	Outdated, use onReaderConfigFinished. The method will be called when the CAPK configuration of the card reader is complete

onBarcodeScanned

Signature	void onBarcodeScanned(String barcode)
Input parameters	barcode – scanned barcode
Return value	No
Description	Outdated, use onReaderConfigFinished. The method will be called if the embedded scanner has read the barcode

onReaderConfigFinished

Signature	void onReaderConfigFinished(boolean success)
Input parameters	success – **true**, if configuration is completed successfully
Return value	No
Description	The method will be called when the card reader configuration is complete

Class PaymentContext

JavaBean data container that is required to make a single payment. Please note that the difference of Summ(AuxData) – Amount of the sum of the item list in the field AuxData and payment amount will be recognized as a prepayment. If Amount > Summ(AuxData), an exception **PaymentException** will be generated. The AuxData field will be ignored if the transaction is made via TTK-protocol.

Class properties

Name	Description
amount	Payment amount
amountBig	Payment amount in BigDecimal representation
currency	Payment currency
description	Payment description
transactionID	Is not used
imageFilePath	Path to the image that is linked to the payment
currency	Payment currency
cash	Cash payment sign (deprecated)
credit	Credit payment sign (deprecated)
nfc	NFC transaction sign
method	Payment method sign
acquirerCode	Acquiring bank code
receiptEmail	Email for a receipt
receiptPhone	Phone number for a receipt
linkedCardID	Linked card ID
paymentProductCode	Linked user’s product code
paymentProductTextData	Values of the text fields of the linked product, pairs of the type <field code, value>
paymentProductImageData	Field values with linked product images, pairs of the type <field code, image path>. Not supported for SOFTPOS readers.
extID	Client application ID
amountCashGot	Accepted in cash

amountCashGotBig	Accepted in cash in a BigDecimal representation
auxData	List of goods in the prescribed format
ern	Unique document number within the shift (used during the transaction via TTK protocol)
suppressSignatureWaiting	Signature suppression sign when the receipt is generated and sent to the customer. **True** is set when it is known in advance before the payment that the signature will not be sent.
Name	**Description**
deferred	Indication of payment via deferred authorization (only for D60 and UROVO readers)
skipFiscalization	Sign of the suppression of a fiscalization operation. **True** is set when fiscalization is not required.

Class methods:

reset

Signature	reset()
Input parameters	No
Return value	No
Description	Clears the object fields

putPurchase

Signature	boolean putPurchase(Purchase purchase)
Input parameters	purchase – purchase information
Return value	**true**, is an item was added successfully
Description	Use this method to add information about the product in a standard format **Purchase**

putInvoiceTag

Signature	boolean putInvoiceTag(int tag, Object value)

Input parameters	tag – tag number value –tag value
Return value	**true**, if a tag is added successfully
Description	Adds (changes the value of) Fiscal Data Format 1.05 tag applied to the receipt. Values of *Structure*, *Array* type should be passed as a hex string

Class RegularPaymentContext

The extension of the class **PaymentContext** containing properties necessary for a recurring payment creation. To make a payment on the last day of the month, the property **dayOfWeek** should have a value that equals constant **LAST_DAY_OF_MONTH**. When creating a recurring payment with a product that is only applicable to managed recurring payments (RecurrentMode == MANAGED), it is required to set the field *Managed* to **true**. The other fields must be filled out with the parameters required to make a payment. Explicitly:

```
context.setManaged(true);
context.setRepeatType(PaymentController.RegularRepeatType.Never);
context.setEndType(PaymentController.RegularEndType.BY_QUANTITY);
context.setDay(0);
context.setRepeatCount(1);
context.setStartDate(new Date());
context.setEndDate(new Date());
context.setArbitraryDays(null);
context.setDayOfWeek(0);
context.setHour(0);
context.setMinute(0);
context.setMonth(0);
```

Class properties:

Name	Description
repeatType	Recurring payment type
endType	Method of finishing the recurring payment
startDate	The date of the first payment in recurring payments
endDate	Expiry date of a recurring payment (if set by date)
repeatCount	The number of recurring payments (if set by number of repetitions)
arbitraryDays	Execution day (if the type is set by a date)
month	Month of the execution date ([1,12] and [1,4] if repeatType == Quarterly)
day	Recurring payment day ([1,31])
dayOfWeek	Payment day of the week ([0,7], 0 – Sunday)
hour	Hour of the payment
minute	Minute of the payment
managed	An indication that the regular payment settings will be defined by the host

The set of required completed properties depends on the type of payment:

Payment type	**Set of properties**
Never	startDate
Weekly	startDate, (endDate or repeatCount)
Monthly	startDate, (endDate or repeatCount), day
Quarterly	startDate, (endDate or repeatCount), month, day
Annual	startDate, (endDate or repeatCount), month, day
ArbitraryDays	arbitraryDays

Parameters repeatType, endType, receiptEmail, receiptPhone are required for all types of regular payments.

Parameters hour, minute are optional for all types of regular payments.

Class ReversePaymentContext

JavaBean data container that is required for a payment reversal / refund. Note that the end user has to verify the corectness of the AuxData content. The difference between Summ(AuxData) – Amount of the sum of the item list amount in the AuxData field and the reversal (refund) Amount will be recognized as a prepayment. To use a credit voucher, the transactionID field must have a value of **null**. A credit voucher can only be used with a payment card.

Class properties:

Name	Description
transactionID	Transaction ID of the payment reversal
action	Reversal type
currency	Currency used for a reversal / return
returnAmount	Payment amount of a reversal that will be executed. For a full reversal set **null**
returnAmountBig	Payment amount of a reversal that will be executed as a BigDecimal representation
auxData	List of goods in a prescribed format
extID	Client application ID
receiptEmail	Email for a receipt
receiptPhone	Phone number for a receipt
ern	Unique document number within the shift (used during the transaction via TTK protocol)
suppressSignatureWaiting	Signature suppression sign when the receipt is generated and sent to the customer. **True** is set when it is known in advance before the payment that the signature will not be sent.
acquirerCode	Acquiring bank code (only for a credit voucher)
nfc	NFC credit voucher sign
skipFiscalization	Sign of the suppression of a fiscalization operation. **True** is set when fiscalization is not required.

Class methods:

reset

Signature	reset()
Input parameters	No
Return value	No
Description	Clears the object fields

putPurchase

Signature	boolean putPurchase(Purchase purchase)
Input parameters	purchase – item information
Return value	**true**, if an item was added successfully
Description	Use this method to add product data in a standard **Purchase** format

putInvoiceTag

Signature	boolean putInvoiceTag(int tag, Object value)
Input parameters	tag – tag number value – tag value
Return value	**true**, if the tag was added successfully
Description	Adds (changes the value of) Fiscal Data Format 1.05 tag applied to the receipt. Values of *Structure*, *Array* type should be passed as a hex string

Class AttachCardContext

JavaBean data container that are required for a card linking

Class properties

Name	Description
currency	Card currency
acquirerCode	Acquiring bank code
deferred	Indication of payment via deferred authorization (only for a reader D60)

Class PaymentResultContext

JavaBean data container that are received when the payment is made or reversed.

Class properties:

Name	Description
transactionItem	Payment transaction / reversal information as a **TransactionItem** representation
scheduleItem	Recurring payment transaction information as a **ScheduleItem** representation
requiresSignature	An indication of the need to send the payer's signature after the payment is conducted
terminalName	A terminal
emvData	EMV (chip) transaction dataset in a representation of **HashMap<String, String>**
attachedCard	Information about the card that is linked when **PaymentController.addLinkedCard()** is called
deferredData	Transaction data that is sent during a deferred authorization
cardHash	Encrypted PAN card
tranId	Transaction ID for a deferred authorization

Interface RegistrationCallback

Request result processor of signing up in a Tap2Go application.

Interface methods:

onFinished

Signature	void onFinished(String accesCode)
Input parameters	accessCode – code that is set to access a new personal account
Return value	No
Description	The method is called if the registration is successful

onFailed

Signature	void onFailed(String error)
Input parameters	error – error message
Return value	No
Description	The method is called if the registration is not successful

Package com.mpos.sdk.entities

Class AbstractEntity

An abstract wrapper class for a data array in JSON representation. It implements the **Serializable** interface.

Class methods:

getJSON

Signature	JSONObject getJSON()
Input parameters	No
Return value	JSON representation of a dataset
Description	Returns a JSON dataset representation

Class TransactionItem

AbstractEntity child class. It is an object representation of a transaction. It contains a set of properties that define it. It also has nested classes.

Class properties:

Name	**Description**
ID	Transaction ID
Date	Time and date of a transaction, according to the GMT of the device
Description	Transaction description
Invoice	Receipt number
ApprovalCode	Verification code
ScheduleID	Recurring payment ID
ScheduleStepID	Recurring payment write-off ID
Amount	Transaction payment amount
AmountEff	Transaction balance
InputType	Payment method
Operation	Operation name
Latitude	Geographical latitude of the transaction location
Longitude	Geographical longitude of the transaction location
HasPhoto	An indication of the attached image
PhotoUrl	URL of an attached image
HasSignature	An indication of the attached signature
SignatureUrl	URL of the attached signature
StateDisplay	Transaction status description
Card	Information about the card that was used for making a payment as a representation of **TransactionItem.Card**
CanCancel	An indication of the possibility to cancel a payment
CanReturn	An indication of the possibility to make a refund
CanCancelPartial	An indication of the possibility to reverse a payment partially

CanReturnPartial	An indication of the possibility to make a partial refund
DisplayMode	Transaction display type as a **DisplayMode** representation
SubstateDisplay	Transaction substate description
CardholderName	Cardhodler
TerminalName	Terminal

Name	Description
ExternalPayment	Payment via link data in a representation of **TransactionItem .ExternalPayment**
CancelReturnTypes	List of input types available for a payment reversal / refund
ReceiptEmail	Email for a receipt
ReceiptPhone	Phone number for a receipt
Balance	Card balance
CanCancelCNP	The transaction can be canceled in a CNP mode
CanCancelCNPpartial	The transaction can be partially canceled in a CNP mode
CanReverseCNP	The transaction can be refunded in a CNP mode
CanReverseCNPPartial	The transaction can be partially refunded in a CNP mode
RRN	Retrieval Reference Number
SignatureRequired	An indicator of the necessity to send a signature
FiscalInfo	Transaction fiscal data in a representation of **TransactionItem .FiscalInfo**
AmountCashGot	Accepted with cash
AuxData	List of goods in an established format
TaxMode	Tax transfer mode used during fiscalization
TaxContributions	Tax accruals in the **TaxContribution** view, excluding goods in AuxData
Taxes	List of taxes in a **Tax** representation, excluding goods in AuxData
TaxSystemName	Tax system that is used during fiscalization
Products	User’s goods linked to the transaction
InvoiceTags	List of Fiscal Document Formats that are applicable to the whole receipt
PANTags	Specific client data
Format	Format of a payment information depiction
TransPos	Information about the user who performed the transaction in a **TransPos** representation

Sets of parameters:

DisplayMode

Type of a transaction display:

Type	Description
DECLINED	Declined transaction
SUCCESS	Successful transaction
REVERSE	Payment reversal / refund
REVERSED	Payment reversal / return is complete
NONFINANCIAL	

TaxMode

Tax accrual mode

Type	Description
FOR_EACH	Tax accrual for an each item
FOR_TOTAL	Receipt tax accrual

Class methods:

isNotCanceled

Signature	Boolean isNotCanceled()
Input parameters	No
Return value	The indication that the payment reversal / return is complete
Description	Returns the attribute of a payment reversal / return

Class TransactionItem.Card

Nested class **TransactionItem**, child class **AbstractEntity**. Contains data about payment card.

Class properties:

Name	Description
Iin	Card type or "cash" (if accepted in cash)
Bin	Bank identification number
Exp	Card expiry date
PanMasked	First and last four digits of the card number divided by the symbol "*"
PanEnding	Last 4 digits of the card number
BankName	Name of the bank
BankCounryID	

Class methods:

isCash

Signature	isCash()
Input parameters	No
Return value	**true**, if a payment was made in cash
Description	Returns an indication that the payment was made in cash

isPrepaid

Signature	isPrepaid()
Input parameters	No
Return value	**true**, if a transaction was performed as a prepayment
Description	Returns an indication of the prepayment

isCredit

Signature	isCredit()
Input parameters	No
Return value	**true**, if the transaction is performed on credit
Description	Returns an indication that the payment was made on credit

isOuter

Signature	isOuter()
Input parameters	No
Return value	**true**, if the transaction is performed with an external POS terminal
Description	Returns an indication that the payment was made with an external POS terminal

Class TransactionItem.ExternalPayment

Nested class **ExternalPayment**, child class **AbstractEntity**. Contains data about payment by via payment link.

Class properties:

Name	Description
Type	Data display type
Link	Link for making a payment
QR	A set of <Title, Content> pairs to be displayed as QR codes

Sets of parameters:

Type

Payment display method

Type	Description
LINK	To display the link
QR	To display the QR-code

Class TransactionItem.FiscalInfo

AbstractEntity child class. Contains fiscal data of the payment.

Class properties:

Name	Description
CVC	CVC
FiscalDocumentNumber	Fiscal Document
FiscalStorageNumber	Fiscal Storage
FiscalMark	Fiscal Mark
FiscalDeviceID	Fiscal Device ID
FiscalDeviceRegNumber	Registration number of the Fiscal Device
FiscalDocSN	Fiscal document number
FiscalPrinterShift	Shift number
FiscalDatetime	Fiscalization date and time
FiscalStatus	Fiscalization status

Sets of parameters

FiscalStatus

Fiscalization status

Type	Description
NONE	Not set
CREATED	Is being fiscalized
SUCCESS	Fiscalization is completed
FAILURE	Fiscalization error has ocurred

Class TransactionItem.Product

AbstractEntity child class. It contains the data of the user product linked to the transaction.

Class properties:

Name	**Description**
Description	Product description as a representation of a **PaymentProductItem**
Fields	Values of the product fields

Class TransactionItem.ProductField

AbstractEntity child class. It contains the field data of the user product linked to the transaction.

Class properties:

Name	**Description**
Description	Field description in a representation of **PaymentProductItemField**
TextValue	Value of the text field
ImageUrl	Link to the image for the field with the image

Class ScheduleItem

AbstractEntity child class. It is an object representation of the recurring payment data.

Class properties:

Name	**Description**
ID	Recurring payment ID
Card	The details of the card that was used for payment in **TransactionItem.Card**

Class Account

AbstractEntity child class. Contains user’s personal account data.

Class properties:

Name	**Description**
SingleStepAuth	An indicator of accessibility of single-factor authorization
Name	Agent name
BranchName	Branch name
BranchAddress	Branch address
BranchPhone	Branch phone number
ClientName	Name of the company
ClientLegalName	Legal name of the company
ClientLegalAddress	Legal address of the company
ClientRealAddress	Real address of the company
ClientPhone	Company phone number
ClientWeb	Company website
BankName	Bank name
TerminalName	Terminal number
AcquirersByMethods	Pairs of the type <Bank code, Bank name> grouped by available input methods
ID	Account’s ID
PaymentOptions	Set of possible payment methods
LinkedCards	Set of linked cards
Email	Account email
NfcNotup	Setting up automatic activation of the NFC module for QPOS readers

Class Account.PaymentOption

Nested **Account** class, **AbstractEntity** child class. It contains data about the payment type supported by the bank.

Class properties:

Name	**Description**
InputType	Payment type
Code	Bank code
AcquirerName	Name of acquiring bank

Class LinkedCard

AbstractEntity child class. It contains data of the linked card.

Class properties:

Name	**Description**
ID	Linked card ID
State	State
isDeleted	An indicator that the card is not linked anymore
Alias	Alias of the payment card to display
PanEnding	Last 4 digits of the card
PanMasked	First and last four digits of the card number divided by the symbol “*”
Balance	Balance of the card. It is used only for the card linking operation.
Bin	Bank Identification Number

Sets of parameters:

State

Card state

Type	**Description**
DISABLED	The card is unavailable for operation
ENABLED	Card transactions are allowed

Class Purchase

AbstractEntity child class. The product data is in a standard format.
Attention! If Fiscal Documents Format tags are present in the product, the corresponding property values specified through the fields “Title”, “Price”, “Quantity”, “TaxCode” will be ignored, i.e. a request, for example, for the “Title” property will return the value of the tag 1030, if such is specified. Otherwise, the value of the "Title” property will be returned.

Constants:

Name	Description
TaxCode.VAT_NA	Tax rate code “No VAT”
TaxCode.VAT_0	Tax rate code “VAT 0%”
TaxCode.VAT_10	Tax rate code “VAT 10%”
TaxCode.VAT_18	Tax rate code “VAT 18%”
TaxCode.VAT_20	Tax rate code “VAT 20%”
VAT_110	Tax rate code VAT 10/110
VAT_120	Tax rate code VAT 20/120

Class properties:

Name	Description
Title	Product name (Tag 1030)
Price	Unit price (Tag 1079)
Quantity	Quantity (Tag 1023)
TitleAmount	The payment amount of the unit, including surcharge (Tag 1043)
TaxCode	List of codes of applicable tax rates (Tag 1199)
Tags	Presentation of the product in the form of a list <tag, value> according to Fiscal Document Format 1.05

Class methods:

Build

Signature	static Purchase Build(String title, double price, double quantity, Double itleAmount, List<String> taxCodes)
Input parameters	Instance property values
Return value	Object Purchase
Description	Enables to create an object Purchse, using product's standard format

Build

Signature	static Purchase Build(Map<Integer, Object> tags) throws IllegalArgumentException
Input parameters	tags – Represenation of the product in the form of a list <tag, value> according to Fiscal Document Format 1.05
Return value	Object *Purchase*
Description	Enables to create an object Purchase, using the product description via Fiscal Document Format 1.05. Supported classes as keys: • Integer (Tags of the types *Integer* and *Flags*) • Double (tags 1079 and 1023 only) • String – the rest Values of the *Structure* and *Array* types should be passed as a hex string. An exception IllegalArgumentException will be generated if a different class is specified.

Class Tax

AbstractEntity child class. It contains data about the Tax Rate.

Class properties:

Name	**Description**
Code	Rate code
Name	Name
Rate	Rate value

Class TaxContribution

AbstractEntity child class. It contains data on tax accrual at the rate.

Class properties:

Name	**Description**
Code	Rate code
Name	Name
Rate	Rate value
Total	Accrual amount

Class PaymentProductItem

AbstractEntity child class. It contains data about the user product.
Attention! When creating recurring payments for products with RecurrentMode == MANAGED, get acquainted with the description of **RegularPaymentContext**.

Class properties:

Name	**Description**
Code	Unique product code
State	Product state
Apply	Available payment types
Title	Product name
TitleReceipt	Product name for the receipt printing
Fields	Product fields
Preparable	Indicates that filling in the product fields should be performed as a result of calling the method *PaymentController.prepare()*.
PreparableEditable	Indicates that editing of uploaded field values is allowed
PreparableOptional	Indicates that the field values can be filled in without calling the method PaymentController.prepare().
RecurrentMode	Recurrent payment settings

Sets of parameters:

State

Product state

Type	**Description**
DISABLED	Product is disabled
ENABLED	Product is enabled

PaymentType

Available payment types for a product

Type	Description
PAYMENT	Regular payment only
RECURRENT	Recurring payment only
BOTH	Both payment types
NONE	Payments are unavailable

RecurrentMode

Setting up a regular payment for a product

Type	Description
UNDEFINED	Not defined
SIMPLE	The schedule and settings of the regular payment are transmitted when creating the payment
MANAGED	The schedule and settings of the regular payment are set by the server and are not transmitted during payment

Class PaymentProductItemField

AbstractEntity child class. Contains data about the user product field. When making a payment with a product, it is recommended to transfer, among other things, inaccessible to the user (!UserVisible) fields with default values.

Class properties:

Name	**Description**
Type	Field type
Code	Unique field code
Required	Required field
Preparable	The value of the field is used when calling the method *PaymentController.prepare()*.
Title	Field name
TitleReceipt	Field name to print in a receipt
DefaultValue	Default value
TextMask	Input mask (regular expression)
TextRegExp	A regular expression to check the accuracy of the input
Multiline	An indicator of a multi-line field
Numeric	Indicates that the field value is a decimal number
ReceiptEmail	Indicates that the value of the field should be used as an email to send the receipt
ReceiptPhone	Indicates that the field value should be used as a phone number to send the receipt
UserVisible	Indicates that the field is available to the user
PrintInReceipt	Indicates that the field should be printed in the receipt

Sets of parameters:

Type

Field type

Type	Description
TEXT	Text field
IMAGE	Field with an image

Class PreparedField

AbstractEntity child class. Wrapper for code/value pairs obtained as a result of Payment Controller.prepare()

Class properties:

Name	Description
Code	Field code
Value	Field value

Class methods:

isPaymentAmount

Signature	boolean isPaymentAmount()
Input parameters	No
Return value	An indicator that the field value should be used as the payment amount
Description	Returns an indication that the value of the field should be used as the payment amount

Class Format

AbstractEntity child class. It contains the data about the format of the data display.

Class properties:

Name	**Description**
CurrencySign	Currency symbol
CurrencySignSafe	Alternative display of the currency symbol (without using special characters)
AmountFormat	Format for a payment amount display according to java.text.DecimalFormat
AmountFormatWithoutCurrency	Format for a payment amount display according to java.text.DecimalFormat without a currency symbol
CurrencyE	The number of decimal places in the total

Class TransPos

AbstractEntity child class. Contains data about the user who completed the transaction.

Class properties:

Name	**Description**
ID	Account ID
Email	Account email
Name	Agent’s name

Class APIResult

AbstractEntity child class. It is a primitive entity containing a response from the server.

Class methods:

getErrorCode

Signature	int getErrorCode()
Input parameters	No
Return value	Error code
Description	Returns the error code. 0 – if the response does not contain error messages, -1 – if the response from the server is not received, or the response format is incorrect

getErrorMessage

Signature	String getErrorMessage()
Input parameters	No
Return value	Error message
Description	Returns an error message

isValid

Signature	boolean isValid()
Input parameters	No
Return value	An indicator that a response doesn’t contain error messages and its format is correct
Description	Returns an indicator that a response doesn’t contain error messages and its format is correct

Class APIGetHistoryResult

APIResult child class. It contains the set of transactions received in response to a history request.

Class methods:

getTransactions

Signature	ArrayList<TransactionItem> getTransactions()
Input parameters	No
Return value	ArrayList of the transactions
Description	Returns the set of transactions contained in the response

getInProcessTransactions

Signature	ArrayList<TransactionItem> getInProcessTransactions()
Input parameters	No
Return value	ArrayList of the transactions
Description	Returns a set of transactions with the status "In process"

Class APIAuthResult

APIResult child class.

Class methods:

getAccount

Signature	Account getAccount()
Input parameters	No
Return value	Object Account
Description	Returns information about the personal account

getTaxID

Signature	String getTaxID()
Input parameters	No
Return value	Taxpayer Identification Number (TIN)
Description	Returns the user’s Taxpayer Identification Number (TIN)

getProducts

Signature	String getProducts()
Input parameters	No
Return value	Set of user products
Description	Returns the set of user products

getFormat

Signature	Format getFormat()
Input parameters	No
Return value	Object Format
Description	Returns the default format for payment information display

Class APIReadLinkedCardsResult

APIResult child class.

Class methods:

getLinkedCards

Signature	ArrayList<LinkedCard> getLinkedCards()
Input parameters	No
Return value	ArrayList pf the linked cards
Description	Returns the set of linked cards contained in the response

Class APITryGetPaymentStatusResult

APIResult child class.

Class methods:

getTransaction

Signature	TransactionItem getTransaction()
Input parameters	No
Return value	TransactionItem with the updated information
Description	Returns the transaction with the updated information

Class SettlementResult

The result of the operation “Settlement”

Class properties:

Name	**Description**
Success	An indicator of a successful transaction
ErrorMessage	Error message

Class APIPrepareResult

APIResult child class.

Class methods:

getFields

Signature	List<PreparedField> getFields()
Input parameters	No
Return value	The list of received values for the fields of the user product
Description	Returns the list of received values for the fields of the user product

Package com.mpos.sdk.ui

Class SignatureView

It is a View that provides the opportunity to put a client's signature by moving a finger or stylus on the screen.

Class properties:

Name	Description
color	Brush color

Class methods:

erase

Signature	erase()
Input parameters	No
Return value	No
Description	Clears the signature field

getBitmap

Signature	Bitmap getBitmap()
Input parameters	No
Return value	Bitmap signature representation
Description	Returns Bitmap signature representation

getBitmapByteArray

Signature	byte [] getBitampByteArray()
Input parameters	No
Return value	byte [] signature representation
Description	Returns byte [] signature representation

getBitmapBlack

Signature	Bitmap getBitmapBlack()
Input parameters	No
Return value	Bitmap signature representation
Description	Bitmap signature representation in black

getBitmapByteArrayBlack

Signature	byte [] getBitampByteArrayBlak()
Input parameters	No
Return value	byte [] signature representation
Description	byte [] signature representation in black

Appendix 1: Slip printing

Data for slips are stored in the event *onFinished*
The client's details can be obtained using the method PaymentController.auth().

Slip fields:

Name	Description
Bank	Account.getBankName()
Company name	Account. getClientName ()
Legal entitiy name	Account. getClientLegalName ()
Company phone number	Account. getClientPhone ()
Company website	Account. getClientWeb ()
Transaction date and time	paymentResultContext.getTransactionItem().getDate()
Terminal number	paymentResultContext.getTransactionItem().getTerminalName()
Receipt number	paymentResultContext.getTransactionItem().getInvoice()
Verification code	result.TransactionItem.AcquirerApprovalCode

Card number and type	paymentResultContext.getTransactionItem().getCard().getIin(),paymentResultContext.getTransactionItem().getCard().getPanMasked()
Transaction EMV tags	paymentResultContext.getEmvData(), are printed as a key-value
Operation type	paymentResultContext.getTransactionItem().getOperation()
Transaction amount	paymentResultContext.getTransactionItem().getAmount()
Fee	R 0.00
Status	Successful
Client’s signature	Place for a signature if paymentResultContext.isRequiresSignature()==true, otherwise «Confirmed by entering PIN»

Slip example:

Bank name
Test Client
"Test Client"
+27 00 000 0000
www.testclient.com
Transaction time and date: 21.03.2017 15:47:34
Terminal: II040001
Receipt: RM7ZEDMAAE7L
Confirmation code: SIMULATION
Payment card: mastercard **** 5631
AID: A0000000041010
TSI: 6800
TVR: 8020008000
Operation type: Purchase
Total: R 33
Fee: R 0.00
Status: Successful
Confirmed by entering PIN.

Appendix 2: Example of a Receipt

Position 1 – Without VAT
Position 2 – VAT 10%
Position 3 – VAT 15%
Position 4 – VAT 18%
Position 5 – VAT 20%

Appendix 3: Error Codes for Readers Operating with TTK-protocol

Error code	Description
B4	ERN incorrect number
BB	Log synchronization is required
FE	Incorrect message format, missing mandatory fields
JF	Reconciliation of totals is required
NF	The original transaction is not found by a bank receipt number
UN	The operation cannot be completed due to the functionality limitations
UP	Software update is required

Appendix 4: List of Supported Currencies

Type	Description
ARS	Argentine peso
BND	Brunei dollar
BOB	Bolivian boliviano
BRL	Brazilian real
CAD	Canadian dollar
CLP	Chilean peso
COP	Colombian peso
CRC	Costa Rican colon
CUP	Cuban peso
DOP	Dominican peso
EUR	Euro
HNL	Honduran lempira
HTG	Haitian Gourde
IDR	Indonesian rupiah
KHR	Cambodian riel
KRW	South Korean won
LAK	Laos kip
MMK	Myanmar kyat
MXN	Mexican peso
MYR	Malaysian ringgit
NIO	Nicaraguan cordoba
PAB	Panamanian balboa
PEN	Peruvian sol
PHP	Philippine peso
PYG	Paraguayan guarani
SGD	Singapore dollar

THB	Thai baht
USD	USA dollar
UYU	Uruguayan peso
VND	Vietnamese dong

Appendix 5: Reader Event Parameter List

INIT_SUCCESSFULLY

For readers D60 only

bootloaderVersion	
hardwareVersion	
firmwareVersion	
posId	Reader serial number

CARD_INFO_RECEIVED

For readers D60 only

panHash	Hashed card number