Technical specification terminal driver for cash register - KIT

Date:	**19.12.2024**
Version:	**5.03.00**
Status:	**Released**
Classification:	**Confidential**

List of document versions

Data: Technical Specification KIT-combined			
Ver.	**Date**	**Author**	**Description**
1.0	12.10.08	Az	Initial version
1.3	31.10.08	Az	Version for release 0.1
1.4	09.11.08	Az	Version for release 0.2
1.5	17.11.08	Mas	Version for release 0.3
1.6	24.11.08	Az	Version for release 0.4
1.7	01.12.08	Az	Version for release 0.5
1.8	08.12.08	Az	Version for release 0.6
1.9	11.12.08	Az	Version for release 0.7
1.10	15.12.08	Az	Version for release 0.8
1.11	22.12.08	Az	Version for release 0.9
1.12	05.01.09	Az	Version for release 1.0
1.13	19.01.09	Az	Version for release 1.1
1.14	21.01.09	Az	Version for release 1.2
1.15	02.02.09	Az	Version for release 1.4
1.16	09.02.09	Az	Version for release 1.5
1.17	23.02.09	Az	Version for release 1.7
1.18	02.03.09	Az	Version for release 1.8
1.19	09.03.09	Az	Version for release 1.9
1.20	12.03.09	Az	Version for release 1.10
1.21	23.07.09	Az	Version for release 1.13
1.22	28.08.09	Az	Version for release 1.15
1.23	31.08.09	Az	Version for release 1.16

Data: Technical Specification KIT-combined			
Ver.	**Date**	**Author**	**Description**
1.24	22.09.09	Mas	Minor addition to release 1.16
1.25	09.12.09	Mas	Version for release 1.17
1.25a	15.03.10	Mas	Version for release 1.17a
1.26	01.03.10	Mas	Version for release 1.18
1.27	03.05.10	Mas	Minor changes to improve dialog understandability
2.0	23.06.10	Mas	First English version
2.1	01.07.10	Mas	Platform specific information added
2.2	20.07.10	Mas	Version for release 1.19, added application currency code in device class with simulation in config file, added new status waiting for transaction request
2.3	10.08.10	Mas	Version for release 1.19a: - added class TransactionInit - changes in device class - changes in configuration file - changes in devicecallbacks
2.4	13.8.10	Mas	Version for release 1.19b: - added diagram to TransactionInit - changed parameters in transactioninit - changed parameters in device
2.4	30.8.10	Mas	Version for release 1.19d: - changes in TransactionInit
2.5	8.9.10	Mas	Version for release 1.19h: - KITDriver
2.6	22.9.10	Mas	Version for release 1.19i: - changes in Transaction.OnRollback() - changes in Device.getIIN()

Data: Technical Specification KIT-combined			
Ver.	**Date**	**Author**	**Description**
2.7	9.11.10	Mas	Version for release 1.20: - added property Transaction.getCVMResults() - added property Transaction.getExtendedTransactionResult() - added enum Transaction.ExtendedTransactionResult - added class Transaction.CVMResults with new methods and enumerations - added Device.isESignatureActive() - new chapter 5.7.1 added - added new configuration file attribute TOGenApp in 2.2.1 - added new configuration file attribute ToTrxRsp in 2.2.1 - added description of application timeouts in 6.1.1 and 4 - extended description in POSEntryMode
2.8	19.11.10	Mas	Version for release 1.20: - changed the names of enumerations in Transaction.CVMResults - added value to the enum Transaction.CVMResults.Code - changed the type of the Transaction.CVMResults.getCodes() - added value to the enum Transaction.CVMResults.Condition
2.9	25.11.10	Mas	Version for release 1.20a: - added new timeouts for the application see 2.2.1 - added new behavior description see 6.1.1
2.10	28.1.10	Mas	Version for release 1.20b: - added description of the signature flag behavior
2.11	28.1.10	Mas	Version for release 1.21a: - added class Receipt - added class Report - added class ReportEntry - published Device.State.isInStates - added Device.requestReceipt - added Device.requestReport - added DeviceCallbacks.onReceiptResponse - added DeviceCallbacks.onReportResponse - added class KITExceptionInvalidOperation
2.12	17.5.10	Mas	Version for 1.22 - various corrections in Transaction and Device classes to match actual API

Data: Technical Specification KIT-combined			
Ver.	**Date**	**Author**	**Description**
2.13	27.5.10	Mas	Version for 1.22a - tipAmt added to transaction - CVM Result code added to Enumeration - remove graphical class overview
2.14	20.7.10	Mas	Version for 1.23 - new Custom Dialog E_CHECK - new method for passing the config file - KITExceptionConfigStreamInvalid - new extended result code added TRX_RESULT_ABORTED_ECR_CONFIRMATION_FAILED
2.15	29.8.10	Mas	E_CHECK description was missing in previous version
2.17	12.9.11	Mas	Changed the picture of the terminal
2.19	13.10.11	Mas	-Added new transaction functions -Added new resource name
2.20.0	10.05.12	Mas	-Added new device methods and attributes (according to Post OCS 125) - new attribute getAcquirers - new methods: swDownload, ejectCardForced, reboot - new class Acquirer - new class ConfigResponse - new onConfigure callback overload
			Major.minor version numbers of the specification correspond now with the major.minor version numbers of the KIT software
2.20.1	16.05.12	Mas	-new property Device.getLastCleaningDate -new chapter 5.6 -new additions and fixes to description of callbacks in 4.3.1 -additions and fixed in 3.2.1 -new class Brand in 3.4 -new class Acquirer
2.20.2	16.05.12	Mas	- Device.getLastCleaningDate returns complete local time - Acquirer.getLastInitTime changed to .getLastInitDate and returns complete local time - ConfigResponse class and parameter was eliminated, the information was moved to Device, isSWDownalodPending was changed to nextSWDownaloadDate - swDownload was changed to swUpdate
2.22.3	30.11.12	Mas	-new Class was added for TransactionDatachange -new callback for the device when reason of transaction data change occurs -transactions can be aborted in more ways (silent implemented)
2.22.4	30.11.12	Mas	-modified members of enumeration TransactionDatachange.Reason -added new descriptions for data of TransactionDatachange

Data: Technical Specification KIT-combined			
Ver.	**Date**	**Author**	**Description**
2.22.5	21.12.12	Mas	-new Transaction.setAmounttAuth
2.23.0	31.01.13	am	-new Device.setReportIINZeroCards -new Device.hasReportIINZeroCards -new Device.getAppPAN
2.23.1	Not public	am	-removed Device.getAppPAN and replaced by Device.getTrack2Data

Data: Technical Specification KIT-combined			
Ver.	**Date**	**Author**	**Description**
4.00.00	23.06.13	am	**Merged following changes for New Loyalty into this version:** **2.16:** Specification for New Loyalty: - new Trx type RedeemOnly - new callback Trx notification - new Trx attributes - new class Voucher - new class RedemptionItem new class TransactionNotification **2.18**: -Changed name of Transaction Nofication classes and callbacks into loyalty specific names -Completed according to new loyalty specs fragments **3.0.0:** -New Loyalty: Stage Gate B (according to NL specs 0.85/NL ECR specs 0.5) **3.1.0:** Added: - Device.setDeviceAddress New Loyalty: Iteration 5 Added: - class LoyaltyProductRecord - class LoyaltyAdvice - class RedemptionItem - DeviceCallbacks.onAddToBasketResponse - DeviceCallbacks.onClearBasketResponse - DeviceCallbacks.onLoyaltyAdviceNotification - Device.addToBasket - Device.clearBasket - LoyaltyPromotion.getRefNum - LoyaltyPromotion.getTrmId - LoyaltyPromotion.getRedemptionItems - Transaction. getRedemptionItems

Data: Technical Specification KIT-combined			
Ver.	**Date**	**Author**	**Description**
4.00.00	23.06.13	am	**(cont)** **3.1.0:** Correction of 3.1.0 version, no KIT changes **4.00.00:** - Method Device.swUpdate() is now correctly referenced in spec - New Loyalty v1.12: CampaignSubTitle and CampaignSummaryText - Improved logging in KIT (now also adds date to timestamp)
4.00.01	19.07.13	am	Added missing documentation (no changes to driver): - Device.addToBasket - Device.clearBasket - DeviceCallbacks.onAddToBasketResponse - DeviceCallbacks.onClearBasketResponse - DeviceCallbacks.onLoyaltyPromotionNotification - DeviceCallbacks.onLoyaltyAdviceNotification
4.00.02		am	Changed documentation (no changes to driver): - onConfigureResponse() does not have any arguments
4.01.00	09.12.13	am	Added Cashback with Purchase Transaction - new Method Transaction.createPurchaseWithCashback - new Extended Transaction Results: TRX_RESULT_DECLINED_EP2_CASHBACK_AMOUNT_EXCEEDED, TRX_RESULT_DECLINED_EP2_CASHBACK_NOT_ALLOWED, TRX_RESULT_ABORTED_EP2_MAXIMUM_CASHBACK_AMOUNT_EXCEEDED, TRX_RESULT_ABORTED_EP2_BELOW_MINIMUM_CASHBACK_AMOUNT, TRX_RESULT_ABORTED_EP2_PURCHASE_AMOUNT_BELOW_MINIMUM_FOR_CASHBACK Bugfix CREDIT Transaction Receipts - Requesting a merchant receipt (Receipt.ReceiptType.TRANSACTION) after a Function CREDIT returned the cardholder receipt and vice versa. KIT now returns the correct receipt for the last transaction only.
4.01.01	14.02.14	am	Added new method Transaction.isCUPTransaction() which returns true for a China UnionPay (CUP) transaction. Bugfix: CVMResults.isSignatureVerified() returns true for CUP transactions. All CUP transactions are PIN- and Signature-Verified.
4.01.02	18.03.14	am	Bugfix: RequestReceipt method was failing after a disconnect/connect to the device
4.01.03	25.06.14	am	Added new ExtendedTransactionResults: TRX_RESULT_APPROVED_EP2_ONLINE TRX_RESULT_APPROVED_EP2_ONLINE_DUPLICATE_OPERATION TRX_RESULT_APPROVED_EP2_ONLINE_ORIGN_RES_DATA_INCLUDED Changed behavior: When an unknown extended transaction result code is found, UNSPECIFIED with it's actual code is returned instead of -1

Data: Technical Specification KIT-combined			
Ver.	**Date**	**Author**	**Description**
4.01.04	06.11.14	am	Added new Device method requestConfirmationTimeExtension(int seconds) By requesting a confirmation time extension, an ECR can increase the timeout of the device waiting for a transaction confirmation (commit). The increase is absolute and will overwrite previous timeouts. Seconds can be between 0 and 30 seconds.
4.02.00	03.02.15	am	Added new Transaction functions "Activate Card" and "Load Card" (available only on ep2 6.0+ terminals if enabled by Acquirer) Added new static Transaction methods: createActivateCard() and createLoadCard(Currency, Amount) Added new Device method setTerminalLanguage(Language)
4.03.00	08.09.17	ts	Added new Device functions: getAppExpDate() getAppPAN() getMyOneBranchenCode()
4.04.00	23.04.18	ts	Added new functionality for “Partial Approval”-Transactions
4.05.00	18.01.19	ts	Added new functionality for Purchase Authorization Transactions: get/setAmountFinal setTrxSeqCnt
4.05.00	08.03.19	ts	Added new Tag TAG_CardholderLng Added new Device accessor get/setCardHolderLanguage Added new TAGs TAG_SurrogatePAN, TAG_ListofActiveSurrogatePANs Added Transaction accessors for SurrogatePAN and ListofActiveSurrogatePANs Added new TRX Result Event TRX_RESULT_ABORTED_MATCHED_SURROGATE_PAN
4.10.00	12.03.19	ts	Added new Dialog Resourcetypes for Freetext and pump selection Remove UIExclusive Checks for Dialogs
4.11.00	29.04.19	ts	Added new enumeration TransactionRequestFlags (SILENT) Added new get/set TransactionRequestFlags for Transaction
4.11.01	30.04.19	ts	Added device property getTrack2Data. Was present in library but not in the documentation Fixed link to CVMResults in 3.9.3 getCVMResults
4.13.00	17.06.19	ts	Added new Dialog Resourcetypes for Mileage and Additional Info
4.13.01	19.06.19	ts	3.17.4.2 Dialog.Button • Removed ‘Clear Button’ and all combinations • Added Left- and RightCustom Button and all combinations 3.16.3.3 ResourceText • Added new Resource Text ‘Free_Text_with_Buttons’

Data: Technical Specification KIT-combined			
Ver.	**Date**	**Author**	**Description**
4.13.02	24.06.19	ts	3.2.2 Device properties • setPrinterWidth • getPrinterWidth
5.00.00	04.07.19	ts	Internal changes, it's now possible to connect and do transactions on multiple devices. Versions > 5.00.00 are not binary compatible anymore with Versions < 5.00.00
5.01.00	30.10.19	ts	3.16.3.3 ResourceText • new Text added 'Choose Function' 3.17.4.3 Dialog.DialogFlags • new flags for changing behavior of dialogues
5.02.00	23.09.2020	ts	3.17.4.3 Dialog.DialogFlags • new flags for overriding default idle texts
5.03.00	09.04.2024	ts	3.9.4.1 Transaction.Currency • added missing currencycodes, should now contain all ISO currencies as per ISO-4217

Table of contents

List of abbreviations

Abbreviation	Description
ECR	Electronic Cash Register
EFT	Electronic Fund Transfer, computer-based financial transaction
EMV	International specification for payment card, originally from Eurocard, Mastercard and Visa.
EP2	Swiss specification for processing of payment cards. Originally eft/pos 2000.
KIT	Cash register integrated driver
PAN	Primary account number
POS	Point of sale, where the transaction takes place

1 Introduction

1.1 Brief description

This document outlines the construction, configuration and functioning of the KIT cash register driver required to connect cash register systems (ECR) to Verdi+ terminals.

The KIT cash register driver enables connected Verdi+ terminals to be controlled from a cash register application. The KIT cash register driver is a part of cash register application and is thus not registered as a service in the operating system. With the help of the KIT, a cash register application can (amongst other things) carry out transactions, query the terminal status, or initiate terminal configuration.

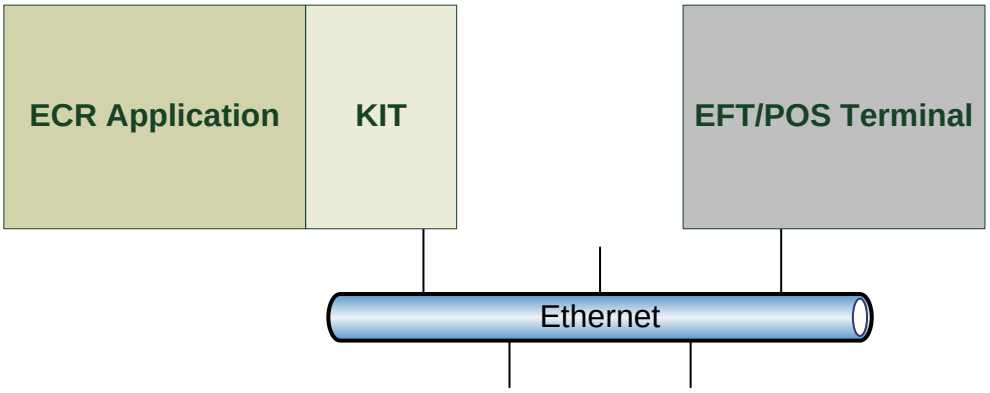
At present, the KIT only supports the KIT+ cash register interface, which was developed specifically for Verdi terminals. The printing of receipts is not supported: these are created by the cash register itself.

1.2 Document limitations

This document is designed for developers intending to integrate the KIT as a driver into a cash register (ECR) application. The functionalities are described in order to send transactions from a terminal to an EFT/POS terminal.

1.3 Overview

The graphical representation below shows the connection of KIT and Verdi+ terminal in a LAN. As a default, the Port 8307 is selected in the Verdi terminal for communication with the KIT.



2 Configuration

2.1 Introduction

The KIT must be configured for the connection of terminals with the cash register in the LAN. The configuration of the KIT driver is undertaken in the XML file. During the instantiation of the driver, an absolute path with the name of the file must be assigned. At this point, it is defined how the connected terminal will be addressed (parameter URL).

M – mandatory fields
O – optional fields

2.2 Configuration data parameters

2.2.1 Regular parameters

The following data indicates all possible parameters in the XML configuration data.

Node type	Value	M/O	Description
Element	Config	M	Root element
Attribute	Trace	O	Trace ON/OFF
Attribute	TraceOutput	O	Tracefile name and path
Element	Device	M	An element for each selected device
Attribute	DeviceId	M	Indication of the device, normally the TerminalId. The value can be queried via the function KITdriver.getDeviceIDFromConfig.
Attribute	URL	M	Information on protocol (kitp), terminal name and port via which the connected terminals can be reached.
Attribute	TONetwork	O	General timeout for network connections in seconds. The denoted value is at the same time the value for the interval of heartbeats between KIT and device. The default value for TONetwork is 5 seconds. The maximum timeout value is 20 seconds.
Attribute	TOInitTrx ToValidCard	O	Timeout between issuing of initialization of transaction and insertion of valid card. Default value is 60 seconds.
Attribute	TOValidCard ToAppSel	O	Timeout between insertion of valid card and either card application selected by terminal or waiting for card application selection by user. Default value is 3 seconds.
Attribute	TOAutoAppSel ToWaitTrxReq	O	Timeout between card application selected by terminal and waiting for transaction request from ECR (end of transaction init). Default value is 3 seconds.
Attribute	TOManAppSel ToWaitTrxReq	O	Timeout between waiting for card application selection by user and waiting for transaction request from ECR(end of transaction init). Default value is 10 seconds.

Node type	Value	M/O	Description
Attribute	TOGenApp	O	Timeout between waiting for the terminal to response on asynchronous request.Default value is 10 seconds. See also 6.1.1
Attribute	TOTrxRsp	O	Timeout between doTransaction and onDoTransaction. Default value is 90 seconds. See also 6.1.1
Attribute	TOInitRsp	O	Timeout for terminal to wait for response on initialize() Default value is 180 seconds. See also 6.1.1
Attribute	TOConfigRsp	O	Timeout for terminal to wait for response on configure() Default value is 30 seconds. See also 6.1.1
Attribute	TOTrxConfirmRsp	O	Timeout for terminal to wait for response on Transaction.rollback() or Transaction.commit(). Default value is 45 seconds. See also 6.1.1
Attribute	TOFinalBalanceRsp	O	Timeout for terminal to wait for response on finalBalance() Default value is 180 seconds. See also 6.1.1

If terminal does not response in defined time, onError with the information about the exceeded timeout will be called.

2.2.2 Debug parameters

For debug purposes, the following parameters can be set. Here, the behavior of some functions and return values can be overridden. The value of the flag Device.Debug has no influence on the parameters. Attributes are ignored when they are not present or contain an empty string. The values are read before each transaction is executed.

NodeType	Value	M/O	Description
Element	Debug	O	Root element for debug settings
Attribute	PostfinanceSimulation	O	ON/OFF. When ON, postal banking transaction types are simulated. Option has no influence on return values.
Element	Transaction	O	Element for transaction-specific debug values
Attribute	amountAvail	O	Overrides the value from the getter of the amountAvail field in transaction class
Attribute	accountNumber	O	Overrides the value from the getter of the accountNumber field in transaction class
Attribute	clientIdentifier	O	Overrides the value from the getter of the clientIdentifier field in transaction class
Attribute	authResult	O	Overrides the value from the getter of the authResult field in transaction class. Dependent on the authResult, the TransactionState is also set.

NodeType	Value	M/O	Description
Attribute	authRespCode	O	Overrides the value from the getter of the authRespCode field in transaction class
Attribute	amountAvailCurrC	O	Overrides the value from the getter of the amountAvailCurrC in transaction class
Attribute	idFlag	O	Overrides the value from the getter of the idFlag field in transaction class
Attribute	applicationCurrencyCode	O	Overrides the value from the getter of the applicationCurrencyCode in device class and provokes after every card insertion an application currency code available status
Attribute	SimulateGiro	O	If true, a purchase is executed via the function CreateGiro
Attribute	SimulateDeposit	O	If true, a cash transaction is executed via the function CreateDeposit
Attribute	SimulateCombined Transaction	O	If true, a purchase is executed via the function Create CombinedTransaction
Attribute	SimulateAccountInquiry	O	If true, a purchase transaction is executed via the function CreateAccount Inquiry
Attribute	SimulateDeposit	O	If true, a credit transaction is executed via the function CreateDeposit
Attribute	SimulateClientId Request	O	If true, a purchase is executed via the function CreateClientId Request

Example

```
<Config
      Trace="ON"
      TraceOutput="C:\Temp\KITLog.txt">
      <Device DeviceId="30140048" URL="kitp://ep2PT30140048:8307"
            TONetwork="10"                TOInitTrxToValidCard="50"
            TOValidCardToAppSel="3"       TOAutoAppSelToWaitTrxReq="5"
            TOManAppSelToWaitTrxReq="10"  TOGenApp="10"
            TOTrxRsp="90"                 TOInitRsp="180"
            TOConfigRsp="30"              TOFinalBalanceRsp="180"
            TOTrxConfirmRsp="45"/>
      <Debug PostfinanceSimulation="ON">
            <Transaction amountAvail="5000000"
                  accountNumber="98798798797"
                  clientIdentifier="887744"
                  authResult="105"
                  authRespCode="76"
                  amountAvailCurrC="CHF"
                  idFlag="true"
                  SimulateAccountInquiry="true"
                  SimulateClientIdRequest="true"
                  SimulateDeposit="true"
                  SimulateGiro="false"
                  SimulateCombinedTransaction="true"/>
            <Device applicationCurrencyCode="756"/>
      </Debug>
</Config>
```

2.3 Communication

The URL parameter indicates how the KIT driver communicates with the terminal. The URL parameter is made up of the three parts: protocol, device and port.

URL part	Description	Example
Protocol	The protocol identification of the cash desk interface used to communicate with the terminal. Only the KIT+ interface(`kitp`) is currently supported.	kitp://
Device	IP address or terminal's host name the KIT is to be connected to	ep2pt30140048 192.168.1.114
Port	The port number which the terminal receives the KIT requests. The KIT+ interface only supports port 8307.	8307

3 KIT Classes

3.1 Class KITDriver

The KITDriver object is the first object that needs to be created in order the rest of the classes. The implementation of the KIT driver is illustrated in separate sample projects for .NET or JAVA.

3.1.1 KITDriver methods

Method	Description
Device **createDevice**()	Creates a device object for the device, which is defined as the first in the configuration file.
String **getDeviceIDFromConfig()**	Returns the TerminalId of the first device defined in the file. This value represents the DeviceId in the configuration data, and not necessarily that of the connected device.
boolean **isATRFromB0**(byte[] ATR)	Returns TRUE if the ATR is an answer from B0 card.

3.1.2 KITDriver properties

The driver object provides the following attributes.

Property	Type	Description
getConfigFilePath	String	Returns path and name of configuration file
isDebug	Boolean	Returns the status of whether the debug mode is activated
setDebug	Boolean	Turns debug mode on or off
setDebugLevel	Integer	Sets the level of debug output: 1-only errors, 5-detailed verbative output
getDevices	List<Device>	Collection of all available device objects
getVersion	String	Returns the KIT driver version number

3.1.3 KITDriver constructor

In order to initialize a new class of KITDriver, the following constructor is made available.

Name	Description
KITDriver(DeviceCallbacks, TransactionCallbacks, String configFilePath)	Initializes new instance of KITDriver class with callback interface for devices and transactions, and the configuration file path. The ConfigFilePath parameter contains the absolute path and data name.
KITDriver(DeviceCallbacks, TransactionCallbacks, InputStream inputStream)	Same as above, but the config data comes from the input stream.

3.2 Class *Device*

The device object is an instance of the connected EFT/POS terminal. It offers various methods for the purposes of carrying out actions on the terminal.

3.2.1 Device methods

The device object provides the following methods. Unless otherwise indicated, all methods are asynchronous. The result of the asynchronous request is sent back to the requesting application via the interface *DeviceCallbacks*. Each asynchronous method possesses its own confirmation event. This is indicated in the *CallbackEvent* column.

Method	Description	CallbackEvent
void **closeShift()**	Closes the cash desk shift. No further transaction delivery.	onCloseShiftResponse()
void **configure()**	Executes device configuration (EP2-SIConfig). It configures among other things also acquirers. If needed the terminal can also schedule an initialization internally after a config. This operation takes place also automatically in the terminal according to the information sent from service center, however no callback is called in this case.	onConfigure()
void **connect()**	Establishes connection to terminal. Definition from configuration data used as connection parameters.	onConnectResponse()
void **disconnect()**	Disconnects connection to terminal.	synchron
void **doTransaction(** Transaction**)**	Executes transaction on terminal. Function requires transaction object as parameter	onDoTransactionResponse(Transaction)
void **initTransaciton()**	Initializes transaction on terminal. It shows an animation requesting insertion of card. It allows acquiring of some card data before issuing of doTransaction command. Details are described in Class TransactionInit	onInitTransaction(TransactionInit)
void **abortTransaction()** void **abortTransaction(AbortType)**	Aborts transaction execution immediately. Already authorized transaction will be reversed if not yet sent. It also aborts a transaction initialization.	no callback is necessary as the handling in all cases will be done automatically without any further information needed from terminal
void **ejectCard()**	Ejects card from terminal. Function can be called in both transaction and dialog mode. The mode will be not changed.	onEjectCardResponse()
void **ejectCardForced()**	Ejects card from terminal. Function can be called anytime and bypasses all checks. It merely gives the motor in the card reader a command to turn on for certain period of time necessary to eject the card completely.	onEjectCardForcedResponse()
void **reboot()**	reboots the terminal immediately, this can cause also a SW update if there is a new SW Version ready for download defined in terminal's service center, this command will lead to socket error in the KIT, because the device will terminate the connection	no callback will be called as the terminal is not responsive anymore after this command for at least 1 minute (best case without actual sw update)
void **swUpdate()**	Terminates the payment application of the device and starts the service application, which can perform a Terminal SW Update if this is necessary according to service center, this command will lead to socket error in the KIT, because the device will terminate the connection	no callback will be called as the terminal is not responsive anymore after this command for at least 1 minute (best case without actual sw update)
void **enableLanguageSelection()**	Offers language selection on the terminal if the card's language is not recognized	onChangeLanguageSelection(sel ectionEnabled)

Method	Description	CallbackEvent
void **disableLanguageSelection()**	Offers no language selection on the terminal if the card's language is not recognized	onChangeLanguageSelection(selectionEnabled)
void **finalBalance()**	Executes day-end closure and sends transactions. Shift will be automatically closed if not already so.	onFinalBalanceResponse()
void **requestDeviceStatus()**	Requests current device status	onDeviceStatusResponse (DeviceStates)
void **initialize()**	Initializes the configured acquirer(s) and receives defined brand initialization data.	onInitializeResponse()
void **openShift()**	Opens cash register shift	onOpenShiftResponse()
void **requestConfirmationTimeExtension(int seconds)**	Request a confirmation time extension from the terminal. Attended terminals have a default timeout of 30s and unattended terminals of 60s. The number of seconds will be the new timeout for a confirmation. Seconds must be between 0 and 30.	none
void **requestReceipt(**Receipt.ReceiptType receiptType, int receiptIDNumeric**)**	Requests a receipt from the terminal. In case the receipt type is Transaction receiptedNumeric is the Transaction Sequence Counter of the wanted transaction.	onReceiptResponse()
void **requestReport**(Report.ReportType reportType)	Request a report from the terminal.	onReportResponse()
void **setTerminalLanguage(string language)**	Set the default language of the terminal. Default language is the language used for receipts, attendant user interfaces and for the cardholder user interface as long as no language code from a card is in use. The supplied value is kept until the terminal reboots or reconfigures itself with the TMS (ServiceCenter) which changes back the default language to the value configured in the TMS. Specify language as an lowercase ISO-639-1 letter code, e.g. 'en'.	none
void **startUIExcl ()**	Places terminal in mode for exclusive display of defined dialogs via showDialog(). Payment transactions not possible in this mode. Transaction mode is default mode explicitly exited with this function	onStartUIExclResponse()
void **stopUIExcl()**	Immediately terminates mode for exclusive display of defined dialog similarly to abortTransaction() function	onStartUIExclResponse()
void **showDialog**(Dialog dialog, Dialog defaultDialog)	Shows a screen with the values defined in the object dialog. A default dialog can be supplied for display if main dialog is ended.	onDialogResponse(DialogResponse)
void **abortDialog()**	Closes the currently displayed dialog and displays the default dialog.	onDialogResponse(DialogResponse)
void **addToBasket**(LoyaltyProductRecord)	This method instructs the terminal to add an item to the basket of the products as being sold to make a possible match in the loyalty processing. This basket will be used for the next transaction and must be fille prior to doTransaction	onAddToBasketResponse()
void **clearBasket**()	This method instructs to delete all items from the current basket.	

3.2.2 Device properties

The device object provides the following properties. Unless otherwise stated, the values are made available after the Device.connect() request.

Property	Type	Description

Property	Type	Description
isActivated	boolean	Indicates current shift status. (TRUE=shift open)
getActSeqCnt	long	Counter for shift activations
getAppPAN	string	Returns the AppPAN if a magstripe card (Track 2) is inserted and the terminal exposes the Track 2 data to the ECR (myOne cards)
getAppExpDate	string	Returns the expiry date if Track 2 data is exposed to the ECR by the terminal (myOne cards)
getMyOneBranchenCode	Int	Returns the Branchencode if Track 2 data is exposed to the ECR by the terminal (myOne Cards)
getATR	byte[]	Answer to Reset-ATR as per ISO-7816-10. Contains value of last successfully read chip or *null* if a chip has never been read.
getIIN	string(6)	First 6 characters of magnetic strip BIN range. Contains value of currently present magnetic stripe or *null* if no magnetic stripe present or cannot be read.
getAID	string	returns application identifier od the card being processed, null if currently not available, this value is available only if devices.states contains APPLICATION_SELECTED
getAppCurrC()	int	The application currency code. If none available it returns Device.UNKNOWN. This method should be called only if the status is OK.
getCashier, setCashier	Int	Enables cashier number to be set
isConnected	boolean	Returns status of connection to device
getDeviceId	string	Identification of connected device. This is normally the TerminalId.
getEcrId, setEcrId	string	Optional identification of connected cash register (ECR)
getHost	string	Name of host indicated in URL, e.g. ep2pt30140048
getLanguage	string	ISO639 language code for EFT-GUI display
getLastCleaningDate	date	Date (no time) of the last cleaning procedure of the card reader.
getPeSeqCnt	long	Counter for number of booking periods. This is raised at each day-end closure.
getPort	Int	Port number on which the terminal receives connections
getProtocol	string	Cash register protocol used by KIT.
getRecentDeviceStates	Device .State[]	Returns status of terminal at the time of last communication with the terminal.
getAcquirers	Acquirer[]	Returns information about all configured acquirers as sent in configuration session from terminas service center.
getRecentDeviceStatesText	string	Returns status of terminal as text value, according to last status request on terminal. Various statuses separated by space. No special status is indicated by NONE.
getSerialNo	string	Serial number of terminal
getSoftwareVersion	string	Software version number installed on terminal.
getSupportedCurrencies	Transaction. Currency[]	Returns all currencies supported by terminal. Assigned a value upon activation of openShift().
getSupportedCurrenciesText	string	Returns the supported currencies as string, separated by gaps. Assigned a value upon activation of openShift().
getSupportedFunctions	Transaction. Function[]	Returns all transaction functions which are supported by the terminal. Assigned a value upon activation of openShift().
getActivatedBrands	Brand[]	Returns information about all brands initialized in the terminal. If the shift is currently closed, this field is empty.

Property	Type	Description
getToNetwork **setToNework**	int	Timeout for network connections and heartbeat between KIT and terminal. Value also used as socket timeout between two TCP packets. In order for a newly set timeout to become active, the terminal must be reconnected via Connect().
isUIExclusive	Boolean	Returns true, when terminal in exclusive dialog mode.
getESignatureActive	Device. ESignatureActive	Returns true, when terminal requests signature on the touchscreen display in case when a transaction must be verified with signature. This flag is available after first status response from the device (spontaneous or requested). See also 5.7.1.
addTransactionDatachangeReason **removeTransactionDatachangeReason**	Transaction. DatachangeReason	Add/Removes reasons for the terminal, which should trigger transaction data change request during the transaction. When the reason occurs OnTransactionDatachangeRequest callback will be triggered.
setTransactionDataChangeReasons **getTransactionDataChangeReasons**	Transaction. DatachangeReason[]	Sets/Gets reasons for the terminal, which should trigger transaction data change request during the transaction.
setReportIINZeroCards **hasReportIINZeroCards**	Boolean	If ReportIINZeroCards is set to TRUE, magstripe cards with IIN (Issuer Identification Number) '000000' are kept in the reader. The cardnumber (AppPAN) is reported in the status notification or can be read through device.getAppPAN. The transaction can be aborted via ECR or [STOP] key. Currently this is used to check myOne Cash cards numbers. This must be switched to TRUE or FALSE before connecting to the device. Changes will take effect after reconnect.
setPrinterWidth **getPrinterWidth**	Int	Gets or sets the value of the receipts generated by the terminal. Default value is set to 34 characters.
setPartialApprovalAllowed **getPartialApprovalAllowed**	Boolean	If the ECR supports "Partial Approval Transactions" set this value to true after connecting to the device. This function must be supported by the device and by the acquirer as well.
getCardHolderLanguage	String	Returns the default language of the customers card.
getTrack2Data	String	Returns the complete track2 as a string for whitelisted, non-EMV cards only!

3.2.3 Device enumerations

3.2.3.1 Device.State

Whenever a change of status occurs on the terminal, the respective information is sent automatically. This is registered in the event onDeviceStatusResponse(). As an argument, this event contains the field DeviceState with the current terminal status, and the event may contain a number of statuses combined together. The individual statuses are explained below:

Constant	Description
ACTIVATED	Terminal is active, shift is open
BUSY	Terminal is currently busy with a non-EFT action (receipt printing, autocancel, UIExclModus etc.)
READER_SLOT_OCCUPIED	Card is partially or completely entered. Value is independent of VALID_CARD_INSERTED
VALID_CARD_INSERTED	Valid card inserted in terminal reader

LOCKED	Terminal is locked with an EFT operation (transaction, initialization, specific custom dialog mode etc.)
APPLICATION_SELECTED	Terminal has recognized a card as an EMV card, and has selected an application for further processing
WAITING_FOR_TRANSACTION_REQUEST	Terminal has recognized a card, could read all the needed data from it and is waiting for transaction request. The request for transaction comes also if there is a card, which needs the amount before an application can be chosen. If a Trm SW Version does not support init transaction functionality this status will be issued simultaneously with VALID_CARD_INSERTED and ACC value is Device.UNKNOWN or value from configuration file.
WAITING_FOR_APPLICATION_SELECTION	Terminal has recognized a card, could read all the data up to the point where a card application must be chosen by user. If a Trm SW Version does not support init transaction functionality, this status won't be issued.

3.2.3.2 Device.ESignatureActive

Constant	Description
ON	Terminal displays a dialog to draw a signature on the touchscreen if signature verified transaction occurs
OFF	Terminal does not display a dialog for e-signature if signature verified transaction occurs
UNKNOWN	Terminal haven't get the information about the e-signature handling from terminal management system yet or Trm SW of currently connected terminal does not support this information

3.2.3.3 Device.AbortType

Constant	Description
SILENT	Aborts the transaction without a notification dialog on the terminal.

3.3 Class Acquirer

This class holds information about the acquirer. Every acquirer can initialize more brands. Currently KIT provides no direct association between brands and acquirers.

3.3.1 Acquirer properties

Property	Type	Description
getID	int	The acquirer ID as registered with ep2 (for the list of acquirers contact the partner who provides the terminal)
getLastInitDate	date	Local Time and Date of the last successful initialization of the acquirer. This can be also null if the acquirer was never initialized.

3.4 Class Brand

This class holds information about the brand. Brand information will be initialized during initialization session with particular acquirer. The exact explanation of the term Brand can be found in ep2 documentation.

3.4.1 Brand properties

Property	Type	Description
getName	string	The brand name as sent by acquirer.

3.5 Class Receipt

This class exposes receipt data requested from terminal.

3.5.1 Receipt properties

Property	Type	Description
getType	ReceiptType	The receipt type as requested or ReceiptType.None if the receipt could not be returned.
getReceiptIDNumeric	int	Numeric ID of the receipt, it is normally the transaction sequence counter, obtained by Transaction.getTrxSeqCnt()
getText	string	Text of the receipt. The text is a set of lines separated with platform specific end of line separator.
isReceiptComplete	boolean	Normally is this flag always true when the receipt object is returned to API in callback.

3.5.2 Receipt enumerations

3.5.2.1 Receipt.ReceiptType

Constant	Descriptions
CONFIGURATION	Receipt of the last configuration executed on terminal.
FINAL_BALANCE	Receipt of the final balance of the last period. If there is currently an opened shift, the unclosed running balance of the period will be returned.
INITIALIZATION	Information about the last initialization of the terminal.
NONE	If a receipt request cannot be returned, because the receipt of the requested type is not available, the type of returning receipt will be changed to NONE and the text is empty.
SHIFT_REPORT	Receipt of the last shift (either running of closed if shift is closed)

Constant	Descriptions
TRANSACTION	Tranasction receipt. The last transaction will be returned if the ReceiptIDNumer holds the value ***Receipt.receiptIDLastAvailable***. Otherwise a specified transaction will be returned. There will be a placeholder with tag <ESignature /> on a separate line if the transaction was made with signature (either electronic or paper).
TRANSACTION_CARDHOLDER	The same as transaction receipt except this is the copy for cardholder.

3.6 Class Report

This class exposes report data requested from terminal. Report can be constructed within application only with a given report type. Every report consists of one or more report entries.

3.6.1 Report properties

Property	Type	Description
getType	ReportType	The report type as requested.
getReportEntries **setReportEntries**	ArrayList<ReportEntry>	In the callback will be returned the list of report entries as returned from the terminal. For the convenience the list of report entries can be changed in an existing report.
isReportComplete	Boolean	Normally is this flag always true when the report object is returned to API in callback.

3.6.2 Report enumerations

3.6.2.1 Report.ReportType

Constant	Descriptions
SUM_OF_SHIFT	This report will list all transactions of current shift or last closed shift.
SUM_OF_ACCOUNTING_PERIOD	This report will list all transactions starting from the last final balance.

3.7 Class ReportEntry

This class exposes single report entry contained in report. This class is just a structured container. It containes no logic exposed to the API. The transactions are grouped into report entry. A report entry represents the sum of transactions amounts per AID, brand, currency, DCC foreign currency and financial counter type. It means it is a sum of amount for transactions where the above mentioned attributes are the same.

3.7.1 ReportEntry properties

Property	Type	Description
getAid **setAid**	byte[]	Application identifier of cards in transactions summed in this entry
getBrand **setBrand**	string	Brand name of cards used in transactions summed in this entry
getCount **setCount**	int	Number of transactions in this entry
getCurrency **setCurrency**	Transaction.Currency	Currency of the cards used in transactions summed in this entry
getDccForeignAmount **setDccForeignAmount**	long	Sum of the transaction amounts in foreign currency
getDccForeignCurrency **setDccForeignCurrency**	Transaction.Currency	Currency effectively used in transaction in case a foreign currency card was used
getFinancialCounterType **setFinancialCounterType**	ReportEntry.FinancialCounterType	Financial counter type of the transactions summed in this entry
getTotalAmount **setTotalAmount**	long	Total sum of amounts of the transactions in this entry

3.7.2 ReportEntry enumerations

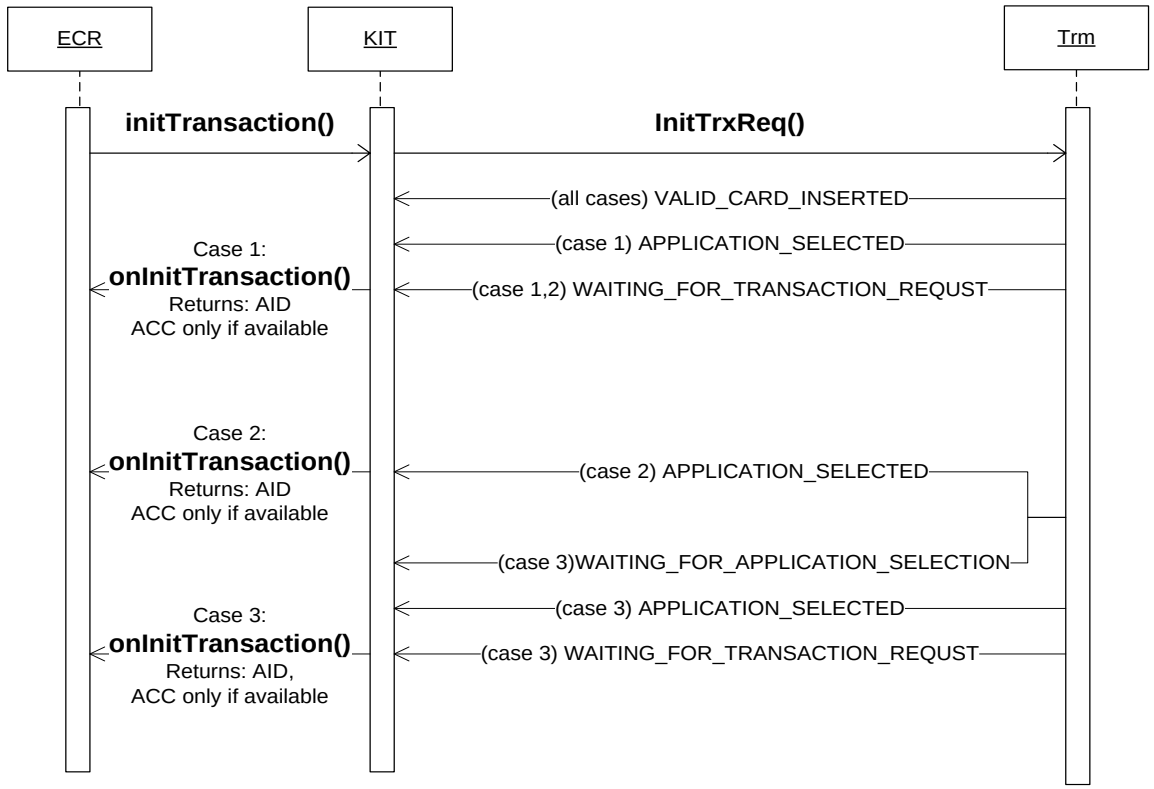
3.7.2.1 FinancialCounterType

The financial counter types are from the sight of cash register.

Constant	Descriptions
DEBIT	The amount was debited to the merchant (amount has + sign)
CREDIT	The amount was credited to the merchant (amount has – sign)
REVERSAL_DEBIT	The amount was reversedly debited it means it was credited to merchant (amount has – sign)
REVERSAL_CREDIT	The amount was reversedly credited it means it was debited to merchant (amount has + sign)

3.8 Class TransactionInit

The class which exposes data acquired during initialization of a transaction. According to EMV Level 2 not all parameters are available for every inserted card although it is a valid card. The changing of the device states and availability of some card parameters after initTransaciton are described below.



Case 1: Trm could automatically select card application
Case 2: Trm needs amount to select the application
Case 3: Trm needs the cardholder to select the application manually

3.8.1 TransactionInit enumerations

3.8.1.1 TransactionInit.Status

This describes the status of the initialization.

Constant	Description
OK	Initialization went ok and is ended.
TIMEOUTED	Transaciton initialization hit a timeout. The description of the timeout is in TransactionInit.Timeout
ABORTED	Transaction initialization was aborted either by user or from ECR.
PENDING	Transaction initialization is pending in process

3.8.1.2 TransactionInit.Timeout

Constant	Description
INIT_TRX_TO_VALID_CARD	Timeout between issuing of initialization of transaction to insertion of valid card

Constant	Description
VALID_CARD_TO_APP_SEL	Timeout between insertion of valid card and either card application selected by terminal or waiting for card application selection from user
AUTO_APP_SEL_TO_WAIT_TRX_REQ	Timeout between card application selected by terminal and waiting for transaction request from ECR (end of transaction init)
MAN_APP_SEL_TO_WAIT_TRX_REQ	Timeout between waiting for card application selection by user and waiting for transaction request from ECR(end of transaction init)

3.8.2 TransactionInit properties

Property	Type	Description
getStatus()	TransacitonInit.Status	The status of the transaction initialization as described in 3.8.1.1
getAppCurrC()	int	The application currency code. If none available it returns Device.UNKNOWN. This method should be called only if the status is OK.
getAID()	string	The application ID. If none available it returns null. This method should be called only if the status is OK.
getTimeout()	InitTransaction.Timeout	The timeout which caused TIMEOUTED status. This field is null if no timeout.

3.9 Class Transaction

The data of the executed transaction are saved in the transaction object, and can be accessed at any time.

3.9.1 Asynchronous transaction methods

The transaction object contains the following methods for asynchronous transaction processing. The confirmation of the asynchronous activation occurs in the interface *TransactionCallbacks*. Each of the methods in this interface has its own callback event (see right-hand column).

Method	Description	CallbackEvent
void **commit()**	Confirms the transaction and allows it to be submitted to the acquirer.	OnCommit(Transaction)
void **rollback()**	Rollbacks the transaction, if the transaction was already submitted a transaction reversal is issued automatically. Transaction object will be update with the latest Cardholdertext, AttendantText and AuthReslt as defined in EP2.	OnRollback(Transaction)

3.9.2 Static transaction methods

The transaction class contains the following static methods for the purposes of creating a transaction object with the corresponding transaction type and with the mandatory parameters. The amount must always be given in the smallest unit of currency (e.g. centimes).

Method	Description
Transaction **createAccountInquiry()**	Creates a transaction object for transaction function *AccountInquiry*
Transaction **createActivateCard()**	Creates a transaction object for transaction function ActivateCard (only available in ep2 V6.0+)
Transaction **createAuthorizationPurchase (**Transaction.Currency, long amount**)**	Creates a transaction object for transaction function A*uthorizationPurchase*
Transaction **createAuthorizationDeposit()**	Creates a transaction object for transaction function *AuthorizationDeposit* (only if postfunctions available)
Transaction **createAuthorizationCredit()**	Creates a transaction object for transaction function *AuthorizationCredit* (only if postfunctions available)
Transaction **createCashAdvance(**Transaction.Currency, long amount*, boolean multiyCurrencyFlag***)**	Creates a transaction object for transaction function *CashAdvance*. Currency flag can be given optionally.
Transaction **createClientIdRequest()**	Creates a transaction object for transaction function *ClientIDRequest*
Transaction **createCombinedTransaction (**Transaction.Currency, long amount, boolean multiyCurrencyFlag**)**	Creates a transaction object for transaction function *CombinedTransaction*
Transaction **createConfirmPhoneAuthorizedReservation (**Transaction.Currency, long amount**)**	Creates a transaction object for transaction function *ConfirmPhoneAuthorizedReservation*
Transaction **createCredit(**Transaction.Currency, long amount*, boolean multiCurrencyFlag***)**	Creates a transaction object for transaction function *Credit*. Currency flag is optional
Transaction **createDeposit (**Transaction.Currency, long amount, String[] recPostalBanking*, boolean multiyCurrencyFlag***)**	Creates a transaction object for transaction function *Deposit*. Currency flag is optional
Transaction **createGiro (**Transaction.Currency, long amount, boolean multiCurrencyFlag**)**	Creates a transaction object for transaction function *Giro*

Method	Description
Transaction **createPurchase(**Transaction.Currency, long amountAuth*, boolean multiCurrencyFlag*)**	Creates a transaction object for transaction function *Purchase*. Currency flag is optional
Transaction **createPurchaseForcedAcceptance (**Transaction.Currency, long amount**)**	Creates a transaction object for transaction function *PurchaseForcedAcceptance*
Transaction createLoadCard (Transaction.Currency, long amount)	Creates a transaction object for transaction function LoadCard (only available in ep2 v6.0+)
Transaction **createPurchaseMail Ordered(**Transaction.Currency, long amount**)**	Creates a transaction object for transaction function *PurchaseMailOrdered*
Transaction **createPurchasePhoneAuthorized (**Transaction.Currency, long amount, string authCode**)**	Creates a transaction object for transaction function *PurchasePhoneAuthorized*
Transaction **createPurchasePhoneOrdered (**Transaction.Currency, long amount**)**	Creates a transaction object for transaction function *PurchasePhoneOrdered*
Transaction **createPurchaseReservation (**Transaction.Currency, long amountAuth, string trxRefNum**)**	Creates a transaction object for transaction function *PurchaseReservation*
Transaction **createPurchaseWithCashback (**Transaction.Currency, long amountAuth, long amountOther**)**	Creates a transaction object for transaction function Cashback
Transaction **createReservation (**Transaction.Currency, long amount**)**	Creates a transaction object for transaction function *Reservation*
Transaction **createReservationAdjustment (**Transaction.Currency, long amount, string trxRefNum**)**	Creates a transaction object for transaction function *ReserverationAdjustment*
Transaction **createReversal()**	Creates a transaction object for transaction function *Reversal*
Transaction **createTip(**Transaction.Currency, long amount**)**	Creates a transaction object for transaction function *Tip (is deprecated in ep2 5.0.x and higher)*

3.9.3 Transaction properties

The transaction object contains the following properties.

Property	Method	Description
getAccountNumber	string	Account number for account transfer if via post
getAid	string	Identification of application as per EMV as hexadecimal digits
getAmountAuth setAmountAuth	long	Authorization amount of transaction in smallest unit (e.g. centimes)
getTipAmt	long	Tip as entered by cardholder during transaction authorization. Contactless transactions do not support tip entry currently.
getAmountAvail	long	Available account balance
getAmountAvailCurrC	Transaction . Currency	Currency code of available account balance
getAppExpDate setAppExpDate	string	Expiry date of current transaction card in format year/month/day: YYMMDD. Must be set before activation of doTranscation if POSEntryMode= POS_ENTRY_MANUAL.

Property	Method	Description
getAppPAN **setAppPAN**	string	Credit card number, maximum 19 digits. Returned by device when POSEntryMode=UNSPECIFIED or POSEntryMode=MAGSTRIPE Must be set before activation of doTranscation if POSEntryMode= POS_ENTRY_MANUAL.
getAppPANEnc	string	Encrypted credit card number
getAppPANPrtAttendant	string	Credit card number formatted to be printed on retailer receipt
getAppPANPrtCardholder	string	Credit card number formatted to be printed on card holder receipt
getAttendantText	string	Text to be shown to cashier
getAuthCode	string(6)	Authorization code of the issuer for current transaction
getAuthRespCode	string	Authorization response code as per EP2 and issuer. Code <76> if currency-flawed business and a transaction currency unequal to account currency
getAuthResult	int	EP2-resulting value of the acquirer for authorization request
getBrand	string	Identifies the brand of the current transaction
getCardholderText	string	Text to be displayed to the card holder
getCurrency	Transaction.Currency	Transaction currency as per ISO-4217 (for values see 3.9.4.1)
getClientIdentifer	string	Serial number of post client
getCVC2 **setCVC2**	string	Card verification code 2 only used for transactions if manual entry of card data occurs (transaction must be started with POSEntryMode= POS_ENTRY_MANUAL)
getCVMResults	Transaction.CVMResults	Cardholder verification method information (value see 3.9.4.6) If Trm SW does not support this functionality the value is always null.
getDisplayName	string	Name of card as defined by card issuer, as to be displayed to card holder
getEcrSequence	int	Cash register-internal sequence number
getESign	byte[]	Electronic signature for transaction as PNG image
getExtendedResult	Transaction.ExtendedResult	Detailed transaction result as specified in EP2 or EMV standards. If Trm SW does not support this functionality the value is always null.
isCUPTransaction	boolean	Returns true when a China UnionPay (CUP) transaction is processed
isIdFlag	boolean	Swiss post banking specific flag which controls printing of the signature field on the post receipt
isMultiCurrencyFlag, **setMultiCurrencyFlag**	boolean	When true, it is a mixed-currency transaction
getPosEntryMode **setPosEntryMode**	Transaction.POSEntryMode	Mode used for card data entry. (Values see 3.9.4.3) For setting only POS_ENTRY_MANUAL and POS_ENTRY_UNSPECIFIED are allowed
getRecPostalBanking, **setRecPostalBanking**	String[]	Message line when paying into own account. Returns NULL (zero) when not set.
getRefNum	long	A number specific to EFT devices used for booking of tips and credits
getTrack2	string	Track2 data of card's magnetic stripe
getTrack3	string	Track3 data of card's magnetic stripe
getTransactionFunction	Transaction.Function	Type of transaction (value see 3.9.4.5)
getTransactionStatus	Transaction.Status	Status of transaction (value see 3.9.4.4)
getTrxRefNum	string(11)	TransactionReference number
getTrxSeqCnt	int	TransactionSequence counter
getTrxSeqCntOri	int	TransactionSequence counter of original transaction by cancellation
getTrxTime	date	Date and time of transaction execution
getVoicePhone	string	Telephone number of acquirer bank for telephone authorization

Property	Method	Description
getCardNumber	string(19)	The card number from the issuing system, it can be truncated
getCardID	string(17)	The Card ID of from the Issuing system
getIsPartiallyApproved()	boolean	Returns "true" if the transaction was partially approved, otherwise false
getAmountRemaining()	long	Returns the remaining amount if the transaction was partially approved. For all other transaction contains the value AmountAuth
getSurrogatePAN()	Byte[]	Gets a hash value of the PAN for identifying recurring customers/cards (petrol use case print receipt where the card is presented a second time)
getListSurrogatePANs() **setListOfSurrogatePANs()**	ArrayList<byte[]>	Sets/Gets a list of surrogate PANs for a transaction for the terminal to check when a new card is inserted (petrol use case with double present for receipt printing)
getTransactionRequestFlag() **setTransactionRequestFlag**	Transaction.TransactionRequestFlags	Get/sets flag for transaction as defined in enumeration Transaction.TransactionRequestFlags

3.9.3.1 Transaction property validity

Flag describes when and who sends the information contained in this property. The meaning of the scope flag is as follows:

imp – set implicitly by application during creation of transaction object

app – application sets the property

trm – terminal changes the property

req – must be set before doTransaction

rsp – valid after onDoTransaction

approved – valid only if transaction status approved

aborted – valid only if transaction status aborted

not aborted – valid only if transaction status not aborted

magstripe – valid only if magstripe ok

icc – valid only if chip ok

post – only valid if swiss postal banking transaction

opt – is optional, for further information see EP2 and EMV specification

tdc – valid only if transaction was returned in TransactionDatachange object

Property	Validity
getAccountNumber	trm,rsp,approved,post
getAid	trm,rsp,approved
getAmountAuth **setAmountAuth**	imp,rsp(only if approved) tdc
getAmountAvail	trm,rsp,approved,opt
getAmountAvailCurrC	trm,rsp,approved,opt
getAppExpDate **setAppExpDate**	trm,rsp(only if POSEntryMode = UNSPECIFIED or MAGSTRIPE), opt app,req(only if POSEntryMode = MANUAL_ENTRY)
getAppPAN **setAppPAN**	trm,rsp, valid only if transaction status REFERRED app,req (POSEntryMode = MANUAL_ENTRY)
getAppPANEnc	trm,rsp,approved
getAppPANPrtAttendant	trm,rsp,approved

Property	Validity
getAppPANPrtCardholder	trm,rsp,approved
getAttendantText	trm,rsp,opt
getAuthCode	trm,rsp,imp(only as follow up trx after trx with status=REFERRED)
getAuthRespCode	trm,rsp,approved
getAuthResult	trm,rsp
getBrand	trm,rsp,approved
getCardholderText	trm,rsp,opt
getCurrency	imp,req,trm,rsp,opt(mandatory except for Reversal,Acc Inquiry,ClientID req)
getClientIdentifer	trm,rsp,opt,post
getCVC2 **setCVC2**	app,rsp (POSEntryMode = MANUAL_ENTRY) req
getCVMResults	trm,rsp
getDisplayName	trm,rsp
getEcrSequence	req,rsp
getESign	trm,rsp,opt (see 5.7.1)
getExtendedResult	trm,rsp
isIdFlag	trm, req,post
isMultiCurrencyFlag, **setMultiCurrencyFlag**	imp,req,rsp,post
getPosEntryMode, **setPosEntryMode**	trm,rsp,approved app,req
getRecPostalBanking, **setRecPostalBanking**	imp,req,post
getRefNum	imp,req
getTrack2	trm,rsp (POSEntryMode = UNSPECIFIED or MAGSTRIPE)
getTrack3	trm,rsp (POSEntryMode = UNSPECIFIED or MAGSTRIPE)
getTransactionFunction	imp,req (implicitly determined by the create* method)
getTransactionStatus	trm,rsp
getTrxRefNum	imp,trm,rsp,opt (only if trx function was RESERVATION)
getTrxSeqCnt	trm,rsp,approved
getTrxSeqCntOri	trm,rsp, opt(comes only if reversal)
getTrxTime	trm,rsp,not aborted
getVoicePhone	trm,rsp,opt
getCardNumber	trm,rsp,post (only when trx function in AUTHORIZATIONCREDIT, AUTHORIZATIONDEPOSIT), approved
getCardID	trm,rsp,post (only when trx function in AUTHORIZATIONCREDIT, AUTHORIZATIONDEPOSIT), opt (only if Postfinance – CardPay), approved
getIsPartiallyApproved()	trm,rsp,approved
getAmountRemaining()	trm,rsp,approved
getSurrogatePAN()	trm, rsp, approved, aborted (if card is recognized)
getListSurrogatePANs() **setListOfSurrogatePANs()**	app app, req
getTransactionRequestFlags() **setTransactionRequestFlags()**	app app, req

3.9.4 Transaction enumerations

3.9.4.1 Transaction.Currency

The value indicates ISO codes and currency information. The ISO-4217 is publicly available, therefore we list only the most often used currencies in the table below.

Constant	Value	Description
CHF	756	Numerical constant for Swiss francs as per ISO-4217

EUR	978	Numerical constant for Euros as per ISO-4217 currency code
USD	840	Numerical constant for US dollars as per ISO-4217 currency code
CHW	948	Numerical constant for WIR swiss francs as per ISO-4217 currency code

3.9.4.2 Transaction.ExtendedResult

This value contains those detailed transaction results. These values can be consulted for further information about the outcome a transaction. **Use Transaction.Result for taking decisions after an outcome of a transaction**. For the detailed description of the codes see the EP2 and EMV specifications.

Constant	Description
UNSPECIFIED	transaction result sent by the terminal could not be recognized
TRX_RESULT_APPROVED	transaction approved
TRX_RESULT_APPROVED_EP2_ONLINE	transaction online approved (ep2 only)
TRX_RESULT_APPROVED_EP2_ONLINE_DUPLICATE_OPERATION	transaction approved
TRX_RESULT_APPROVED_EP2_ONLINE_ORIGN_RES_DATA_INCLUDED	transaction approved
TRX_RESULT_REFERRED	transaction referred
TRX_RESULT_DECLINED_EMV_FIRST_TAA	Transaction declined during first Terminal Action Analysis
TRX_RESULT_DECLINED_EMV_FIRST_CAA	Transaction declined during first Card Action Analysis
TRX_RESULT_DECLINED_EMV_SECOND_TAA	Transaction declined during second Terminal Action Analysis
TRX_RESULT_DECLINED_EMV_SECOND_CAA	Transaction declined during second Card Action Analysis
TRX_RESULT_DECLINED_EMV_CDA_FAILED_FIRST_CAA_TC	CDA failed during first Card Action Analysis; returned cryptogram was a TC
TRX_RESULT_DECLINED_EMV_CDA_FAILED_FIRST_CAA_ARQC	CDA failed during first Card Action Analysis; returned cryptogram was an ARQC
TRX_RESULT_DECLINED_EMV_CDA_FAILED_SECOND_CAA	CDA failed during second Card Action Analysis
TRX_RESULT_DECLINED_EMV_HIGHER_CRYPTOGRAM_TYPE_SECOND_CAA	The ICC returned a higher cryptogram type than requested during second Card Action Analysis
TRX_RESULT_DECLINED_EMV_ARQC_SECOND_CAA	The ICC returned an ARQC during second card action analysis
TRX_RESULT_DECLINED_EMV_AAR_SECOND_CAA	The ICC returned a no longer valid AAR cryptogram during second Card Action Analysis
TRX_RESULT_ABORTED_CARD_REMOVED	The transaction is aborted by prematurely removing the card.
TRX_RESULT_DECLINED_EP2_FIRST_TAA	Ep2 magstripe transaction has been declined offline by First Terminal Action Analysis.
TRX_RESULT_DECLINED_EP2_SECOND_TAA	Ep2 magstripe transaction has been declined offline by Second Terminal Action Analysis.
TRX_RESULT_DECLINED_EP2_ONLINE_GENERIC	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_CARD_ERROR	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_CARD_EXPIRED	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_CARD_UNKNOWN	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_DUPLICATE_TRX	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_FUNDS_TOO_LOW	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_AMT_INVALID	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_TRX_INVALID	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_PIN_INCORRECT	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_SYSTEM	see EP2 specification

Constant	Description
_ERROR	
TRX_RESULT_DECLINED_EP2_ONLINE_TRX_REPETITION	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_TRY_LATER	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_WITHDRAWAL_AMT_EXCEEDED	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_WITHDRAWAL_FREQ_EXCEEDED	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_PIN_TRIES_EXCEEDED	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_CARD_BLOCKED	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_PIN_MANDATORY	see EP2 specification
TRX_RESULT_DECLINED_EP2_CASHBACK_AMOUNT_EXCEEDED	see EP2 specification
TRX_RESULT_DECLINED_EP2_CASHBACK_NOT_ALLOWED	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_REFERRAL	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_REFERRAL_WRONG_AUTH_CODE	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_REFERRAL_WRONG_AMT_AUTH	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_REFERRAL_OTHER	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_CAPTURE_CARD	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_CAPTURE_CARD_INFO_TO_CLIENT	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_CAPTURE_CARD_ORDER_TO_CLIENT	see EP2 specification
TRX_RESULT_DECLINED_EP2_ONLINE_CAPTURE_CARD_TIMEOUT_REMOVING_CARD	see EP2 specification
TRX_RESULT_ABORTED_STOP_BY_CARDHOLDER	Cardholder aborted transaction by pressing the 'Stop' key.
TRX_RESULT_ABORTED_STOP_BY_ATTENDANT	Cardholder aborted transaction by pressing the 'Stop' key.
TRX_RESULT_ABORTED_ECR_CONFIRMATION_FAILED	An error occurred during confirmation of an approved transaction by the ECR
TRX_RESULT_ABORTED_CARDHOLDER_NOT_READY	Cardholder did not react in a timely manner.
TRX_RESULT_ABORTED_EXTERNAL	The transaction was aborted by the attendant resp. cash register.
TRX_RESULT_ABORTED_BY_CLIENT	Cardholder aborted transaction, but not during PIN entry or application selection.
TRX_RESULT_ABORTED_FALLBACK_TO_MAGSTRIPE_FAILED	The magstripe could not be read after failed chip processing.
TRX_RESULT_ABORTED_CARD_UNKNOWN	The card is unknown to all magstripe & proprietary chip applications on the terminal and could not be processed by EMV.
TRX_RESULT_ABORTED_NOT_CONFIRMED_BY_ECR	An error or timeout occurred while waiting for the ECR to confirm an approved transaction.
TRX_RESULT_ABORTED_REVERSAL_NOT_POSSIBLE	Reversible original transaction could not be referenced.
TRX_RESULT_CARD_NOT_READABLE	Card did not contain valid chip or magstripe interface. Note: This code is usually not returned to the cash register but indicated to the cardholder with 'read error' without terminating the transaction.
TRX_RESULT_INVALID_TRANSACTION_REQUEST	The transaction request cannot be processed due to invalid data.
TRX_RESULT_FUNCTION_NOT_SUPPORTED	The requested function is not supported by the terminals resp. the application configuration.

Constant	Description
TRX_RESULT_ABORTED_UNEXPECTED_ERROR	An error occurred that was not foreseen by the software.
TRX_RESULT_ABORTED_EMV_APP_CONFIRMATION_NOT_SUPPORTED	The card's application(s) require(s) confirmation, but the terminal user interface does not allow application confirmation
TRX_RESULT_ABORTED_EMV_APP_CONFIRMATION	Cardholder aborted EMV applicatoin selection
TRX_RESULT_ABORTED_EMV_PIN_ENTRY	Cardholder has aborted EMV PIN entry by pressing the [STOP] key
TRX_RESULT_ABORTED_EMV_AMOUNT_CONFIRMATION	The card's application(s) require(s) confirmation, but the terminal user interface does not allow application confirmation
TRX_RESULT_ABORTED_EMV_CARDHOLDER_NOT_READY	The cardholder did not perform an appropriate action within the timeout period during application confirmation or PIN entry
TRX_RESULT_ABORTED_EMV_INVALID_TRX_REQ	The transaction request message sent to the EMV Application Kernel was malformed
TRX_RESULT_ABORTED_EMV_INVALID_APP_CONFIG_DATA	The application config data sent to the EMV Application Kernel was malformed
TRX_RESULT_ABORTED_EMV_CARD_BLOCKED	The card turns out to be blocked during Application Selection
TRX_RESULT_ABORTED_EMV_APP_BLOCKED	The application turns out to be blocked during Application Selection
TRX_RESULT_ABORTED_EMV_APP_NOT_SUPPORTED	No application was found that is supported by both the card and the terminal
TRX_RESULT_ABORTED_EMV_CONDITIONS_OF_USE_NOT_SATISFIED	The card answered the GET PROCESSING OPTIONS command with SW1 SW2
TRX_RESULT_ABORTED_EMV_SERVICE_NOT_ALLOWED_FIRST_CAA	The Cryptogram Information Data indicate 'Service not allowed' during first Card Action Analysis
TRX_RESULT_ABORTED_EMV_SERVICE_NOT_ALLOWED_SECOND_CAA	The Cryptogram Information Data indicate 'Service not allowed' during second Card Action Analysis
TRX_RESULT_ABORTED_EMV_IO_ERROR	The ICC reader returned an error during APDU transmission
TRX_RESULT_ABORTED_EMV_INVALID_SW1SW2_FINAL_SELECT	The ICC returned an error status during Final Selection
TRX_RESULT_ABORTED_EMV_INVALID_SW1SW2_GPO	The ICC returned an error status during Get Processing Options
TRX_RESULT_ABORTED_EMV_INVALID_SW1SW2_READ_APP_DATA	The ICC returned an error status during Reader Application Data
TRX_RESULT_ABORTED_EMV_INVALID_SW1SW2_CAA	The ICC returned an error status during Card Action Analysis
TRX_RESULT_ABORTED_EMV_INVALID_SW1SW2_INT_AUTH	The ICC returned an error status while performing DDA
TRX_RESULT_ABORTED_EMV_INVALID_SW1SW2_VERIFY	The ICC returned an error status during offline PIN verification
TRX_RESULT_ABORTED_EMV_MANDATORY_DATA_MISSING	After Read Application Data, missing mandatory elements were detected
TRX_RESULT_ABORTED_EMV_REDUNDANT_PRIMITIVE_OBJECT	Read Application Data yielded redundant primitive data ojects
TRX_RESULT_ABORTED_EMV_MALFORMED_ICC_DATA	The ICC returned malformed data in a response APDU
TRX_RESULT_ABORTED_EMV_HIGHER_CRYPTOGRAM_TYPE	The ICC returned a higher cryptogram type than requested by the terminal
TRX_RESULT_ABORTED_EMV_SECURITY_ERROR	An error was returned by the security module while performing EMV processing
TRX_RESULT_ABORTED_EMV_AAR_FIRST_CAA	The ICC returned a no longer valid AAR cryptogram during first Card Action Analysis
TRX_RESULT_ABORTED_EMV_MANDATORY_DATA_MISSING_SECOND_CAA	Missing mandatory elements detected during second Card Action Analysis
TRX_RESULT_ABORTED_EMV_MALFORMED_ICC_DATA_SECOND_CAA	The ICC returned malformed data during second Card Action Analysis
TRX_RESULT_ABORTED_EMV_INVALID_SW1SW2_SECOND_CAA	The ICC returned an error status during second Card Action Analysis
TRX_RESULT_ABORTED_EMV_UNEXPECTED_ERROR	An unexpected error occurred in the EMV Application Kernel
TRX_RESULT_ABORTED_EMV_CONTACTLESS_NOT_ALLOWED	Contactless transaction not allowed.

Constant	Description
TRX_RESULT_ABORTED_EMV_CONTACTLESS_ERROR	An error occurred during contactless card reading.
TRX_RESULT_ABORTED_EMV_CONTACTLESS_CURRENCY_NOT_SUPPORTED	Currency not supported for contactless processing.
TRX_RESULT_ABORTED_EMV_CONTACTLESS_FATAL_ERROR	Fatal error condition during contactless processing.
TRX_RESULT_ABORTED_EMV_CONTACTLESS_CARD_INSERTION	Contactless transaction was aborted due to the insertion of a contact card.
TRX_RESULT_ABORTED_EMV_CONTACTLESS_AAC_FIRST_CAA	Contactless transaction was aborted during First Card Action Analysis.
TRX_RESULT_ABORTED_EP2_LUHN_CHECK_FAILED	The LUHN check of the PAN failed.
TRX_RESULT_ABORTED_EP2_MALFORMED_PAN	The PAN does not meet basic format requirements.
TRX_RESULT_ABORTED_EP2_MANUAL_PAN_ENTRY_NOT_ALLOWED	The ep2 PAN Key Indicator forbids manual card data entry for this AID.
TRX_RESULT_ABORTED_EP2_INTERNATIONAL_TRX_NOT_ALLOWED	The Country Restriction Flags forbids international transactions for this AID.
TRX_RESULT_ABORTED_EP2_DOMESTIC_TRX_NOT_ALLOWED	The Country Restriction Flags forbids domestic transactions for this AID.
TRX_RESULT_ABORTED_EP2_CURRENCY_NOT_SUPPORTED	The requested transaction currency is not supported for this AID.
TRX_RESULT_ABORTED_EP2_FUNCTION_NOT_SUPPORTED	The requested transaction function is not supported for this AID.
TRX_RESULT_ABORTED_EP2_MAXIMUM_AMOUNT_EXCEEDED	The requested Amount, Authorised exceeds the configured maximum amount for this AID.
TRX_RESULT_ABORTED_EP2_BELOW_MINIMUM_AMOUNT	The requested Amount, Authorised was below the configured minimum amount for this AID.
TRX_RESULT_ABORTED_EP2_DOUBLE_TRANSACTION	A double transaction has been detected and refused by the attendant.
TRX_RESULT_ABORTED_EP2_MAXIMUM_CASHBACK_AMOUNT_EXCEEDED	The requested Amount, Other exceeds the configured Maximum Cashback Amount for this AID.
TRX_RESULT_ABORTED_EP2_BELOW_MINIMUM_CASHBACK_AMOUNT	The requested Amount, Other was below the configured Minimum Cashback Amount for this AID.
TRX_RESULT_ABORTED_EP2_BELOW_MINIMUM_CASHBACK_AMOUNT	The requested Amount, Other was below the configured Minimum Cashback Amount for this AID.
TRX_RESULT_ABORTED_MATCHED_SURROGATE_PAN	If the SurrogatePAN of the card presented matches the list of SurrogatePANs of the transaction, this event is triggered to indicate that the PAN matches with the list

3.9.4.3 Transaction.POSEntryMode

This value indicates whether the card data of a card in the terminal is to be read, or is to be given in attributes of the transaction object.

Constant	Value	Description
POS_ENTRY_MANUAL	1	Card data for the transaction are taken from the transaction object attributes AppExpDate, AppPAN and CVC2
POS_ENTRY_MAGSTRIPE	90	Card data for the transaction are taken from the transaction object attributes Track2 and Track3 respectively (The cash register has a magnetic stripe reader) Currently not supported by Transaction object.
POS_ENTRY_UNSPECIFIED	0	Card data for the transaction is obtained from the terminal itself.
POS_ENTRY_BAR_CODE	3	Bar code
POS_ENTRY_OCR	4	Optical character recognition
POS_ENTRY_ICC	5	Chip card
POS_ENTRY_MAGSTRIPE_AFTER_FIRST_UNSUCCESSFUL_IC_READ	91	Fallback to magnetic stripe after IC card reading failed; last transaction in the log has another POS Entry Mode.
POS_ENTRY_MAGSTRIPE_AFTER_UNSUCCESSFUL_IC_READ	92	Fallback to magnetic stripe after IC card reading failed repeatedly.

Constant	Value	Description
POS_ENTRY_MAGSTRIPE_AFTER_IC C_PROCESSING_FAILED	93	Fallback to magnetic stripe after IC card processing failed.
POS_ENTRY_RFID_EMV	97	Contactless, chip profile.
POS_ENTRY_RFID_MAGSTRIPE	98	Contactless, magnetic stripe profile.

3.9.4.4 Transaction.Status

This value indicates the status of the current transaction. When a transaction request has occurred, the status value of the return value is conveyed in the field *AuthResult*.
The following status values are possible:

Transaction.Status	Description
ABORTED	Termination of transaction by user or terminal
APPLICATION_ ABORTED	Transaction terminated by ECR driver
APPROVED	Transaction processing successfully confirmed
DECLINED	Transaction processing declined
NEW	New transaction still to be processed
PENDING	Transaction currently being processed
REFERRED	Transaction rejected with status: "telephone authorization required". Cash register application must start transaction type, createPurchasePhoneAuthorized`

3.9.4.5 Transaction.Function

The value *Transaction.Function* indicates the type of transaction to be executed. The following list shows all possible values.

Transaction.Function	Description
ACCOUNTINQUIRY	Balance inquiry
AUTHORIZATIONPURCHASE	Authorization
CASHADVANCE	Cash withdrawal (only if Postfunctions available)
CLIENTIDREQUEST	Client identification (only if Postfunctions available)
COMBINEDTRANSACTION	Combined transaction (only if Postfunctions available)
CONFIRMPHONEAUTHORIZED RESERVATION	Confirmation telephone reservation (only if Postfunctions available)
CREDIT	Credit
DEPOSIT	Deposit (only if Postfunctions available)
GIRO	Cash payment paying-in slip (only if Postfunctions available)
PURCHASE	Booking
PURCHASEFORCEDACCEPTANCE	Forced booking
PURCHASEMAILORDERED	Distance purchase via internet/mail
PURCHASEPHONEAUTHORIZED	Telephone authorization
PURCHASEPHONEORDERED	Distance purchase via telephone
PURCHASERESERVATION	Booking reservation
RESERVATION	Reservation
RESERVATIONADJUSTMENT	Reservation increase
REVERSAL	Reversal
TIP	Tip (this function is obsolete as of EP2 5.x)

Transaction.Function	Description
AUTHORIZATIONDEPOSIT	Authorization of deposit (only if Postfunctions available)
AUTHORIZATIONCREDIT	Authorization of credit (only if Postfunctions available)

3.9.4.6 Transaction.TransactionRequestFlags

Constant	Description
SILENT	Starts a transaction without updating the GUI of the terminal

3.9.5 Class Transaction.CVMResults

This class contains information needed to hold the results of cardholder verification method.

3.9.5.1 Transaction.CVMResults properties

Property	Type	Description
getCodes()	Transaction.CVMResults.Codes	This property provides information contained in a code for referencing a cardholder verification method chosen for the current transaction. It is a list of one or more codes of the enumeration.
getCondition()	Transaction.CVMResults.Conditions	This contains the condition applied to the cardholder verification method.
getResult()	Transaction.CVMResults.Result	This describes the result of the actually chosen cardholder verification method performed during the transaction.
isPINVerified()	boolean	Returns true if the transaction was verified by input of the cardholder's PIN. Otherwiese false. For further details see 5.7.1
isSignatureVerified()	boolean	Returns true if the transaction was verified by cardholder's signature. Otherwise false. For further details see 5.7.1

3.9.5.2 Transaction.CVMResults enumerations

For the detailed description of the codes see the EMV Books.

3.9.5.2.1 Transaction.CVMResults.Code

FAIL_CARDHOLDER_VERIFICATION_IF_THIS_CVM_IS_UNSUCCESSFUL
APPLY_SUCCEEDING_CV_RULE_IF_THIS_CVM_IS_UNSUCCESSFUL
FAIL_CVM_PROCESSING
PLAINTEXT_PIN_VERIFICATION_PERFORMED_BY_ICC
ENCIPHERED_PIN_VERIFIED_ONLINE
PLAINTEXT_PIN_VERIFICATION_PERFORMED_BY_ICC_AND_SIGNATURE
ENCIPHERED_PIN_VERIFICATION_PERFORMED_BY_ICC
ENCIPHERED_PIN_VERIFICATION_PERFORMED_BY_ICC_AND_SIGNATURE
RESERVED_FOR_FUTURE_USE_BY_THE_EMV_SPECIFICATION
SIGNATURE
NO_CVM_REQUIRED
RESERVED_FOR_USE_BY_THE_INDIVIDUAL_PAYMENT_SYSTEMS
RESERVED_FOR_USE_BY_THE_ISSUER
NO_CVM_PERFORMED
UNKNOWN

3.9.5.2.2 Transaction.CVMResults.Condition

ALWAYS
IF_UNATTENDED_CASH
IF_NO_CASH
IF_SUPPORTED
IF_MANUAL_CASH
IF_CASHBACK
IF_UNDER_X

IF_OVER_X
IF_UNDER_Y
IF_OVER_Y

3.9.5.2.3 Transaction.CVMResults.Result

UNKNOWN
FAILED
SUCCESS

3.10 Class TransactionDatachange

This class contains the information about datachange reason of a transaction, when it is triggered by terminal. There are various types of reasons.

3.10.1 TransactionDatachange methods

Property	Description
response(Transaction**)**	This method returns changed transaction data back to terminal after the transaction data change was initiated by onTransactionDatachangeRequest callback.

3.10.2 TransactionDatachange properties

Property	Type	Description
getReason()	TransactionDatachange.Reason	This property provides information which reason caused creation of this class. The reason comes from the terminal and cannot be changed.
getReasonData()	Byte[]	This contains reason specific data bytes. The description can be found in enumeration.
getTransaction()	Transaction	Transaction object of the transaction, which was interrupted.

3.10.3 TransactionDatachange enumerations

3.10.3.1 TransactionDatachange.Reason

Enumeration	Description
NONE	No transaction data change event will be triggered during transactions
MYONE_CARD_DETECTED	Terminal triggers OnTransactionDatachangeRequest callback, when a myOne card with rabat code was presented
BONUS_CARD_DETECTED	Terminal triggers OnTransactionDatachangeRequest callback, when a Bonus card was presented

Description of the data bytes for the particular data change reasons, which come back from the terminal in case the data change has been triggered.

Enumeration	Byte offset	Description
MYONE_CARD_DETECTED	0	Rabat code on the card (e.g. 19)
BONUS_CARD_DETECTED	0	Bonus card brand number (see 3.10.3.1.1)

Bonus card brand numbers:

Name	Code
Jelmoli Paycard	1
Jelmoli Visa Bonus Card	2
Visa Bonus Card	3
Jelmoli Geschenkkarte	4

3.11 Class LoyaltyPromotion

This class contains information about currently running loyalty transaction.

3.11.1 LoyaltyPromotion constructor

Name	Description
No public constructor	

3.11.2 LoyaltyPromotion properties

The LoyaltyPromotion object contains following properties:

Property	Type	Description
getLoyaltyAcquirerID	long	Uniquelly identifies the acquirere of the loyalty core
getAmountAuthorised	long	Original amount minus the amounts corresponding to active campaigns (excluding any earned discount vouchers)
getLoyaltyTrmId	string	Terminal ID as used for loylalty operations
getAID	string	Identification of application as per EMV as hexadecimal digits
getCurrency	Transaction. Currency	Transaction currency as per ISO-4217 (for values see 3.9.4.1)
getTrxSeqCnt	int	Transaction sequence counter
getLoyaltyResult	LoyaltyResult	Result of the communication with loyalty core
getCampaigns	Campaign[]	Array of campaigns used in the transaction
getPointsAvailable	long	Loyalty points still available to the cardholder
getPointsDebited	long	Loyalty points debited during this transaction
getPointsEarned	long	Loyalty points earned during this transaction
getPointsRedeemed	long	Loyalty points redeemed during this transaction
getPointsPrevious	long	Points available to the cardholder before this transaction
getRedemptionItems	RedemptionItem[]	Array of redemption items available for the customer to choose from (in case of 2 step process)
getRefNum	string	Reference number, if null we are in a two-step process of loyalty processing, a LoyaltyAdviceNotification should be expected

In the one-step process the values are valid they are the only information achieved during the transaction processing. After this only the final transaction result follows.

In the two-step process of the loyalty processing (getRefNum is not set here) only getCampaigns has a value, which should be considered defnitelly as the rest of values can change in the LoyaltyAdvice object.

However transaction object will contain the resulting information about loyalty processing in both cases.

3.11.3 LoyaltyPromotion enumerations

3.11.3.1 LoyaltyResult

Following loyalty results can be obtained from the loyalty core:

Code	Description
0	Transaction approved
13	Account does not exist

97	Customer does not exist
200	Service not available
212	Purchase amount must be specified
700	Customer history record does not exist
701	Merchant history record does not exist
50005	Transaction definition not found
50030	Product type not defined in an active scheme
60001	Invalid card
60002	Invalid Merchant ID
60003	Invalid Terminal ID
60004	Invalid Acquirer ID
60006	Invalid Store
60007	Invalid Message Type
60009	Invalid institution
60102	Card expired
60104	Duplicate transaction
60105	Insufficient points for redemption
90920	Invalid customer number
90921	Invalid institution password
90933	Invalid language code

3.12 Class LoyaltyAdvice

This class is currently identical with LoyaltyPromotion. The only difference is the origin. This class comes from the second notification in case of the two-step process. The content of the object should correspond with actual result situation of the loyalty processing after the cardholder has made all choices and loyalty core server responded accrodingly.

3.13 Class Campaign

This class contains information about a campaign, which is used for transactions containing Aduno New Loyalty data.

3.13.1 Campaign properties

The cmpaign object contains following properties:

Property	Type	Description
getType **setType**	CampaignType	Type of the campaign
getID **setID**	string(16)	Id of the campaign as used in communication with loyalty host
getRewardType **setRewardType**	RewardType	Type of a reward campaign; RewardType::UNDEFINED if not a reward campaign
setSubTitle **getSubTitle**	string	Campaign Subtitle
getSummaryText **setSummaryText**	string	Campaign summary text
getPrice **setPrice**	long	The price of a product in a redemption campaign
getPoints **setPoints**	long	Amount of points which activatea the percentual discount
getValue **setValue**	long	Value of a reward or voucher campaign, meaning depends on reward type

getCurrency **setCurrency**	Transaction.Currency	Only present if RewardType=DISCOUNT_AMOUNT, ISO4217 code
getTitle **setTitle**	string	Campaign title in cardholder language
getText **setText**	string	Campaign text in cardholder language
getAppliedFlag **setAppliedFlag**	string	Applied Flag, according to New Loyalty Data Dictionary

3.13.2 Campaign enumerations

3.13.2.1 CampaignType

Following types of campaigns are available:

Code	Value	Description
-1	UNDEFINED	Campaign currently unknown to KIT
1	REWARD_CAMPAIGN	The campaign is something to be rewarded
2	VOUCHER	The campaign is a voucher
3	COMMUNICATION_CAMPAIGN	The campaign is only for communication to the cardholder

3.13.2.2 RewardType

Following types of rewards are available:

Code	Value	Description
1	DiscountPercent	The discount as percentage of the amount
2	DiscountAmount	Discount as a absolute amount of money
3	PointsMultiplierOnStandard	The standard goods points will be multiplied
4	PointsAdditional	Additional points are added to the standard points
5	PointsFixed	Fixed points
6	FreeItems	Items are for free

3.14 Class RedemptionItem

This class contains information about the redemption item. This is used for redemption item sent from the loyalty core.

3.14.1 RedemptionItem properties

The cmpaign object contains following properties:

Property	Type	Description
getCode **setCode**	string	product code usually an EAN
getDescription **setDescription**	string	any suitable product description
getAmount	long	The price of the product after loyalty processing (how much is the actual price of the item after spending loyalty points on it).
getPoints	string	Price of the item in loyalty points

3.15 Class LoyaltyProductRecord

This class contains information about the product as used basket items which are usually known by the ECR and will be sent to inform the terminal what is actually sold, so it can make the matching and adjust the price accordingly to information from the loyalty core and/or make other adjustments inside the loyalty processing.

3.15.1 LoyaltyProductRecord constructor

Name	Description
RedemptionItem(**code,description,price)**	

3.15.2 LoyaltyProductRecord properties

The cmpaign object contains following properties:

Property	Type	Description
getCode **setCode**	string	product code usually an EAN
getDescription **setDescription**	string	any suitable product description
getPrice **setPrice**	long	The price of the product as registered in the ECR

3.16 Class *Screen*

The screen object enables the definition of a screen to be displayed as a default when no explicit dialog is shown with showDialog(). The object can only be used in exclusive dialog mode (*Device.startUIExcl()*)

3.16.1 Screen constructor

In order to initialize a new screen class, the following constructor is available.

Name	Description
Screen()	Initializes a new instance of the screen class without values
Screen(Screen.Icon icon, Screen.Brand brand)	Initializes a new instance of the screen class with the icon to be displayed and the brand
Screen(Screen.Icon icon, Screen.Brand brand, Screen.ResourceText text, String[] textArguments	Initializes a new instance of the screen class with the icon to be displayed, the brand, the text and arguments to the text.
Screen(Screen.Icon icon, Screen.ResourceText text, String[] textArguments)	Initializes a new instance of the screen class with the icon and text to be displayed, as well as the arguments to the text

3.16.2 Screen properties

The screen object contains following properties:

Property	Type	Description
getBrand, setBrand	Brand	Card logo to be displayed
getIcon, setIcon	Icon	Icon or animation to be displayed
getText, setText	Screen. ResourceText	ResourceId of text to be displayed
getTextArguments, setTextArguments	String[]	Array with arguments of text to be displayed

3.16.3 Screen enumerations

3.16.3.1 Icon

The following icons and animations are available for display:

Value	Description
Approved	Animation processing OK
Declined	Animation processing rejected
InsertCard	Animation for card insertion
Mobile	Image for telephone
NoIcon	No image or animation is displayed
PleaseWait	Animation: please wait
Question	Animated question mark
RemoveCard	Animation for card removal

3.16.3.2 Brand

Value	Description
Coopmobile	Coop mobile logo
Orange	Orange logo
SwissPost	Post logo in terminal language

PostFinance	PostFinance logo
Sunrise	Sunrise logo
Swisscom	Swisscom mobile logo
Tele2	Tele2 logo
Yallo	YalloLogo

3.16.3.3 ResourceText

The following table shows all possible texts which can be viewed on the terminal display. The arguments of a ResourceText are assigned as a string array (string[]) within the function *setTextArguments()*.

Value	German text	Argument
No_recharge_card	Keine Recharge Karte	-
Phone_number_currency_ amount	<tel nummer> <currency> <amount>	1. Numerical string 2. 3-letter currency code as per ISO-4217 3. Amount as decimal value
Phone_number_or_ pre_paid_card	Bitte Tel.Nummer oder PrePaid-Karte	-
Phone_number_too_long	Tel. Nummer zu lang	-
Phone_number_too_short	Tel. Nummer zu kurz	-
Please_enter_your_ phone_number	Bitte Telefonnummer eingeben	-
Please_wait	Bitte warten	-
Processing_failed	Verarbeitung fehlgeschlagen	-
Processing_OK	Verarbeitung OK	-
Register_card_question	Karte anmelden	-
Signature_please	Unterschrift bitte	-
Welcome_insert_card	Willkommen Karte bitte	-
Your_phone_number_is	Ihre Telefonnummer lautet <tel nummer>	1. Numerical string
Account_transfer	Von <LastKonto> <LastKontoTyp> <Curr> <Lastkonto-Bezeichnung auf <GutsKonto> <GutsKontoTyp> <Curr> <GutsKonto-Bezeichnung <Currency> <Amount>	1. Debit account number 2. Debit account type 3. Debit account currency code as per ISO-4217 4. Debit account denotation 5. Credit account number 6. Credit account type 7. Credit account currency code as per ISO-4217 8. Credit account denotation 9. Currency code as per ISO-4217 10. Amount as decimal value
Confirm_account_transfer	<LastKonto> <LastKontoTyp> <Currency> <Currency> <Amount Available> <GutsKonto> <GutsKontoTyp> <Currency> <Currency> <Amount Available>	1. Debit account number 2. Debit account type 3. Debit account currency code as per ISO-4217 4. Debit account available balance 5. Credit account number 6. Credit account type 7. Credit account currency code as per ISO-4217 8. Credit account available balance
E_CHECK	Bezug ab Konto <Debit Account> <Debit Currency> <Debug Amount> OK?	1. Debit account (string) 2. Debit account currency code as per ISO-4217 3. Debit amount

Value	German text	Argument
Signature_Legal	Ich habe die Sendungen und die darauf aufgeführten Inhalte erhalten:	
Pump_Selection	Bitte Säule wählen und bestätigen:	1. Text to display (optional)
Free_Text	(no text)	1. Text to display
Free_Text_with_Brands	(no text)	1. Text to display
Mileage_Input	Kilometerstand eingeben + OK	1. Text to display (optional)
AdditionalInfo_Input	Zusatzinfo eingeben + OK	1. Text to display (optional)
Choose function	Please choose function + OK	2. Text to display (optional)
Free_Text_with_Buttons	(no text)	1. Text display 2. Custom Text for Left Button (F1) 3. Custom Text for Right Button (F4)

3.17 Class *Dialog*

The dialog object acts for the purpose of defining an individual screen. With the help of the method Device.ShowDialog() this is shown on the screen of the terminal.

3.17.1 Dialog constructor

In order to initialize a new dialog class, the following constructor is available.

Name	Description
Dialog()	Creates a new instance of dialog class without values
Dialog(Dialog.InputType inputType, int timeout, Screen.ResourceText text)	Initializes a new instance of dialog class requiring the mandatory fields InputType, Timeout and TextRessource.
Dialog(Dialog.InputType inputType, Screen.Icon icon, Screen.Brand brand, Screen.ResourceText text, String[] textArguments, int timeout, Dialog.Buttons buttons	Initializes a new instance of dialog class requiring InputType, the icon to be displayed, brand, text resource, timeout and buttons.
Dialog(Dialog.InputType inputType, Screen.Icon icon, Screen.Brand brand, Screen.ResourceText text, String[] textArguments, int timeout, Dialog.Buttons buttons, int minLength, int maxLength, string referencePassword, string[] watermark)	Initializes a new instance of dialog class requiring InputType, the icon to be displayed, brand, text resource, arguments for text resource, Timeout, buttons, and default input value, minimum and maximum input length, reference password and watermark.

3.17.2 Dialog properties

The dialog object contains the following properties:

Property	Type	Description
getButtons, setButtons	Button	Buttons to be displayed on the screen
getBrand, setBrand	Screen.Brand	Credit card logo to be displayed on the screen
getIcon, setIcon	Screen.Icon	Icon or animation to be displayed
getInputType, setInputType	Dialog InputType	Defines which type of input is to be expected in the user interface (e.g. electronic signature)
getMaxLength, setMaxLength	Integer	Maximum valid length of entered value. Must be given with DialogInputType Number, NumberOrTrack2, NumericString or Password. When value set at -1, there is no limitation from KIT, but on the device (internally). When value set at 0, no input is possible.
getMinLength, setMinLength	Integer	Minimum valid length of entered value. Must be given with DialogInputType Number, NumberOrTrack2, NumericString or Password. When value set at -1 or 0, no limitation from KIT.
getReferencePassword, setReferencePassword	String	Password to be checked with InputType= Password
getTimeout, setTimeout	long	Shows how long the dialog is to be displayed. If value set at -1 or Dialog.infiniteTimeout, dialog waits indefinitely.
getText, setText	Screen. ResourceText	ResourceId of text to be displayed
getTextArguments, setTextArguments	String[]	Array with the arguments of the text to be displayed
getWatermark, setWatermark	String[]	Enables the input of a string array to display as watermark behind the picture (e.g. consignment number). The device does not check if the watermark is too long or wrongly formatted. Formatting and size have to be considered when setting the watermark via setWatermark()

getDialogFlags **setDialogFlags**	Dialog.DialogFlags	Set dialog options, see DialogFlags enum.

3.17.3 Dialog constants

The dialog object contains the following constants.

Constant	Type	Description
infiniteTimeout	long	Infinite timeout (value of -1)

3.17.4 Dialog enumerations

3.17.4.1 Dialog.InputType

For the function showScreen() the following values for type of input on the display can be given:

Value	Description
Empty	No input expected
ESign	Panel for input of electronic signature is displayed
Number	Enables input of a number via display of number pad. The input of leading 0 is not possible
NumericString OrTrack2	Either the card reader a input of a number via the number pad will be used for input. Leading 0 is not possible
NumericString	Enables the input of a number, the leading 0 will be not truncated
Password	Displays the number pad in order to allow the input of a password
Track2	The card reader will be used to read the track 2 of the card as an input

3.17.4.2 Dialog.Button

For the function showScreen() the following values for the displaying of the buttons can be given:

Value	Description
OK	Displays [OK] button
Cancel	Displays [CANCEL] button
OKCancel	Displays [OK] and [CANCEL] buttons
NONE	Displays no buttons
LeftCustom	Displays leftmost function button with custom text (F1)
RightCustom	Displays rightmost function button with custom text (F4)
LeftCustomOK	Displays [OK] and F1
LeftCustomCancel	Displays [Cancel] and F1
LeftCustomOKCancel	Displays [OK] and [Cancel] and F1
RightCustomOK	Displays [OK] and F4
RightCustomCancel	Displays [Cancel] and F4
RightCustomOKCancel	Displays [OK] and [Cancel] and F4
ALL	Displays [OK], [Cancel], F1 and F4

3.17.4.3 Dialog.DialogFlags

When displaying a Dialog on the terminal screen the following flags change the behavior of the displayed messages.

Value	Description
NONE	
LEAVE_ONSCREEN	The dialog message remains on screen, even after the user interaction is finished

WITH_DISPLAYTEXT	Display text along with user input
OVERRIDE_IDLE_TEXT	Replace default idle texts ‘Welcome’ & ‘Welcome, present card’
LEAVE_ONSCREEN_WITH_DISPLAYTEXT	LEAVE_ONSCREEN and WITH_DISPLAYTEXT
LEAVE_ONSCREEN_OVERRIDE_IDLE_TEXT	LEAVE_ONSCREEN and OVERRIDE_IDLE_TEXT
WITH_DISPLAYTEXT_OVERRIDE_IDLE_TEXT	WITH_DISPLAYTEXTa and OVERRIDE_IDLE_TEXT
ALL	LEAVE_ONSCREEN and WITH_DISPLAYTEXT and OVERRIDE_IDLE_TEXT

3.18 Class DialogResponse

When the user presses a button on the screen, inserts a card or the timeout has expired, the values entered by the user are returned in the object *DialogResponse*.

3.18.1 DialogResponse properties

The following values can be queried in the object *DialogResponse*.

Property	Type	Description
getESign	byte[]	Electronic signature returned as PNG image. The returned image is black & white and has a size of 320x105 pixels (horizontal format)
isInputMatch	bool	True when entered password is correct
getResult	Result	Executed action, e.g. OK button pressed
getResultNumber	long	Entered numerical value
getResultString	String	Entered alpha-numerical value
getTrack2	String	Track 2 data from scanned card

3.18.2 DialogResponse enumerations

3.18.2.1 Result

The result property of the DialogResponse objects can possess the following values:

Value	Description
OK	Button [OK] was pressed
Stop	Button [STOP] was pressed
Timeout	Timeout defined time has expired
Cancel	Processing was cancelled by an abortDialog() command
LeftCustom	Leftmost Function Button was pressed (F1)
RightCustom	Rightmost Function Buton was pressed (F4)

3.19 Class *Error*

Should an error occur in the KITDriver, the detailed error data is deposited in the error object.

3.19.1 Error properties

The error object contains the following properties:

Property	Type	Description
getErrorCode	Error.Code	Exception error code as error enumeration
getErrorMessage	string	Exception error text
getErrorSource	string	Program line containing error
getNativeMessage	int	Native error message directly from EFT device
getNativeStatus	string	Native error status directly from EFT device

3.19.2 ErrorCode enumeration

In the ErrorCode field of the error object, the following errors can be returned:

Enum	ErrorCode	Text
ConnAttemptWithClosedSocket	1000	Attempted connection - closed socket
SendAttemptWithClosedSocket	1001	Message sent - closed socket
SendFailed	1002	Message sent to device - failed
CloseShiftFailed	1004	CloseShift activated when terminal already deactivated
ConfigureFailed	1006	Configure attempted when device activated
InitializeFailed	1007	Initialization activated when device still active
DownloadFailed	1008	Download attempted when device activated
FrameCorrupted	1009	Received frame is corrupt
SocketTimeout	1010	Socket timeout occurred
SocketIOError	1011	Socket I/O error
DeviceErrorResponse	1012	Device sent general error. Check native status of error object
DeviceNotFound	1013	Device not found
DeviceNotResponding	1014	Device does not respond
ConfirmationResponse OnNoTransaction	1015	Confirmation response on a non-existent transaction
TransactionResponse OnNoTransaction	1016	Transactions response on a non-existent transaction
CommitOnAbortedTransaction	1017	Commit attempt on transaction with status ABORTED
CommitOnRollbackedTransaction	1018	Commit attempt on transaction already rolled back
CommitOnDeclinedTransaction	1019	Commit attempt on transaction with status DECLINED
CommitOnUnresponded Transaction	1020	Commit attempt on transaction with no reply
RollbackOnUnresponded Transaction	1021	Rollback attempt on transaction which has not yet received a reply
ConfirmationResponse WithoutConfirmationRequest	1022	Confirmation response received without confirmation request
InvalidValue	1023	Invalid value received from terminal
Application timeout	1024	An application timeout occurred as described in 6.1.1
Transaction timeout	1025	An timeout occurred during waiting for onDoTransaciton as described in 6.1.1

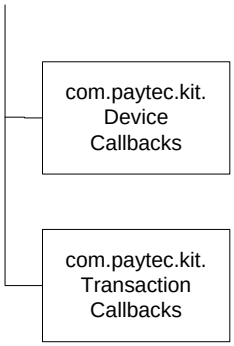
4 KIT Callbacks

4.1 Introduction

For the reception of events, callback classes are used in the KIT. These have to be given upon instantiation of the KIT driver classes. These callbacks are mostly triggered through asynchronous communication with the device. There are defined timeouts for each such an asynchronous response. The onError callback is triggered, when the timeout expires. The corresponding timeouts are in the callbacks description table if applicable.

4.2 Callbacks class hierarchy

The KIT contains the following callback classes whose purpose is to receive events.



4.3 Class DeviceCallbacks

The DeviceCallbacks class receives events based on requests from the device object.

4.3.1 DeviceCallbacks methods

The DeviceCallbacks class has the following methods:

Event	Description	Device method	Timeout
onAddToBasketResponse()	Triggered when the item was added to the basket successfully	addToBasket()	TOGenApp
onClearBasketResponse()	Triggered when the basket was cleared of all items	clearBasket()	TOGenApp
onCloseShiftResponse ()	Triggered if shift has been closed.	closeShift()	TOGenApp
onConfigureResponse ()	Triggered when the terminal finishes the configuration process successfully.	configure()	TOConfigRsp
onConnectResponse ()	Triggered on successful connection to terminal	connect()	TOGenApp
onDeviceStatusResponse(List<DeviceState> deviceStates**)**	Returns device status as soon as terminal status changes. Current statuses are contained in the argument.	getDeviceStatus() or spontaneously	TOGenApp
onDialogResponse (DialogResponse**)**	Triggered if action executed on defined dialog. Returns the *DialogResponse* object with all values.	onShowDialog()	TOTrxRsp
onDoTransactionResponse (Transaction**)**	Triggered if transaction processing successfully concluded. Returns the respective transaction object.	doTransaction(Transaction)	TOTrxRsp

Event	Description	Device method	Timeout
onInitTransaction(TransactionInit**)**	Triggered if transaction initialization reached a point where it needs data from host application or the initialization is ended, this method needn't be implemented if application does not make use of initTransaction otherwise it issues an exception.	initTransaction()	ToInitTrx ToValidCard, ToAppSel, ToWaitTrxReq, ToManAppSel
onEjectCardResponse ()	Triggered on removal of card. Device.State.READER_SLOT_OCCUPIED indicates status when card removed from reader	ejectCard()	TOGenApp
onEjectCardForcedResponse ()	Triggered when an acknowledge comes from the termina that the command was sent to the motor of the reader.	ejectCardForced()	TOGenApp
onError(Error error)	When an error occurs during call of an asynchronous method, the event onError() is called. The *Error* object contains all error details.	*any asynchronous method*	-
onFinalBalanceResponse ()	Triggered when final balance is finished	finalBalance()	TOGenApp
onInitializeResponse ()	Triggered on response on cards initialization request. Notice, that the result of the device initialization cannot be currently determined through KIT driver.	initialize()	TOInitRsp
onOpenShiftResponse ()	Triggered when shift opened.	openShift()	TOGenApp
onReceiptResponse(Receipt receipt**)**	Triggered when a complete receipt data were received terminal. If no receipt of the requested type was found, the type of the receipt in the receipt class is NONE and text is empty.	requestReceipt	TOGenApp
onReportResponse(Report report**)**	Triggered when a complete report data were receipt from terminal.	requestReport	TOGenApp
onStartUIExclusiveResponse()	Triggered when terminal goes into customer UI mode. All EFT operations are locked and cannot be executed.	startUIExcl()	TOGenApp
onStopUIExclusiveResponse()	Triggered when terminal returns back from customer UI mode. EFT operations are unlocked and can be executed again.	stopUIExcl()	TOGenApp
onChangeLanguageSelection(bool selectionEnabled**)**	Triggered when language selection flag in terminal has changed. This is normally caused by Device.enableLanguageSelection or Device.disableLanguageSelection method. The parameter selectionEnabled indicates the current status in terminal.	disableLanguageSelection() enableLanguageSelection()	TOGenApp
onTransactionDatachangeRequest(TransactionDatachange[]**)**	Triggered when a reason occurs on the terminal which requests the ecr to change some transaction data. The data can be then changed from the ECR and transaction proceeds further. The triggering of a particular reason must be set with the methods of the Device object.	*spontaneously when needed by the terminal*	
onLoyaltyPromotionNotification(LoyaltyPromotion**)**	Triggered when terminal is processing transaction with loyalty information. LoyaltyPromotion object contains information about currently processed transaction.	doTransaction()	TOTrxRsp
onLoyaltAdviceNotification(LoyaltyAdvice**)**	Triggered when terminal is processing two-step transaction with loyalty information. LoyaltyPromotion object contains information about currently processed transaction.	doTransaction()	TOTrxRsp

4.4 Class TransactionCallbacks

The TransactionCallbacks class receives events based on asynchronous method requests from the transaction object.

4.4.1 TransactionCallbacks methods

The TransactionCallbacks class has the following methods:

Event	Description	Transaction method	Timeout
onCommitResponse (Transaction trx**)**	Is activated if transaction commit was confirmed. Also returns transaction object.	Commit()	TOGenApp
onRollbackResponse (Transaction trx**)**	Confirms transaction rollback (cancellation) and returns transaction object.	Rollback()	TOGenApp

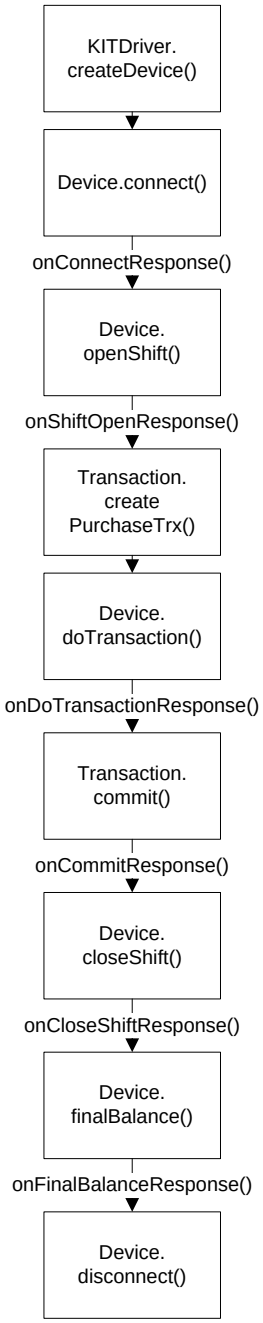
5 Implementation

5.1 Introduction

This chapter describes the implementation of the KIT driver in an application. The most important functionality concerning communication with the terminal will be described here. All examples are available in sample projects.

5.2 Cycle

The following chart demonstrates the program cycle during the execution of a simple transaction.



5.3 Instantiation of KIT driver

In order for the KIT driver to start, an instance of the KITDriver object needs to be created. This object needs to be given the callback device for device and transaction events as well as the name of the configuration data. As a rule, only one thread is permitted for the execution of the KIT driver. Thus for example the function Device.abortTransaction() must be called from the same thread as the function Device.doTransaction(). An example of callback functions is detailed in chapter 6.

5.4 Terminal connection

In order to connect the KIT to a terminal, an instance of the device needs to be created with the function KITDriver.createDevice(). This function retrieves the connecting URL of the device from the configuration data (first entry for parameter DeviceID).

Via the command Device.connect() a connection to the terminal can be created. When the terminal has been successfully connected, the field Device.isConnected has the value TRUE.

5.5 Terminal activation

In order that a transaction can be executed on the terminal, a shift has to be opened. This is achieved via the command Device.openShift().

5.6 Special terminal functions

There are additional ep2 and nonep2 functions, which can be controlled through KIT. Normally their execution is handled by terminal itself without the need to call them explicitly from KIT. However some situations can arise, where it is necessary to call them explicitly.

5.6.1 EP2 functions

These functions are described in detail in the ep2 specification.

5.6.1.1 Configure

The configuration is normally scheduled in the terminal by service center and is being called regularly according to timeplan defined in service center. KIT cann trigger this function explicitly if there is a need to get the newly configured data in the service center sooner than this regular call from terminal.

Configure can change some attributes of the terminal defined in service center. One of them is terminal SW download. If there is a flag in the serice center, which says that the terminal should update its SW, the execution of a SW download routine as described in 5.6.2.1 will be scheduled internally in the terminal (usually to 5 seconds after configuration ended).

5.6.1.2 Initialize

The initialization sends a request for initi data to the acquirers, which are brought to terminal during configuration process. These init data contains information about the brands and further detail connected with the payment process through parti. The initialization takes normally place after configuration. It happens basically then, when the configuration changed some information about acquirers.

5.6.2 Non EP2 functions

5.6.2.1 swUpdate

The swUpdate ends the main payment application and starts the service mode, which connects to service center and looks for the availability of the new Terminal SW. If there is a SW download ready for the terminal. Terminal starts an SW update process. Configuration routine can be executed after the SW update is finished.

5.6.2.2 reboot

This reboots the device without any further checks immediately. This can be usefull, if the main payment application has problem and/or is not responsive to other KIT commands. There are some special cases when even this routine can have no effect on the device.

5.7 Transaction execution

In order that a transaction can be executed on the connected terminal, a transaction object has to be created beforehand. For each type of transaction, the KIT driver makes available a static method with corresponding parameters for the creation of a transaction object.

The transaction object must be given to the method Device.doTransaction(), in order that a transaction can be executed on the terminal.

Once the transaction execution has been successfully executed, the transaction needs to be confirmed via the command Transaction.commit() din order for it to be deemed financially valid.

5.7.1 E-Signature handling

Whether a signature is needed for the transaction is determined by terminal and is dependent on the data provided by acquirer and card issuer during the transaction processing. There are methods, which provide information about cardholder's verification of the current transaction. Transaction.CVMResult.isPINVerified() and Transaction.CVMResult.isSignatureVerified() provides information, which cardholder verification method was actually executed for the transaction. It can be none, one, or both of them.

The terminal can display an e-signature dialog for customer to sign on the touchscreen in case a paper signature would be necessary. This behavior is controlled by Terminal Management System and the information is transferred to the terminal during terminal configuration. Whether the terminal actually shows the e-signature dialog during cardholder verification can be find out with Device.isESignatureActive().

5.7.2 Transaction data change

The transaction data can be changed even after the transaction has been already requested. This can be done only if a special reason occurs on the terminal and the terminal is configured to trigger this event. The configuration of the reasons, which should trigger transaction data change request, can be done on the device object and will be effective with the next connect command (notice that a connect command can be sent even if the terminal has a connected status already). The configuration of those reasons can be done with the **device.set/getTransactionDatachangeReasons** or **device.add/removeTransactionDatachangeReason**.

In case the reason has been triggered, **deviceCallbacks.onTransactionDatachangeRequest** will be triggered and the terminal awaits a **transactionDatachange.response** command to be called with the changed transaction data. It is a good idea to use the transaction object, which was given back by the callback, for the modified data.

The detailed list of reasons and the date which is being sent with it from the terminal can be found in the description of the TransactionDatachange class.

5.8 Terminal deactivation

In order to deactivate the terminal, the shift needs to be closed. This is achieved via the command Device.closeShift().

5.9 Final balance closing

To ensure that all transactions are submitted to the acquirer, a final balance (end of day) closing has to be carried out on the device. For this, the method device.finalBalance() is made available. This command submits all unsubmitted transaction to the acquirer and closes the shift.

6 Asynchronous processing

6.1 Introduction

All commands which result in communication via the network to the device are executed asynchronously. The result is returned to the application via DeviceCalback and TransactionCallback. The application is responsible for the callback functions being correctly implemented.

It is not allowed to call methods which can trigger a callback within another callback in this case an exception will be thrown. E.g there cannot be a call to doTransaction() inside of the implementation of onConnect() callback.

For each command, there is a corresponding event function in the CallbackDevice for the reception of the asynchronous activation's result, e.g. for the function Transaction.commit() the result is returned in the event TransactionCallbacks.onCommit(). Within the KIT, there are two callback classes: the class DeviceCallbacks for device events and the class TransactionCallbacks for the transaction events.

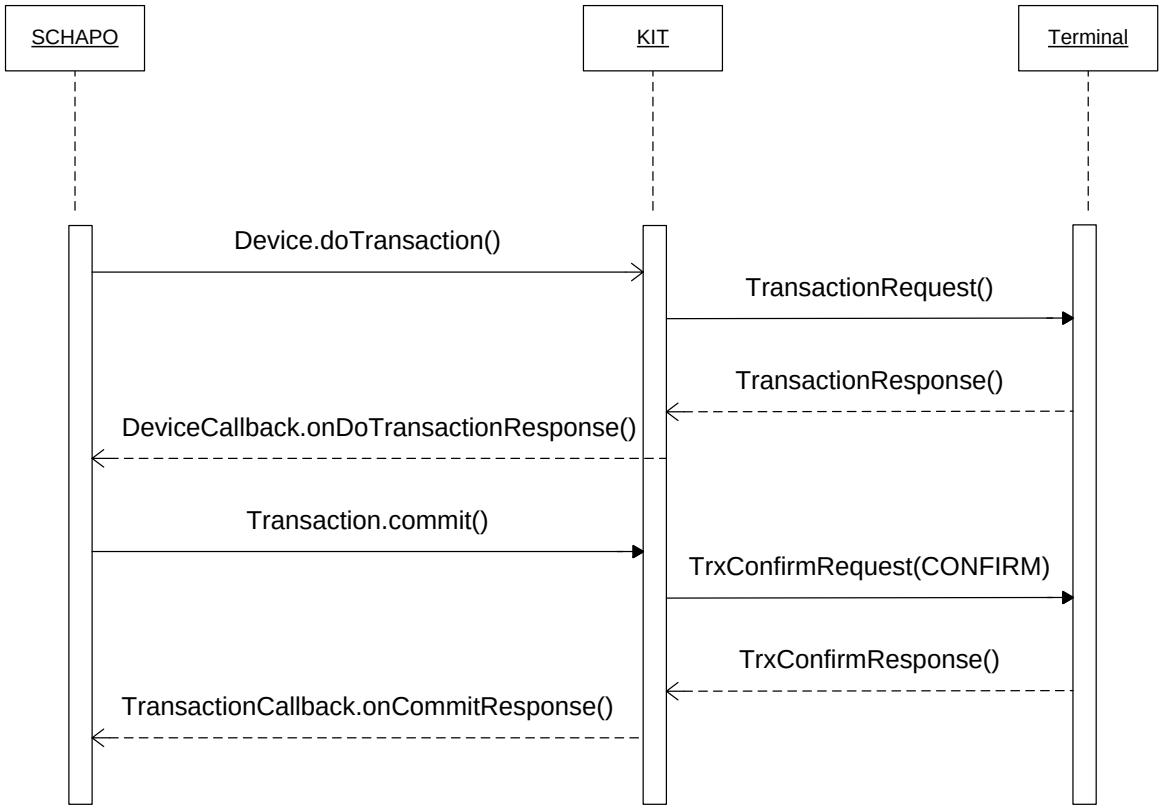
6.1.1 Application timeouts

Normally after calling an asynchrounous method, which sends a request to terminal a corresponding response from terminal is expected. There are some situations (e.g. application software freeze on the terminal side), which cannot be detected through heartbeat watchdog. In this case an internal application timeout occures which calls onError callback with corresponding Error object.

In case there are more than one request pending and an error occurs for one of them the timeouts for the others will be cancelled, too.

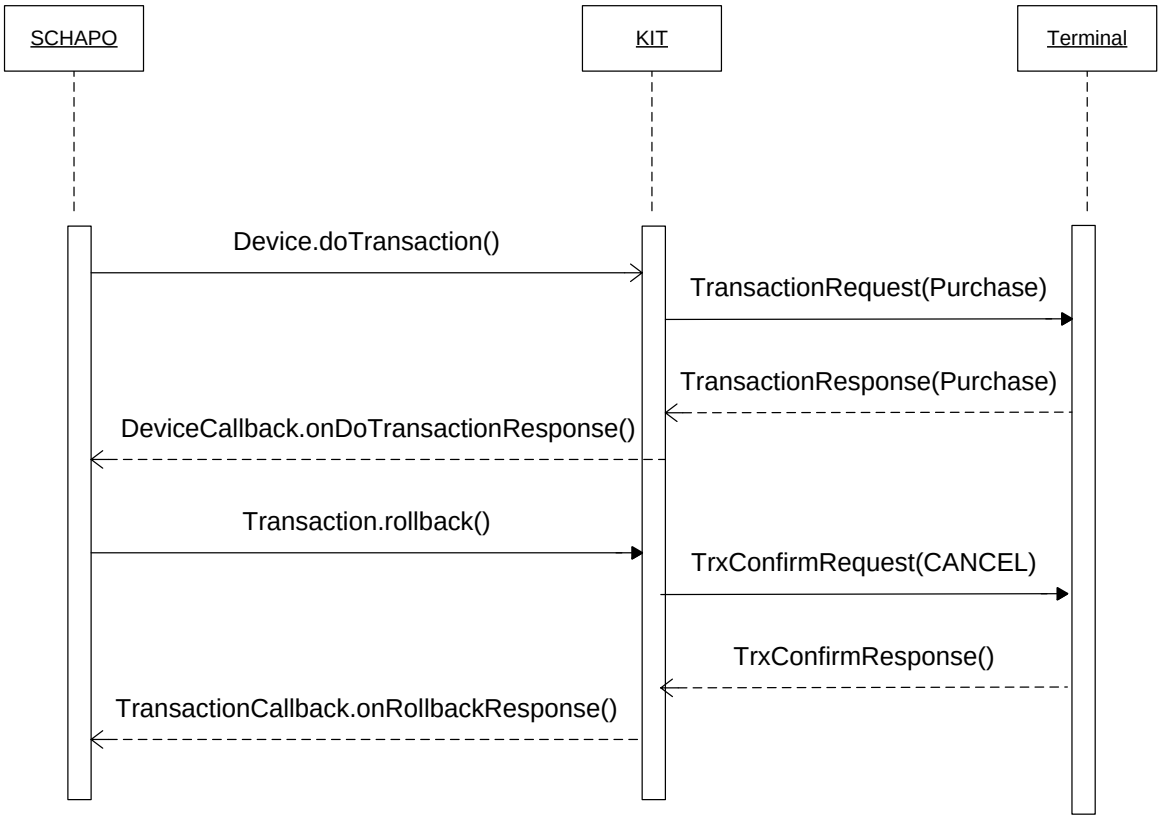
6.2 Sequence transaction processing with Commit

The diagram below demonstrates a successful transaction processing procedure, in which the cash registers confirms the transaction with Commit().



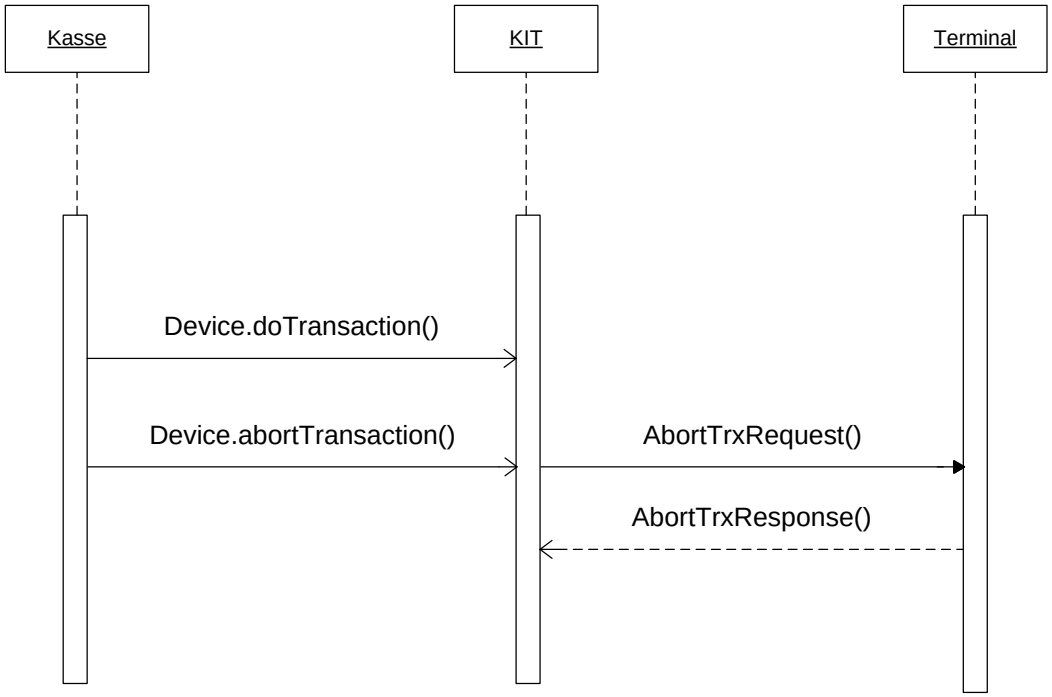
6.3 Sequence transaction processing with Rollback

The diagram below, a transaction already successfully executed on the device is reversed via Rollback(). The KIT sends the terminal a termination message.



6.4 Sequence transaction processing with Abort

While transaction processing is running, the drive can terminate the transaction execution at any time via the synchronous command Transaction.abortTransaction(). The KIT ends the processing immediately and internally ensures that the transaction on the device will be accordingly terminated. The pending callback for doTransaction() is no longer returned.



7 Error correction

7.1 Introduction

This chapter deals with procedure regarding errors which can occur when using the KIT cash register driver. Here, it is necessary to differentiate between application and program errors.

7.2 Program errors

Program errors are defined as errors which occur due to invalid function calls or erroneous program sequences. In these cases, the KIT driver generates exceptions. The possible exceptions are detailed in part 7.4.

Example:

```
Exception in thread "main" com.paytec.kit.KITExceptionConfigFileInvalid:
Configfile missing or not accessible
```

7.3 Application errors

Application errors occur after problems with business processes. When errors occur on the terminal, they are sent back to the application via the event onError(). This error occurs also in case of missing reponse from the terminal on application level. This event contains the error object (see 0) with detailed error data.

7.4 Exceptions

All program errors are returned back to the main program as KITException, which is derived from Exception. There are various sub-classes of KITException which are explained below. The Function column contains all functions in which this KITException can occur.

KITException	Description	Function
KITException ConfigFileInvalid	Configuration file does not exist or has invalid format	KITDriver:getDeviceIDFromConfig, createDevice
KITExceptionConfig StreamInvalid	Configuration data does not exist or has invalid format	KITDriver:getDeviceIDFromConfig, createDevice
KITException ExecutionInCallback	Occurs when method enacting callback with another callback is activated	Device:closeShift, configure, doTransaction, ejectCard, finalBalance, initialize, openShift, requestDeviceStatus, showDialog, startUIExcl, stopUIExcl, abortTransaction Transaction:commit, rollback
KITException InvalidArgument	False or incomplete data was transmitted to a method	Device:showDialog,doTransaction, Transaction:setAppExpDate, setAppPAN, setCVC2, setPosEntryMode
KITExceptionIvalidO peration	Occurs in case a operation was executed from the API, which caused an invalid state of the driver, e.g. a request was called, to which no corresponding callback was implemented in the application	Device:initTransaction, requestReceipt, requestReport TransactionDatachange: response
KITException ProtocolNotSupport ed	In the URL, a cash register protocol not supported by KIT was given. At present, only the value `kitp://` is supported	Device:connect
KITException UIExclusiveModeErr or	The exclusive mode for the GUI display was falsely applied	Device:abortDialog, showDialog, startUIExcl

KITException	Description	Function
KITExceptionNotSupported	This exception is called when an operation is currently not supported, this can be due not fully implemented functionality or due functionality which is in current combination driver-device software not supported and would deliver wrong information, there is a possibility to suppress this exception through emulating the values in config file when applicable	Transaction:getCVMResults,getExtendedResults

8 Platform specific information

8.1 Java edition

The KIT is originally developed and implemented in Java. The minimal tested Java Runtime Environment version is 1.5.12. Further details can be found in sample tester application for Java.

8.2 .NET edition

The .NET edition is compiled after every Java edition update anew using IKVM. IKVM is a tool, which can convert Java byte-code into CLR op-code. For this to be possible all needed Java classes must be available in CLR op-code format, too (part of the IKVM). The complete list of all needed libraries and further details on current version of IKVM product can be found in sample tester application for .NET.