



Monek Bridge Specification

Technical Specification for the Monek Bridge Library

For use with the .net UpgBridge.dll and UpgBridgeForCom.dll

Version 2.7.4. Monek Limited. All rights reserved.

For further help, telephone +44 (0) 345 269 6645 or email support@monek.co.uk



Introduction

Purpose

This document provides a detailed understanding of the technical architecture of the proposed solution. Its purpose is to scope the development work required for partners in their respective businesses.

Processing Requirements

The solution involves the POS till application collecting transaction information and submitting it to a PED, via the Monek Bridge. The PED will pass the transaction data in APACS 40 format to Monek who will transmit to the acquiring bank for authorisation. Their response will be routed back to the PED in APACS 40. The PED will then relay the response to the POS till application, via the Monek Bridge. The solution will ensure that card data is processed to PCI DSS standards and that PAN data is not stored by the partner or transmitted across the network between the PED and the POS till system.

The POS till application will integrate with the Monek Bridge, which will in turn manage the processing of transaction data with the PED and subsequently Monek.

PCI DSS Compliance Notes

The following notes describe how the design of the solution relate to the requirements of PCI DSS compliance.

1. All processing for each Chip and PIN or Swiped transaction is handled directly on the PED.
2. Cardholder data is never transmitted between the PED, Monek Bridge or POS software.
3. Authorisation messages transmitted to the Payment Gateway Client are encrypted using industry standard SSL/TLS.
4. All handling of the authorisation message is done by Monek PCI DSS level 1 accredited systems.

Confidentiality

The contents of this document may not be reproduced without the written consent of Monek.

Contents

Introduction	2
Purpose	2
Processing Requirements	2
PCI DSS Compliance Notes	2
Confidentiality	2
Technology Overview	4
Monek Bridge	4
PED	4
Architecture	5
Component View	5
Till Components	5
PED Components	5
Monek Components	6
Process Overview	7
Transaction Types	7
Transaction Processes	8
Overview	8
Basic Process	8
Monek Bridge Interface	9
Terminal class	9
TransactionDetail class	17
TerminalConfiguration class	19
Enumerators	20
Appendix A: Messages	23
Values	23
Appendix B: Glossary	24

Technology Overview

Monek Bridge

The Monek Bridge component will be supplied by Monek and will provide a high-level interface to the PED. This will remove the complexity involved in the management of the device and transactions.

PED

The PED being used for this application is a Spire terminal.

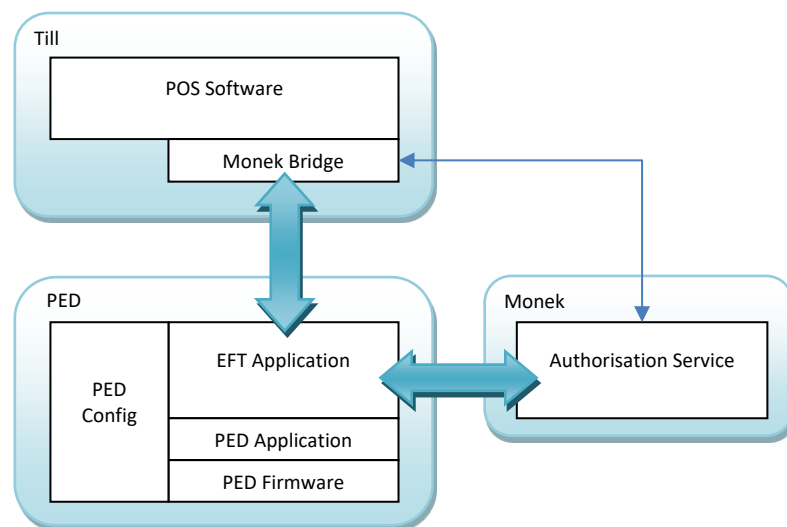
Physical communication between the PED and the till will be via Ethernet connection.

An Ethernet connection with routed access to Monek is required so it is suggested that the tills also utilise this link for communications with the POS.

Architecture

Component View

The diagram below shows the components that make up the solution and the main communication paths between them. The remainder of this section describes the most important components that make up the solution.



Till Components

POS Software

The software is the POS package running on the tills and will communicate with the PED via the Monek Bridge.

Monek Bridge

The Monek Bridge will be supplied by Monek and will provide a high-level interface to the PED. This will remove the complexity involved in the management of the device.

The Monek Bridge will still function when the PED is not present or configured. This will permit a call to be made to check if the PED is available and allow for consistent Error responses in scenarios where a PED is not available.

PED Components

PED Configuration

The PED Configuration Files are a set of merchant specific files that contain the encryption keys and configuration settings to determine how a transaction is processed. Many of these settings are merchant specific and relates to floor limits and rules for risk management.

PED Application

The PED Application is a software layer in the PED that manages transactions.

PED Firmware

The PED Firmware is a software layer in the PED that provides low level services to communicate with the PED hardware.

Monek Components

Authorisation Service

The Authorisation Service is the service provided by Monek for the authorisation of credit card transactions. Responsibility for communication with this service will be with the Monek Bridge which will be supplied by Monek.

Process Overview

Transaction Types

When looking at the different types of transaction that can be processed they fall into two general categories. The categories are:

- Initiated Transactions: Transactions that are initiated by the POS.
- Automatic Transactions: Transactions that are initiated automatically in response to a previous transaction.

Of the transaction types identified each fall into one of the categories as follows:

Transaction Type	Description	Initiated	Automatic
Sale	A standard POS sale transaction, this could be Chip and PIN, Chip and Signature or swipe and Signature.	✓	
Refund	A standard POS refund transaction, this could be Chip and PIN, Chip and Signature or swipe and Signature.	✓	
Referral	This is a transaction type that is processed automatically and is the resubmission of a transaction as the result of a referral. It uses information from the parent transaction in addition to the bank provided authorisation code.		✓
Sale Reversal	This is a transaction type that is processed automatically and is usually the result of a signature problem. It uses information from the parent transaction that is being reversed.		✓
Refund Reversal	This is a transaction type that is processed automatically and is usually the result of a signature problem. It uses information from the parent transaction that is being reversed.		✓

Transaction Processes

Overview

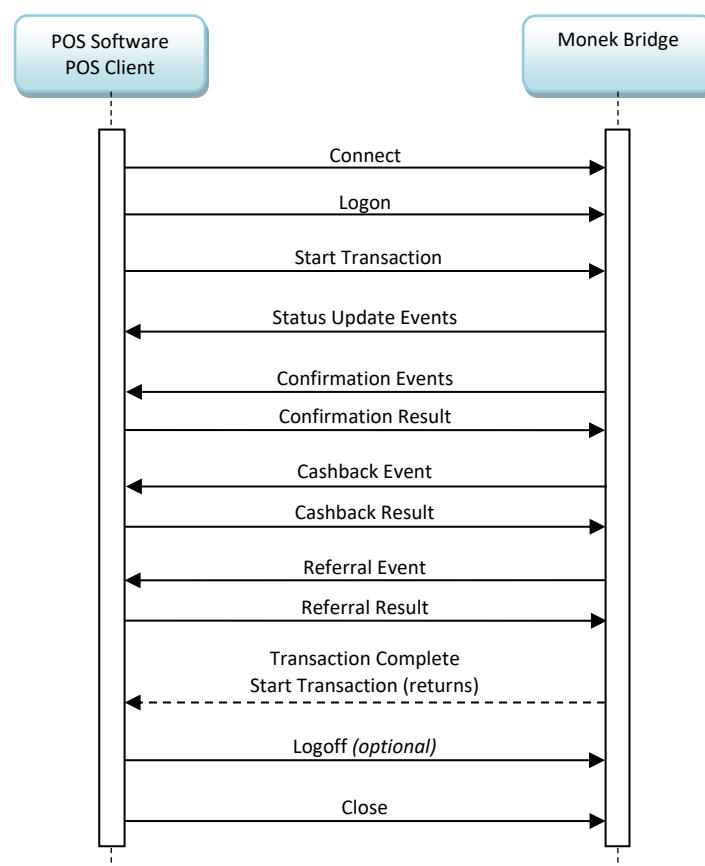
Transactions require the PED to be attached to the till via the network and are for cardholder present transactions, this covers:

1. Chip and PIN Transactions
2. Chip and Signature Transactions
3. Swipe and Signature Transactions

Once the transaction has been initiated the process is managed by the Monek Bridge with events being raised at various stages to allow the POS software to update the status of the transaction.

Basic Process

The following diagram shows the basic process followed during a PED Transaction.



For simplicity the above diagram does not include details of the user interaction with the PED.

If you receive a Confirmation, Cashback, or Referral event, you will need to pass information back to the Monek Bridge. For example, if you receive a message asking for a signature confirmation, you will need to send a response that is Boolean (these are all listed in the ConfirmationTypes enumerator).

Monek Bridge Interface

Terminal class

The Terminal classes provide properties, methods and events to support transaction processing. The terminal class group is comprised of a TerminalBase base class defining common properties, methods and core functionality and derived classes that implement the specific requirements for your target terminal.

Terminal Classes

The following terminal specific classes are available in the Monek Bridge.

Class Name	Description
HypercomT4220	For Hypercom/Spire T4220 series terminals. [Obsolete]
SpireSPx	For Spire SPx series terminals.

Properties

Name	Type	Description
AllErrors	List<Exception>	A collection of all
LastError	Exception	Indicates the type of transaction
TerminalConfiguration	TerminalConfiguration	Allows configuration of terminal connectivity and configuration details.
TransactionDetail	TransactionDetail	Provides information about the active or recently completed transaction.
TerminalVersion	string	Provides the terminal version as reported by the device.

Methods

Cancel

Attempts to cancel the current terminal command.

Note: Currently only supported for SharedLogin.

Syntax

```
void Cancel()
```

Usage

This method should be called when a previous command is still in progress which will attempt to cancel that command.

Clear

Clears all transaction details.

Syntax

```
void Clear()
```

Usage

This method should be called once a transaction is fully completed. All transaction data will be automatically cleared when calling the StartTransaction method.

Close

Closes the connection to the terminal.

Syntax

```
void Close()
```

Usage

This method can be called once a transaction is fully completed.

Note: It is not necessary to log off the terminal prior to closing the connection.

Connect

Open a connection to the terminal using communication settings specified in the TerminalConfiguration object.

Syntax

```
void Connect()
```

Usage

This method must be called prior to any other communication. Once connected the device will remain connected until the Close method is called.

GetDefaultSettings

Returns a TerminalConfiguration object populated with the default connectivity setting for the current terminal.

Syntax

`TerminalConfiguration` `GetDefaultSettings()`

GetPostTransactionDetail

Additional transaction detail may be available post authorisation.

Syntax

`bool` `GetPostTransactionDetail()`

Returns

`bool`. Indicates if additional data was retrieved from Monek.

Usage

This is retrieved automatically prior to a referral event and before the *StartTransaction* method returns. If the information is not immediately available, then the *GetPostTransactionDetail* call may be used to request the data again.

GetTransactionDetail

Retrieves additional transaction detail for a previously completed transaction.

Note: This call does not require connectivity to a terminal and can be called independently to allow processing from back office systems. It does require a correctly configured API Key for the target Merchant ID.

Syntax

`bool` `GetPostTransactionDetail(merchantId, terminalId, receiptNumber, amount)`

Parameters

Name	Type	Description
merchantId	string	The Monek Merchant ID used to perform the required transaction.
terminalId	String	The Terminal ID used to perform the required transaction.
receiptNumber	Int	The Receipt Number from the required transaction.
amount	Int	The amount, in minor currency, of the required transaction.

Returns

`bool`. Indicates if original transaction data was retrieved from Monek.

Additional transaction details retrieved will be available on the *TransactionDetail* object.

Usage

Additional transaction information is typically available during the transaction, however there are some use cases where it may not be. As an example, an offline authorised contactless transaction will not be allocated a Monek Cross Reference until it is sent online.

The *GetTransactionDetail* method can be used to request these transaction details at a later point in time.

One approach would be to use this during end-of-day processing. The standard *ReconcileTotals* call will ensure all offline transactions have been processed and the *GetTransactionDetail* method can then be used to request the additional details.

Login

Logs the POS into the terminal on the active connection.

Syntax

```
bool Login()  
bool Login(password)
```

Parameters

Name	Type	Description
password	string	[optional] If not specified then password set in TerminalConfiguration is used.

Returns

bool. Indicates if logon was successful.

Usage

This method must be called prior to commencing a transaction. Once logged in the device will remain logged in until the *Logout* method is called. Calling the *Close* method will not log the client out, subsequent connections will resume the active session.

The *Login* method can be called on a device that is already logged in and will behave in the same way.

Logout

Logs out of the terminal on the active connection.

Syntax

```
bool Logout()
```

Returns

bool. Indicates if logout was successful or if the device was already logged out.

Note: It is not typically necessary to log off the terminal in between transactions.

ReconcileTotals

Reconciles the internal transaction totals on the terminal.

Syntax

`bool ReconcileTotals()`

Returns

bool. Indicates if cleardown of reconciliation was successful.

Usage

This method can be used at any time to clear down the totals on the terminals.

This must be run at the end of each day after trading has completed. This function can be run at any point between the last transaction of the day and midnight.

Additionally, this can be configured to run automatically on the PED. When run directly from the PED, a nightly report is printed from the internal paper roll.

SharedLogin

Logs the POS into a shared terminal on the active connection. If the terminal is in use logon attempts will continue until the terminal becomes available or the configured SharedLoginTimeout is reached.

Syntax

`bool SharedLogin()`
`bool SharedLogin(password)`

Parameters

Name	Type	Description
Password	string	[optional] If not specified then password set in TerminalConfiguration is used.

Returns

bool. Indicates if logon was successful.

Usage

This method must be called prior to commencing a transaction. Once logged in the device will remain logged in until the *Logout* method is called. Calling the *Close* method will not log the client out, subsequent connections will resume the active session.

The *Login* method can be called on a device that is already logged in and will behave in the same way.

StartTransaction

Begins a new transaction.

Syntax

`TransactionResult StartTransaction(transactionType, amount)`
`TransactionResult StartTransaction(transactionType, amount, password)`
`TransactionResult StartTransaction(transactionType, amount, currencyCode)`
`TransactionResult StartTransaction(transactionType, amount, currencyCode, password)`

Parameters

Name	Type	Description
transactionType	TransactionType	The transaction type to start
amount	int	The amount for the transaction
currencyCode	short	[optional] Currency code for transaction. If not specified the default terminal currency set in TerminalConfiguration is used.
password	string	[optional] Password, if required for transaction type. If required but not specified then default password set in TerminalConfiguration is used.

Returns

TransactionResult. Indicates the result of the transaction.

Comments

If any errors occur this method will return TransactionResult.Error. All error details, either handled or terminal are available from the LastError and AllErrors properties.

UpdateFirmware

Instructs the device to call the terminal management system and perform any required software or configuration updates.

Syntax

```
bool UpdateFirmware()
```

Returns

bool. Indicates if the update process was successfully initiated.

Usage

This method can be used at any time to check for software or configuration changes on the terminal management hosts. It is not typically required as standard terminal configuration automatically performs this check on a weekly basis.

Note: Calling this method will disconnect the Bridge and will typically reboot the terminal. To reconnect after the check, call the Connect and Login methods.

Events

Cashback

Indicates that the cashback is available for the card used in this transaction. To add cashback to the active transaction set the requested value in the Amount field using minor currency notation. e.g. for £20 cashback set Amount to 2000.

Where cashback is not required the amount should be set to 0.

Syntax

Cashback([object](#) sender, [CashbackEventArgs](#) e)

CashbackEventArgs Properties

Name	Type	Description
Amount	int	Return field indicating the cashback amount requested; or 0 where no cashback is required.

ConfirmationRequest

Indicates that the transaction requires merchant confirmation for a particular stage of the transaction. Receipt data may be available at this point to allow the POS to print the merchant or customer receipts and request a signature as appropriate. The transaction will be suspended until the event returns and will continue based on the contents of the "continue" property.

The transaction will be completed or reversed automatically.

Syntax

ConfirmationRequest([object](#) sender, [ConfirmationRequestEventArgs](#) e)

ConfirmationRequestEventArgs Properties

Name	Type	Description
ConfirmationType	ConfirmationType	Indicates the transaction event that requires confirmation.
Continue	bool	Return field indicating if the event is successful and the transaction should continue.

ProcessReferral

Indicates that the transaction has been referred and provides information to process the referral. The transaction will be suspended until the event returns and will continue based on the contents of the "AuthorisationCode" property. If an authorisation code is supplied the transaction will continue, if none is supplied the transaction will be cancelled.

The referral response request or reversal will be carried out automatically.

Syntax

ProcessReferral([object](#) sender, [ProcessReferralEventArgs](#) e)

ProcessReferralEventArgs Properties

Name	Type	Description
Message	string	Message associated with referral
PhoneNumber	string	Referral phone number
AuthorisationCode	string	Return field indicating authorisation code provided during referral processing

StatusUpdate

Provides feedback of transaction status changes

Syntax

StatusUpdate([object](#) sender, [StatusUpdateEventArgs](#) e)

StatusUpdateEventArgs Properties

Name	Type	Description
Message	string	Message indicating current device status
Status	TransactionStatus	Indicates the current transaction status

TransactionDetail class

The TransactionDetail class provides information used to print the customer and merchant receipts.

Properties

Name	Type	A ¹	R ²	Description
Amount	long	r	m	Transaction amount
ApplicationCryptogram	string	r	m ³	Application cryptogram
ApplicationID	string	r	m ³	ICC application number
AuthorisationCode	string	r	m ³	Transaction authorisation code
AuthorisationMessage	string	r		Transaction authorisation message
CardDescription	string	r	m ³	Card type description
CardNumber	string	r	m	Obscured card number
CardNumberHashed	string	r		Cryptographic card hash
CardInputMethod	CardInputMethod	r	m	Indicates the card input method
CardSubType	string	r		
CardType	CardType	r		Enumerated card type
CardTypeCode	string	r		The Monek card type code
CrossReference	string	r		Transaction cross reference
CurrencyCode	short	r	o	Transaction currency code
ExpiryDate	string	r	o	Card expiry date
IsComplete	bool	r		Indicates if the active transaction is complete
IsPinOK	bool	r	o ⁴	Indicates if the PIN was successfully checked
IsReferred	bool	r		Indicates if this transaction was referred
IsSignatureOK	bool	r		Indicates if a valid signature was obtained
IssueNumber	string	r		Card issue number (retired)
MerchantID	string	r	M	Acquirer merchant number
Message	string	r		General transaction message

PanSequenceNumber	string	r	o	The PAN sequence number from a Chip card.
ReceiptNumber	int	r	m	The receipt number for this transaction
ReferralPhoneNumber	string	r		The phone number for a referred transaction
StartDate	string	r	o	Card start date
TerminalID	string	r	m	Terminal identifier
TransactionResult	<i>TransactionResult</i>	r		Indicates the transaction result
TransactionTime	String	r	m	Transaction date and time
TransactionType	<i>TransactionType</i>	r	m	Indicates the type of transaction

*1 Field access level. r = Read, w = Write, r/w = Read/Write

*2 Receipt detail indicator.

*3 Mandatory if available/applicable.

*4 Receipt should display "Verified by PIN" if PIN was successfully validated.

TerminalConfiguration class

The TerminalConfiguration class is used to provide configuration details for the connected terminal.

The TerminalConfiguration class is primarily used to configure communication settings between the POS and the Terminal. Low level terminal details such as acquirers, merchant details and processing rules are all maintained centrally by Monek.

Properties

Name	Type	A* ¹	Description
CurrencyCode	short	r/w	Default currency code for this terminal.
DebugLogPath	string	r/w	Sets the path to write debug logs. Log files are generated as daily date stamped files in the form: <i>UpgBridge_yyyymmdd.log</i>
EnableDebugLogging	bool	r/w	Indicates if debug logs should be created.
Hostname	string	r/w	Ethernet hostname or IP Address
LoginTimeout	int	r/w	Login timeout in milliseconds
Password	string	r/w	Terminal password
Port	int	r/w	Ethernet port number
PrintReceiptsOnTerminal	bool	r/w	Indicates if the terminal should print receipts
PrintWidth	byte	r/w	Terminal print width
SharedLoginTimeout	int	r/w	Shared Login timeout in milliseconds
TransactionTimeout	int	r/w	Transaction processing timeout
UpgApiKey	string	r/w	API key for post transaction data calls
UpgInformationHosts	List<IPEndPoint>	r/w	Monek location of hosts for extra transaction information required for your records and receipt printing.

*¹ Field access level. r = Read, w = Write, r/w = Read/Write

Enumerators

CardInputMethod

Indicates the card input method.

Values

Value	Description
Keyed	Card details were keyed
Swiped	Card details were read from the magnetic strip
Chip	Card details were read from the ICC
Contactless	Card details were read using the contactless interface

CardType

Indicates the card type.

Values

Value	Description
Unknown	Unknown card type
Visa	Visa credit card
VisaDebit	Visa debit card
Electron	Visa Electron card
MasterCard	MasterCard credit card
MasterCardDebit	MasterCard debit card
Maestro	International Maestro card
MaestroUK	UK domestic Maestro card
Solo	Solo card
Laser	Laser card
JCB	JCB card
AmericanExpress	American Express card
Diners	Diners Club card
Other	Other unidentified card

ConfirmationType

Specifies the type of confirmation requested.

Values

Value	Description
SignatureOkay	Card holder signature confirmation is required. Note: If this event is fired then the cardholder merchant receipt should be printed at this point to capture the signature.
AllowPinBypass	Customer PIN was incorrect or unavailable. The terminal is requesting authorisation to bypass PIN confirmation.
ReceiptPrintedOkay	The terminal is requesting confirmation that the merchant receipt has been printed okay.
RemoveReceiptConfirmation	The terminal is prompting for confirmation that the merchant receipt has been removed.

TransactionStatus

Indicates the transaction status reported by the StatusUpdate event.

Values

Value	Description
StatusResponseError	The terminal issued a status update but there was an error decoding the message. Check Terminal.LastError for details.
Diagnostic	Diagnostic messages
DisplayOnly	Terminal display messages
InsertCard	Insert card prompt
SwipeCard	Swipe card prompt
CardAccepted	Card input was accepted
CardRejected	Card input was rejected
ApplicationSelect	Chip application select prompt
AwaitingCustomerInput	Terminal is awaiting customer input
AwaitingPinInput	Terminal is awaiting PIN input
PinAccepted	PIN has been accepted
PinFailed	PIN has failed

Connecting	Terminal is connecting to auth hosts or TMS
Authorising	Terminal is authorising a transaction
Referred	Transaction has been referred
AwaitingSignature	Terminal is awaiting signature confirmation
Reversing	Transaction is being reversed
Cancelling	Transaction is being cancelled
RemoveCard	Remove card prompt

TransactionResult

Indicates the result of a completed transaction.

Values

Value	Description
NotStarted	The transaction has not started and is the default value
InProgress	The transaction is in progress
Authorised	The transaction was authorised
Declined	The transaction was declined
KeepCard	The transaction was declined, and the card should be retained if possible.
Void	The transaction was cancelled
Error	An error occurred attempting to process the transaction
ResponseSequenceError	Internal Error, the transaction has FAILED.
InvalidTransactionType	The transaction passed to the bridge is not supported.

TransactionType

Indicates the type of transaction being processed.

Values

Value	Description
None	No transaction type specified or no active transaction
Refund	Refund transaction
Sale	Sale transaction

Appendix A: Messages

The following list details the typical transaction message responses where the transaction result is *TransactionResult.Error*.

This is the current exhaustive list of expected error codes. This list may be subject to change as and when the terminal is updated. This should be dealt with on a case by case basis.

Values

Descriptions	
internal procedure error	ECR abort
unsupported command	zero amount not allowed
invalid amount	cash not allowed
command not supported	Date/time invalid
connecting error	transaction not allowed
printer error	Cashback not permitted
card expired	Start Date Missing
fallback not allowed	Cash Ceiling Limit Exceeded
unsupported card	Transaction Ceiling Limit Exceeded
no transactions present	Bad MAC – Call Help Desk
card invalid	Expiry Date Missing
card unknown	Offline Store Full – Call Help Desk
general error	

Appendix B: Glossary

Business Partner	A generic term used to describe Business to Business clients to whom we provide payment solutions and services.
Card Issuer	An organisation which issues payment cards (prepaid, credit and debit) to enable their cardholders to purchase goods and services from merchants.
Card Schemes	Organisations which process payments between merchant acquirers and the banks of purchasers that use their scheme. Principle schemes are Visa and MasterCard, but other schemes may be accepted by the Business Partner.
Management Products	Controls to assist the Project Manager with delivery of a project e.g. a project plan.
Merchant	A business or organisation which accepts payment for goods and services by card.
Merchant Acquirer	An organisation (normally a bank) which works for a merchant to provide a single interface to the principle card schemes to authorise transactions, complete settlement and reconciliation with the issuing bank and manage the payment of all principle card scheme and issuer fees. They also act as an intermediary in the event of card claims, returns and refunds.
PCI DSS	Payment Card Industry Data Security Standards
PED	PIN Entry Device; the terminal used by a cardholder to identify themselves at Point of Sales and confirm their transaction details are correct.
POS	Point of sale. The merchant location where a transaction originates.
UPG 2.1	Messaging API used by the Monek transaction gateway.

This is a generic glossary, not all terms may be directly relevant to this project.