iOS Framework Developer Integration Guide

Version 1.6

PayGuardian iOS Framework Developer Integration Guide

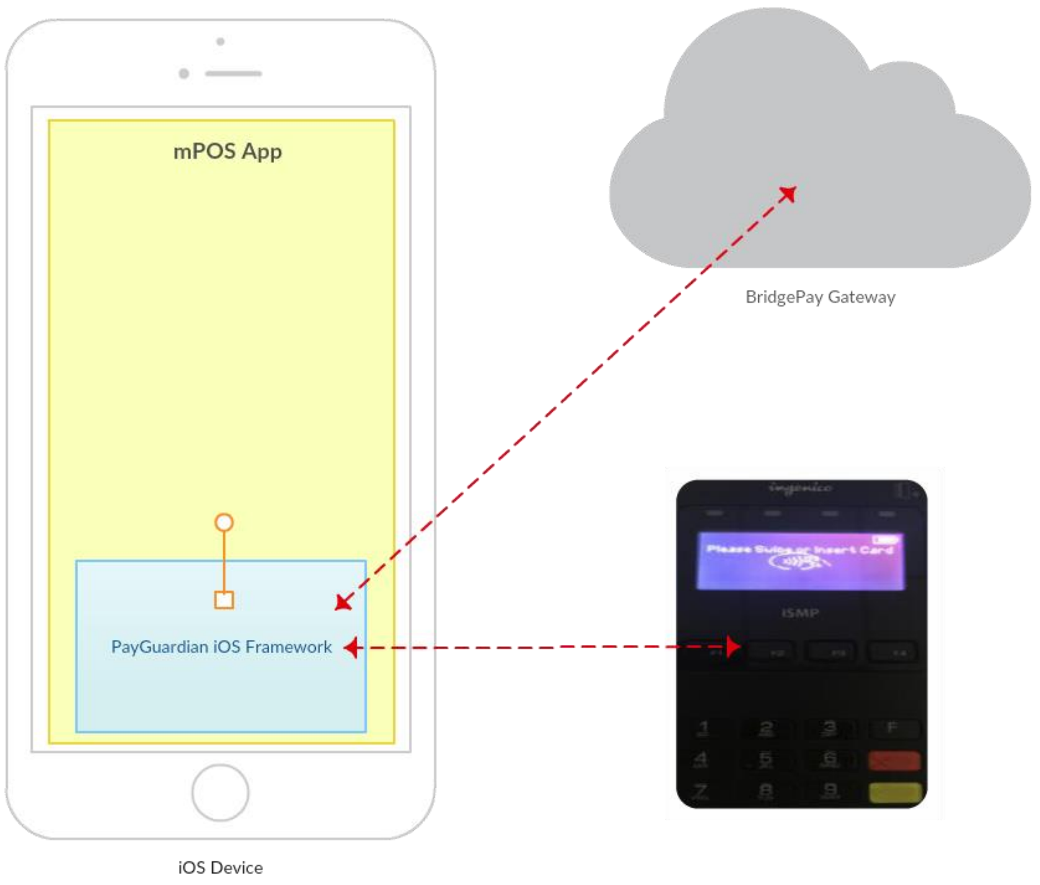
Version 1.6 Released October 20, 2017

Table of Contents

Introduction

Mobile Point-of-Sales (mPOS) systems that process sensitive payment information are required to certify their payment applications to the Payment Application Data Security Standard (PA-DSS). The addition of EMV certification and continued need for both encryption and tokenization has become a concern for both merchants and integrators. Instituting and maintaining these standards requires significant financial and employee resources in order to adhere to the Payments Card Industry Data Security Standards (PCI DSS) compliance requirements. Subsequently, any changes made in the mPOS system, may require a partial or full recertification, which increases the cost and time to market. BridgePay has engaged these issues through our product line, the Pay Guardian suite, to better serve the needs of our integrators and merchants.

PayGuardian iOS is a light weight, highly secure iOS framework library that integrates seamlessly into mPOS applications. PayGuardian iOS facilitates the transaction process by handling the collection and transmission of sensitive payment information as an out of scope PA-DSS solution, thereby offloading the certification responsibility from merchants and integrators. PayGuardian iOS is EMV enabled, handles point to point encryption, and tokenizes all transactions to ensure transaction processing is seamless and secure.



- ✓ The mPOS application collects transaction data such as the dollar amount, invoice number, tender type, transaction type, merchant account information, etc…
- ✓ The mPOS application constructs a BPNPaymentRequest object and passes it to a new instance of the PayGuardianTransaction object.
- ✓ PayGuardian obtains the card information via the claiming of the mPOS device using the Bluetooth API (re: Ingenico RBA SDK) and validates the transaction request object.
- ✓ The mPOS device prompts to swipe or insert the card and transmits the card information to the PayGuardian application, which captures the card data as a swiped or EMV transaction respectively.
- ✓ PayGuardian constructs the payload request and transmits the transaction request to the Payment Gateway.
- ✓ PayGuardian transmits the transaction response returned from the Payment Gateway back to the mPOS application.
- ✓ The mPOS application implementation responds accordingly to the transaction response.

1. PayGuardian Setup

1.1. Requirements

Operating Systems

PayGuardian has a PA-DSS certification for the following operating systems:

- ✓ iOS SDK 8.0 and above

1.2 iOS Application Setup Guide

1.2.1 Minimum Requirements

- iPad/iPhone device with iOS 8.0 and above.
- iPad/iPhone device must have the Bluetooth connectivity feature.

1.2.2 Project Dependencies

Add the following bundled dependencies to the Linked Libraries and Frameworks section of the Xcode project within the mPOS application.

- PayGuardian_SDK.framework
- Ono.framework
- RBA_SDK.framework
- IDTECH_UniMag.a
- libMTSCRA.a

Add the following bundled dependency to the Embedded Binaries section within the mPOS application.

- Ono.framework

Add the following standard iOS frameworks to the Linked Libraries and Frameworks within the mPOS application.

- CFNetwork.framework
- CoreAudioKit.framework
- CoreAudio.framework
- AudioToolbox.framework
- MediaPlayer.framework
- MessageUI.framework
- AVFoundation.framework
- ExternalAccessory.framework
- CoreTelephony.framework
- CoreBluetooth.framework
- UIKit.framework
- Foundation.framework
- CoreGraphics.framework
- libxml2.tbd
- libstdc++.tbd

> ⓘ The bundled framework files are located in the 'Dist' folder of the **PayGuardian_SDK** project.

1.2.3 Device configuration

> ⓘ iOS device and card reader must be connected through the Audio Jack, Accessory Port, or Bluetooth Pairing connectivity.

1.2.3.1 Ingenico

How to connect iOS Device and Card reader Device?

ICMP and iSMPc Devices

Use Card Reader Version: 15.0.4 RBA

Hit the “F” button four times quickly; the device will beep and return to the Bluetooth Pairing Screen. Bluetooth pairing can begin from the Bluetooth Pairing Screen.

- Restart the device by simultaneously pressing and holding the [.,#*] and [Yellow] keys until the device beeps.
- Wait for the device to display the RBA or UIA initialization screen.
- When the RBA or UIA initialization screen appears, quickly press the [2] [6] [3] [4] [Green] [F] [F] keys.
- If successful, the Functions menu should appear. If the Functions menu does not appear, return to Step 1 and restart the device.
- Select the Bluetooth pairing options and select the iOS device from the list to connect the card reader unit.

1.2.3.2 IDTech

How to connect iOS Device and Card reader Device?

UniMag / Shuttle and UniPay III Devices

- Connect the card reader directly to the Audio Jack on the iOS device.

1.2.3.3 MagTek

How to connect iOS Device and Card reader Device?

iDynamo Device

- Connect the card reader directly to the Accessory Port on the iOS device.

2. Integration Process

2.1. Getting Started

The PayGuardian iOS Framework files must be added to the XCode project prior to the start of an integration.

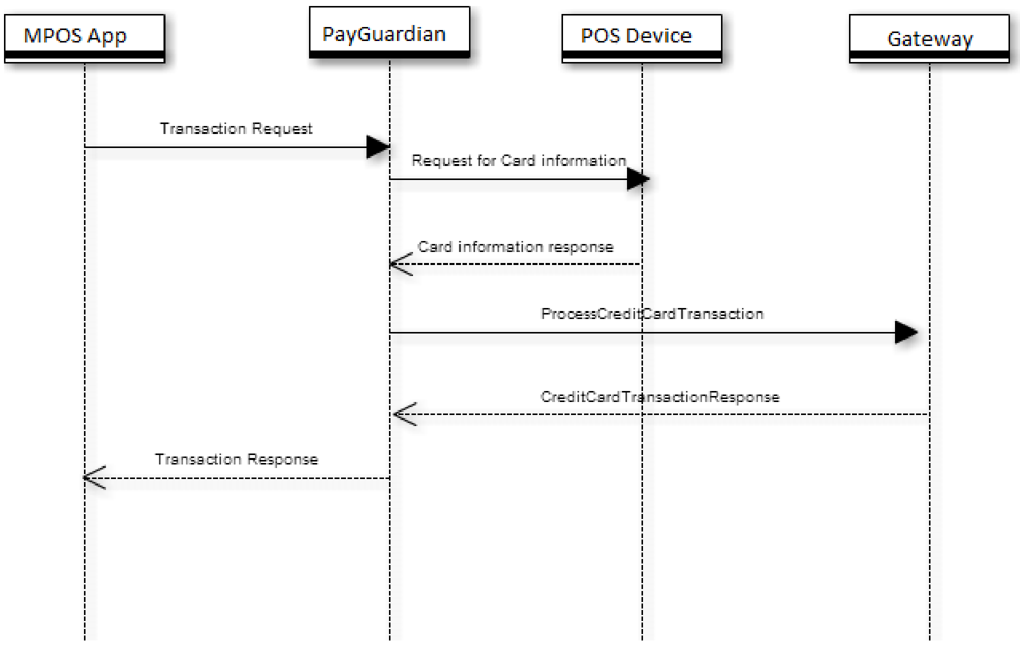
Contact: Developer.Support@bridgepaynetwork.com to receive the developer application build of the PayGuardian iOS Framework library. The developer application build points to the PayGuardian UAT environment for integration and certification testing.

.

2.2. Transaction Flow Summary

The following list summarizes the steps involved in processing a transaction:

1. The mPOS system collects order information and determines that the customer will pay with one of the supported tender types.
2. The mPOS system invokes the PayGuardian iOS Framework library.
3. The payment screen loads and prompts the user to enter customer and payment data.
4. After collecting all transaction information, PayGuardian transmits the data to the server.
5. After receiving a response from the host, the payment screen returns the processing results back to the mPOS system.
6. The mPOS system analyses response data.


MPOS App
PayGuardian
POS Device
Gateway
Transaction Request
Request for Card information
Card information response
ProcessCreditCardTransaction
CreditCardTransactionResponse
Transaction Response

BridgePay

3. PayGuardian Integration

3.1. iOS Framework Integration

Programmatically, a transaction is comprised of a four step process:

- Create a BPNPaymentRequest object.
- Pass the BPNPaymentRequest object to the constructor of a PayGuardianTransaction object.
- Call the runOnCompletion method and provide an implementation for the completion.
- Implement a handler for the returned BPNPayment and/or NSError.

3.1.1. BPNPaymentRequest

The constructor of the BPNPaymentRequest object is:

```
- (instancetype) initInvoiceNumber:(NSString *_Nonnull) invoiceNumber
pnRefNum:(NSString *_Nullable) pnRefNum
amount:(NSDecimalNumber *_Nullable) amount
tipAmount:(NSDecimalNumber *_Nullable) tipAmount
cashBackAmount:(NSDecimalNumber *_Nullable) cashBackAmount
tenderType:(NSString *_Nonnull) tenderType
transactionType:(NSString *_Nonnull) transactionType
username:(NSString *_Nonnull) username
password:(NSString *_Nonnull) password
merchantCode:(NSString *_Nonnull) merchantCode
merchantAccountCode:(NSString *_Nonnull) merchantAccountCode
paymentAccountNumber:(NSString *_Nullable) paymentAccountNumber
token:(NSString *_Nullable) token
expirationDate:(NSString *_Nullable) expirationDate
terminalType:(NSString *_Nullable) terminalType
industryType:(NSString *_Nonnull) industryType
healthCareData:(BPNHealthCare *_Nullable) healthCareData
disableEmv:(BOOL) disableEmv
testMode:(BOOL) testMode;
```

Property	Description	Data Type / Min - Max Values

Amount	Required. Total transaction amount (includes subtotal, cash back, tax, and tip) (Format example: DDDD.CC)	NSDecimalNumber
		Min = 4;
		Max = 12

TipAmount	The tip amount. The tip is not added into the total amount, it is an in-addition-to the specified amount. (Format example: DDDD.CC)	NSDecimalNumber
		Min = 4;
		Max = 12

InvoiceNumber	Required. POS system invoice/tracking number. (Alpha/Numeric characters only)	NSString
		Min = 1;
		Max = 12

TenderType	Required. Valid values are: **TENDER_TYPE_CREDIT,** **TENDER_TYPE_DEBIT,** **TENDER_TYPE_GIFT,** **TENDER_TYPE_LOYALTY**	NSString; Constant
TransactionType	Required. Valid values are: **TRANSACTION_TYPE_SALE:** Makes a purchase with a credit card, debit card, gift card, or loyalty card. **TRANSACTION_TYPE_SALE_AUTH**: (Authorization only) Verifies/authorizes a payment amount on a credit card. **TRANSACTION_TYPE_REFUND:** Returns a credit card or debit card payment. **TRANSACTION_TYPE_VOID**: Removes a credit card transaction from an unsettled batch. **TRANSACTION_TYPE_CAPTURE:** Places a 'TRANSACTION_TYPE_SALE_AUTH' transaction into the open credit batch for settlement. Requires the Original Reference Number and updated Amount. **TRANSACTION_TYPE_TIP_ADJUST**: Modifies previously authorized 'TRANSACTION_TYPE_SALE' transaction to include a TipAmount and updates the transaction total amount. Requires the Original Reference Number. Can only be used on an unsettled transaction.	NSString; Constant – valid values are listed.

Username	Required. Username of the BridgePay Merchant.	NSString;
		Min Value = 5;
		Max Value = 25

Password	Required. Password of the BridgePay Merchant.	NSString Min Value = 7; Max Value = 25
MerchantCode	Required. BridgePay Merchant Code.	NSString Numeric (0-9)
MerchantAccountCode	Required. BridgePay Merchant Account Code.	NSString Numeric (0-9)
industryType	Required. Indicates the Industry type of the merchant assigned to a transaction.	NSString;

	Valid values: **TRANSACTION_INDUSTRY_TYPE_RETAIL** **TRANSACTION_INDUSTRY_TYPE_RESTAURANT** **TRANSACTION_INDUSTRY_TYPE_ECOMMERCE** **TRANSACTION_INDUSTRY_TYPE_DIRECT_MARKETING**	Constant – valid values are listed.
healthCareData	Required. Represents the inclusion of Healthcare related fields. Valid values are: **nil** (indicates no Healthcare related fields present) **healthCareAmt** (NSDecimalNumber). The portion of the Total Amount spent on healthcare related services or products. **transitAmt** (NSDecimalNumber). The portion of the Total Amount spent on healthcare related transportation. **prescriptionAmt** (NSDecimalNumber). The portion of the Healthcare Amount spent on prescription drugs or products. **visionAmt** (NSDecimalNumber). The portion of the Healthcare Amount spent on vision related medical services. **dentalAmt** (NSDecimalNumber). The portion of the Healthcare Amount spent on dental related medical services. **clinicAmt** (NSDecimalNumber). The portion of the Healthcare Amount spent on hospitalization. **isQualifiedIIAS** (NSString). Indicates whether purchase items are verified against IIAS as qualified for healthcare purchases. Can be lower case 'true', or lower case 'false'.	NSString; or NSDecimalNumber (as indicated) Min Value = 1; Max Value = 50
pnRefNum	Gateway transaction ID/Previous transaction reference number. Only required for voids, refunds, and tip adjustments.	NSString Min Value = 1; Max Value = 15
disableEmv	Optional. Used to test non-EMV transactions.	Boolean; Valid Value = Yes or No
CashBackAmount	Not implemented.	NSString
PaymentAccountNumber	Card number.	NSString Min Value = 13; Max Value = 16

expirationDate	Card number expiration date in MMYY format.	NSString;
		Min Value = 4;
		Max Value = 4

ShippingAddress	Reserved for future use.	NSString
terminalType	Optional. Valid values are: **nil** (indicates no terminal used) **TERMINAL_TYPE_INGENICO_BLUETOOTH** **TERMINAL_TYPE_INGENICO_IP** **TERMINAL_TYPE_MAGTEK_IDYNAMO**	NSString; Constant – valid values are listed.

	TERMINAL_TYPE_IDTECH_SHUTTLE **TERMINAL_TYPE_IDTECH_UNIPAYIII** **TERMINAL_TYPE_BBPOS_CHIPPER2** **TERMINAL_TYPE_BBPOS_WISEPAD2** **TERMINAL_TYPE_PAX_D180**	

testMode	Gateway testing mode flag.	Boolean;
		Value = **Yes** or **No**

Token	Represents the tokenized card number received from a token request. Used in place of the PaymentAccountNumber.	NSString;
		Value = 22 characters

3.1.2. PayGuardianTransaction

The construction for a PayGuardianTransaction is:

*initWithPaymentRequest:(BPNPaymentRequest *)request;*

The run method with completion is:

*- (void)runOnCompletion:(void (^)(BPNPayment *payment, NSError *error)) onCompletion onStateChanged:(void (^)(PayGuardianTransactionState state)) onStateChanged;*

The completion must be implemented to capture the results of a transaction request.

3.1.3. BPNPayment

The BPNPayment is the response from a PayGuardianTransaction. It contains two fields: BridgeCommResponse and BPNReceipt. The BridgeCommResponse contains the gateway response fields and the BPNReceipt contains the fields necessary for an EMV compliant receipt.

3.1.4. BridgeCommResponse

Property	**Description**	**Data Type / Min - Max Values**
TransactionId	Transaction authorization code from the payment processor. This value is an approval code for approved transactions or an error code/message for declined transactions.	NSString Max Value = 15
RequestType	The actual amount approved by the host. This may differ from the requested amount.	NSString Max Value = 15

ResponseCode	BridgeCommResponseCodeSuccess = 0
	BridgeCommResponseCodeCancel = 1
	BridgeCommResponseCodeError = 2
	BridgeCommResponseCodeDeniedByCustomersBank = 30032
		Enum
		Max Value = 6

ResponseDescription	Plain text response message.	NSString N/A
Token	Unique token assigned to a transaction.	NSString Max = 22
ExpirationDate	Card expiration date.	NSString Max = 7
AuthorizationCode	Authorization code for a transaction.	NSString Max = 50
OriginalReferenceNumber	Original reference number from a request.	NSString Max = 12
AuthorizedAmount	Amount authorized by the processor.	NSDecimalNumber Max = 12
OriginalAmount	Original requested amount of the transaction.	NSDecimalNumber Max = 12
GatewayTransactionID	Unique ID of a transaction in the gateway.	NSString Max = 12
GatewayMessage	Details from the processor or payment gateway regarding the transaction result.	NSString Max = 50
InternalMessage	Internal Message from the gateway.	NSString Max = 50
GatewayResult	Result message from the gateway.	NSString Max = 5
AVSMessage	AVS Match Result Message.	NSString Max = 10
AVSResponse	AVS Response code.	NSString Max = 10
CVMessage	CVV/CVV2 Match Result Message.	NSString Max = 10
CVResult	CVV/CVV2 Result code.	NSString Max = 10
TransactionCode	Reserved for future use.	NSString
TransactionDate	Date/time of the transaction.	NSString Max = 10
RemainingAmount	Amount remaining of the original amount.	NSDecimalAmount Max = 12
ISOCountryCode	Country code for currency.	NSString Max = 3
ISOCurrencyCode	Currency code for a transaction.	NSString Max = 3
ISOTransactionDate	Date of transaction from the processor.	NSString Max = 10
ISORequestDate	Date of request at a processor.	NSString Max = 10
NetworkReferenceNumber	Unique ID of a transaction at the processor.	NSString Max = 12

NetworkCategoryCode	Processor specific category code.	NSString Processor dependent
NetworkMerchantId	Merchant ID from the gateway.	NSString Processor dependent
NetworkTerminalId	Network terminal ID from the gateway.	NSString Processor dependent
CardType	Reserved for future use. Type of card.	NSString
MaskedPAN	Reserved for future use. Masked card number.	NSString
IsCommercialCard	“True” or “False”	NSString
StreetMatchMessage	Reserved for future use. Message about street match for AVS.	NSString
SecondsRemaining	Reserved for future use.	NSString
MerchantCode	Gateway merchant code used in a transaction.	NSString Numeric (0-9)
MerchantAccountCode	Gateway merchant account code used in a transaction.	NSString Numeric (0-9)
MerchantName	Name of a merchant.	NSString Max = 50
ReceiptTagData	EMV tag data. Processor dependent.	NSString
IssuerTagData	EMV issuer tag data. Processor dependent.	NSString
ApplicationIdentifier	Reserved for future use.	
TerminalVerificationResults	Reserved for future use.	
IssuerApplicationData	Reserved for future use.	
TransactionStatusInformation	Reserved for future use.	

3.1.5. BPNReceipt

The EMV compliant receipt components returned in the BPNPayment object.

Property	**Description**	**Data Type / Min - Max Values**

MaskedCardNumber	PA-DSS masked card number for the transaction.	NSString
		Min = 13;
		Max = 19

ChipCardAID	Reserved for future use.	NSString
Invoice	Invoice number from transaction.	NSString Min = 1/Max = 24
Seq	Sequence number	NSString

AuthorizationCode	Gateway authorization code.	NSString
		Min = 5;
		Max = 50

EntryMethod	Card number entry method.	NSString Max = 50
TotalAmount	Total amount of the transaction.	NSDecimalNumber Max = 12
AppLabel	Name of application	NSString Max = 50

CardHolderName	Name on the card	NSString
		Min = 1;
		Max = 50

NetworkMerchantID	Merchant ID. Dependent on processor.	NSString
NetworkTerminalID	Terminal ID. Dependent on processor.	NSString

CardFirstFour	First four digits of a card	NSString
		Min = 4;
		Max = 4

CardType	Type of card	NSString
		Min = 4;
		Max = 4

3.1.5.1 Sample examples

Basic transaction request / response and receipt output examples.

MSR Sale Type Request

An example of a basic MSR sale transaction request.

```
    _request = [[BPNPaymentRequest alloc]
initInvoiceNumber:@”8888”
pnRefNum:nil
amount:[NSDecimalNumber decimalNumberWithString:@"5.50"]
tipAmount:nil
cashBackAmount:nil
tenderType:TENDER_TYPE_CREDIT
transactionType:TRANSACTION_TYPE_SALE
username:@"emvpgtest"
password:@"57!sE@3Fm"
merchantCode:@"320000"
merchantAccountCode:@"320001"
paymentAccountNumber:nil
token:nil
expirationDate:nil
terminalType:(Valid terminal type here)
industryType:TRANSACTION_INDUSTRY_TYPE_RETAIL
healthCareData:nil
disableEMV:NO
testMode: YES];


_transaction = [[PayGuardianTransaction alloc] initWithPaymentRequest:_request];
```

```
[_transaction runOnCompletion: ^(BPNPayment *payment, NSError *error) {
//process response
} onStateChanged: ^(PayGuardianTransactionState state) {
//process state change
}];
```

MSR Sale Type Response

An example of a basic MSR sale transaction response.

```
BPNPayment.BridgeCommResponse:
transactionID:   8888
requestType:    004
responseCode: 0
responseDescription:    Successful Request
token:   11110000000068530608
expirationDate: 1225
authorizationCode:       290144
originalReferenceNumber:       MCC7754741020
authorizedAmount:       $5.50
originalAmount:            $5.50
gatewayTransactionID:  214973401
gatewayMessage:         A01 - Approved
internalMessage:           Approved: 290144 (approval code)
gatewayResult: 00000
AVSMessage:
AVSResponse:
CVMessage:
CVResult:
transactionCode:           (null)
transactionDate:           20161020
remainingAmount:        $0.00
ISOCountryCode:          (null)
ISOCurrencyCode:         USD
ISOTransactionDate:      2016-10-20 14:32:47.86
ISORequestDate:          2016-10-20 14:32:47.86
networkReferenceNumber:      MCC7754741020
merchantCategoryCode:
networkMerchantID:     518564010126944
networkTerminalID:      PPB01.
cardType:           Mastercard
maskedPAN:      5***********0608
responseTypeDescription:        sale
isCommercialCard:        False
streetMatchMessage:
secondsRemaining:       (null)
merchantCode: (null)
merchantAccountCode:             (null)
merchantName:            (null)
receiptTagData:            (null)
```

```
issuerTagData:  (null)
applicationIdentifier:     (null)
terminalVerificationResults:       (null)
issuerApplicationData:   (null)
transactionStatusInformation:   (null)
```

MSR Sale Receipt

An example of the values returned on a basic MSR Sale receipt.

```
BPNPayment.BPNReceipt:
maskedCardNumber:    5***********0608:
chipCardAID:      (null) //← No AID for non-emv trx
invoice: 8888
seq:      8888
authorizationCode:       290144
entryMethod:   Swipe_Read
totalAmount:     5.5
appLabel:
cardHolderName:          (null)
networkMerchantId:    518564010126944
networkTerminalId:       PPB01.
cardFirstFour:   5413
cardType:          Mastercard
requiresSignature:        YES
pinEntered:       NO
```

Sale_Auth Transaction Type Request

An example of a basic authorization only transaction request.

```
  _request = [[BPNPaymentRequest alloc]

initInvoiceNumber:@”33333”
pnRefNum:nil
amount:[NSDecimalNumber decimalNumberWithString:@"7.50"]
tipAmount:nil
cashBackAmount:nil
tenderType:TENDER_TYPE_CREDIT
transactionType:TRANSACTION_TYPE_SALE_AUTH
username:@"emvpgtest"
password:@"57!sE@3Fm"
merchantCode:@"320000"
merchantAccountCode:@"320001"
paymentAccountNumber:nil
token:nil
expirationDate:nil
terminalType:(Valid terminal type here)
industryType:TRANSACTION_INDUSTRY_TYPE_RETAIL
healthCareData:nil
disableEMV:NO
testMode: YES];


_transaction = [[PayGuardianTransaction alloc] initWithPaymentRequest:_request];
[_transaction runOnCompletion: ^(BPNPayment *payment, NSError *error) {
//process response
} onStateChanged: ^(PayGuardianTransactionState state) {
//process state change
}];
```

Sale_Auth Transaction Type Response

An example of a basic authorization only transaction response.

```
BPNPayment.BridgeCommResponse:
transactionID:   33333
requestType:    004
responseCode: 0
responseDescription:    Successful Request
token:   11110000000077149269
expirationDate: 0325
authorizationCode:       786986
originalReferenceNumber:
authorizedAmount:       $7.50
originalAmount:          $7.50
gatewayTransactionID:  214972201
gatewayMessage:         A01 - Approved
internalMessage:          Approved: 786986 (approval code)
gatewayResult: 00000
AVSMessage:
AVSResponse:
CVMessage:
CVResult:
transactionCode:          (null)
transactionDate:          20161020
remainingAmount:        $0.00
ISOCountryCode:          (null)
ISOCurrencyCode:         USD
ISOTransactionDate:      2016-10-20 14:02:46.153
ISORequestDate:          2016-10-20 14:02:46.153
networkReferenceNumber:
merchantCategoryCode:
networkMerchantID:    518564010126944
networkTerminalID:     PPB01.
cardType:           Visa
maskedPAN:      4***********9269
responseTypeDescription:         auth
```

```
isCommercialCard:        False
streetMatchMessage:
secondsRemaining:        (null)
merchantCode: (null)
merchantAccountCode:             (null)
merchantName:            (null)
receiptTagData:          4F:A0000000031010;95:8080008000;9F10:06010A03A00000;
   9B:6800;91:;8A:
issuerTagData:
applicationIdentifier:     A0000000031010
terminalVerificationResults:      8080008000
issuerApplicationData:   06010A03A00000
transactionStatusInformation:   6800
```

Sale_Auth Receipt

An example of the values returned on a basic authorization only receipt.

```
BPNPayment.BPNReceipt:
maskedCardNumber:   4***********9269:
chipCardAID:     A0000000031010
invoice: 33333
seq:      33333
authorizationCode:       786986
entryMethod:  Chip_Read
totalAmount:    7.5
appLabel:         VISA DEBIT
cardHolderName:          CARDHOLDER/VALUED
networkMerchantId:    518564010126944
networkTerminalId:      PPB01.
cardFirstFour:   4204
cardType:         Visa
requiresSignature:       YES
pinEntered:      NO
```

Capture Transaction Type with Tip Request

An example of a basic capture transaction request that includes a tip amount in the summary total.

```
   _request = [[BPNPaymentRequest alloc]
initInvoiceNumber:@”33333”
pnRefNum:@”214972201”  //gatewayTransactionID from SALE_AUTH
amount:[NSDecimalNumber decimalNumberWithString:@"9.50"] //adjust amount
with tip
tipAmount:[NSDecimalNumber decimalNumberWithString:@"2.00"] //tip amount for
record
cashBackAmount:nil
tenderType:TENDER_TYPE_CREDIT
transactionType:TRANSACTION_TYPE_CAPTURE
username:@"emvpgtest"
password:@"57!sE@3Fm"
merchantCode:@"320000"
merchantAccountCode:@"320001"
paymentAccountNumber:nil
token:nil
expirationDate:nil
terminalType:nil
industryType:TRANSACTION_INDUSTRY_TYPE_RETAIL
healthCareData:nil
disableEMV:NO
testMode: YES];


_transaction = [[PayGuardianTransaction alloc] initWithPaymentRequest:_request];
[_transaction runOnCompletion: ^(BPNPayment *payment, NSError *error) {
//process response
} onStateChanged: ^(PayGuardianTransactionState state) {
//process state change
}];
```

Capture Transaction Type with Tip Response

An example of a basic capture transaction response that includes a tip amount in the summary total.

```
BPNPayment.BridgeCommResponse:
transactionID:    33333
requestType:     019
responseCode: 0
responseDescription:     Successful Request
token:    (null)
expirationDate: (null)
authorizationCode:        (null)
originalReferenceNumber:
authorizedAmount:        $0.00
originalAmount:              $0.00
gatewayTransactionID:  214972201
gatewayMessage:          A01 - Approved
internalMessage:           (null)
gatewayResult: 00000
AVSMessage:    (null)
AVSResponse:  (null)
CVMessage:      (null)
CVResult:          (null)
transactionCode:           (null)
transactionDate:            (null)
remainingAmount:         $0.00
```

```
ISOCountryCode:          (null)
ISOCurrencyCode:         (null)
ISOTransactionDate:      (null)
ISORequestDate:          (null)
networkReferenceNumber:      (null)
merchantCategoryCode:            (null)
networkMerchantID:    (null)
networkTerminalID:      (null)
cardType:          (null)
maskedPAN:
responseTypeDescription:         (null)
isCommercialCard:        (null)
streetMatchMessage:   (null)
secondsRemaining:        (null)
merchantCode: (null)
merchantAccountCode:             (null)
merchantName:             (null)
receiptTagData:             (null)
issuerTagData:  (null)
applicationIdentifier:     (null)
terminalVerificationResults:       (null)
issuerApplicationData:   (null)
transactionStatusInformation:   (null)
```

Capture Transaction with Tip Receipt

An example of the values returned on a basic capture transaction receipt that includes a tip amount in the summary total.

```
BPNPayment.BPNReceipt:
maskedCardNumber:    :
chipCardAID:      (null)
invoice: 33333
seq:      33333
authorizationCode:        (null)
entryMethod:   (null)
totalAmount:     (null)
appLabel:
cardHolderName:          (null)
networkMerchantId:    (null)
networkTerminalId:       (null)
cardFirstFour:    (null)
cardType:          (null)
requiresSignature:        NO
pinEntered:       NO
```

Tip Adjust Transaction Type Request

An example of a basic transaction request that adds or updates a Tip Amount into a Sales Transaction and sums both amounts into the summary total of the transaction. Can only be performed on an unsettled Sale transaction.

```
_request = [[BPNPaymentRequest alloc]
initInvoiceNumber:@"8888"
pnRefNum:@"235531904"  //gatewayTransactionID from SALE
amount:nil    //adjustment will be calculated at gateway from tip
tipAmount:[NSDecimalNumber decimalNumberWithString:@"2.00"] //tip amount
for adjustment
cashBackAmount:nil
tenderType:TENDER_TYPE_CREDIT
transactionType:TRANSACTION_TYPE_TIP_ADJUST
username:@"emvpgtest"
password:@"57!sE@3Fm"
merchantCode:@"320000"
merchantAccountCode:@"320001"
paymentAccountNumber:nil
token:nil
expirationDate:nil
terminalType:nil
industryType:TRANSACTION_INDUSTRY_TYPE_RETAIL
healthCareData:nil
disableEMV:NO
testMode: YES];

_transaction = [[PayGuardianTransaction alloc] initWithPaymentRequest:_request];
[_transaction runOnCompletion: ^(BPNPayment *payment, NSError *error) {
//process response
} onStateChanged: ^(PayGuardianTransactionState state) {
//process state change
}];
```

Tip Adjust Transaction Type Response

An example of a basic tip adjust transaction.

```
BPNPayment.BridgeCommResponse:
transactionID: 8888
requestType: 004
responseCode: 00000
responseDescription: Successful Request
token: (null)
expirationDate: (null)
authorizationCode: 987090
ReferenceNumber:
authorizedAmount: $7.00
originalAmount: $5.00
```

```
gatewayTransactionID: 235531904
gatewayMessage: A12 – Tip Adjustment Posted
internalMessage: Tip Adjustment Posted
gatewayResult: 00000
AVSMessage: (null)
AVSResponse: (null)
CVMessage: (null)
CVResult: (null)
transactionCode: (null)
transactionDate: (null)
remainingAmount: $0.00
ISOCountryCode: (null)
ISOCurrencyCode: (null)
ISOTransactionDate: (null)
ISORequestDate: (null)
networkReferenceNumber: (null)
merchantCategoryCode: (null)
networkMerchantID: (null)
networkTerminalID: (null)
cardType: (null)
maskedPAN:
responseTypeDescription: (null)
isCommercialCard: False
streetMatchMessage: (null)
WalletID: (null)
WalletPaymentMethodID: (null)
WalletResponseCode: (null)
WalletResponseMessage: (null)
```

Sale Transaction Type Request using a Token

An example of a basic sale transaction using a token to replace the payment card.

```
_request = [[BPNPaymentRequest alloc]
initInvoiceNumber:@”33333”
pnRefNum:nil
amount:[NSDecimalNumber decimalNumberWithString:@"8.00"]
tipAmount:nil
cashBackAmount:nil
tenderType:TENDER_TYPE_CREDIT
transactionType:TRANSACTION_TYPE_SALE
username:@"emvpgtest"
password:@"57!sE@3Fm"
merchantCode:@"320000"
merchantAccountCode:@"320001"
paymentAccountNumber:nil
token:@"11110000000077149269" // ← Token from a previous sale or auth
expirationDate:@”0325”  //← expirationDate used with Token
terminalType:nil  // ← terminalType not used with Token
industryType:TRANSACTION_INDUSTRY_TYPE_RETAIL
healthCareData:nil
disableEMV:NO
testMode: YES];

_transaction = [[PayGuardianTransaction alloc] initWithPaymentRequest:_request];
[_transaction runOnCompletion: ^(BPNPayment *payment, NSError *error) {
//process response
} onStateChanged: ^(PayGuardianTransactionState state) {
//process state change
}];
```

Sale using Token Transaction Response

An example of a basic sale transaction response where a token was submitted in the transaction request.

```
BPNPayment.BridgeCommResponse:
transactionID:    33333
requestType:     004
responseCode: 0
responseDescription:     Successful Request
token:    11110000000077149269
expirationDate: 0325
authorizationCode:        099912
originalReferenceNumber:
authorizedAmount:       $8.00
originalAmount:          $8.00
gatewayTransactionID:  214972001
gatewayMessage:        A01 - Approved
internalMessage:         Approved: 099912 (approval code)
gatewayResult: 00000
AVSMessage:
AVSResponse:
CVMessage:
CVResult:
transactionCode:         (null)
transactionDate:          20161020
remainingAmount:        $0.00
ISOCountryCode:          (null)
ISOCurrencyCode:        USD
ISOTransactionDate:      2016-10-20 13:55:27.3
ISORequestDate:          2016-10-20 13:55:27.3
networkReferenceNumber:
merchantCategoryCode:
networkMerchantID:    518564010126944
networkTerminalID:      PPB01.
cardType:
maskedPAN:
responseTypeDescription:         sale
isCommercialCard:        False
streetMatchMessage:
```

```
secondsRemaining:       (null)
merchantCode: (null)
merchantAccountCode:             (null)
merchantName:            (null)
receiptTagData:            (null)
issuerTagData:  (null)
applicationIdentifier:     (null)
terminalVerificationResults:       (null)
issuerApplicationData:   (null)
transactionStatusInformation:   (null)
```

Sale using Token Receipt

An example of the values returned on a basic sale transaction where a token was submitted in the transaction request.

```
BPNPayment.BPNReceipt:
maskedCardNumber:    :
chipCardAID:      (null)
invoice: 33333
seq:      33333
authorizationCode:       099912
entryMethod:   (null)
totalAmount:     8
appLabel:
cardHolderName:          (null)
networkMerchantId:    518564010126944
networkTerminalId:       PPB01.
cardFirstFour:    (null)
cardType:
requiresSignature:        NO
pinEntered:       NO
```

Invalid Tender Type Request

An example of how to submit a sale transaction with an unacceptable tender type.

```
_request = [[BPNPaymentRequest alloc]
        initWithAmount:[NSDecimalNumber decimalNumberWithString:@"7.00"]
        tipAmount:nil
        invoiceNumber:@"12345"
        tenderType:@"abcd" //<----- Invalid Tender Type will fail validation
        transactionType:TRANSACTION_TYPE_SALE
        username:@"emvpgtest"
        password:@"57!sE@3Fm"
        merchantCode:@"320000"
        merchantAccountCode:@"320001"
        originalReferenceNumber:nil
        connectionService:nil
        connectionQueue:queue
```

```
                cashBackAmount:nil
                PaymentAccountNumber:nil
                ExpirationDate:@""
                shippingAddress:nil
                DeviceType:@"icmp"
                testMode:YES
                withToken:@""];

    _transaction = [[PayGuardianTransaction alloc]
initWithPaymentRequest:_request];

    [_transaction runOnCompletion: ^(BPNPayment *payment, NSError *error) {
        //process response


    } onStateChanged: ^(PayGuardianTransactionState state) {
        //process state change


    }];
```

Invalid Tender Type Response

An example of a sale transaction response error where an unacceptable tender type was processed.

```
NSError:

Error Domain=com.bridgepaynetwork.PayGuardian
Code=2 "Invalid payment request"
UserInfo={NSLocalizedDescription=Invalid payment request}
```

Cancel Transaction Request

An example of how to submit a sale transaction to simulate the cancellation of a transaction after the transaction has been initiated.

```
    _request = [[BPNPaymentRequest alloc]
                initWithAmount:[NSDecimalNumber decimalNumberWithString:@"6.75"]
                tipAmount:nil
                invoiceNumber:@"7575"
                tenderType:TENDER_TYPE_CREDIT
                transactionType:TRANSACTION_TYPE_SALE
                username:@"emvpgtest"
                password:@"57!sE@3Fm"
                merchantCode:@"320000"
                merchantAccountCode:@"320001"
                originalReferenceNumber:nil
                connectionService:nil
                connectionQueue:queue
                cashBackAmount:nil
                PaymentAccountNumber:nil
                ExpirationDate:nil
                shippingAddress:nil
                DeviceType:nil
                testMode:YES
                withToken:nil];

    _transaction = [[PayGuardianTransaction alloc]
initWithPaymentRequest:_request];

    [_transaction runOnCompletion: ^(BPNPayment *payment, NSError *error) {
        //process response

    } onStateChanged: ^(PayGuardianTransactionState state) {
        //process state change

    }];
```

Cancel Transaction Error Response

An example of a cancelled sale transaction error response where the cancellation of a transaction occurred after the transaction was initiated.

```
Error Domain=com.bridgepaynetwork.PayGuardian
Code=6 "Card failed to read"
UserInfo={NSLocalizedDescription=Card failed to read}
```

3.2. EMV Transaction Support

3.2.1. EMV Implementation Details

PayGuardian also supports EMV transactions. Upon receiving the transaction request, PayGuardian calls the device to obtain the card information. The device will prompt to Swipe or Insert the Card. Upon the card insertion or card swipe, the transaction will be processed according to the user selection.

Minimum Requirement:

Card Reader Version: 15.0.4 RBA

When a transaction is being processed as an EMV transaction, the communication messages between the Card Reader device and the PayGuardian Application are uniquely identified by a “33.” message identifier. There are currently seven message types used during EMV transactions. The message type is selected using a subcommand identifier which is embedded in the message. Subcommand identifiers are used to identify the different message types:

1. **EMV Transaction Initiation Message**
   The '00.' EMV Transaction Initiation message is sent from the PayGuardian to the device to indicate the type of transaction purchase.

2. **EMV Track 2 Equivalent Data Message**
   The '02.' EMV Track 2 Equivalent Data message is sent from the device to the PayGuardian. This data is similar to the Track 2 data which is stored on a magnetic stripe card.

3. **EMV Authorization Request Message**
   The '03.' EMV Authorization Request message is sent from the device to the PayGuardian to provide the cryptographic information necessary to authorize the transaction. The authorization process is initiated by the device issuing a request to the PayGuardian.

4. **EMV Authorization Response Message**
   The '04.' EMV Authorization Response message is sent from the PayGuardian to the device in response to the EMV Authorization Request message. This message includes cryptographic information which is read by the embedded microchip on the card.

5. **EMV Authorization Confirmation Response Message**
   The '05.' EMV Confirmation Response message is sent from the device to the PayGuardian, and contains the results from applying the authorization data to the embedded microchip on the EMV card.

3.2.2. EMV Purchase Transaction Flow

This section describes the message flow for EMV transactions once PayGuardian receives the transaction request from the mPOS application.

- PayGuardian will activate the Device based on the terminal type specified in the transaction payment request.
- The card number, expiration date, and cardholder name will be transmitted in the transaction request through the Device (Ingenico terminals will prompt for amount confirmation).
- The transaction response will be returned to PayGuardian and to the Device. If the Device has a signature capture pad, the customer will be prompted to Sign and Accept on the Device.
- If the electronic signature is captured, it will be returned encoded in the response object.
- Remove the EMV card from the Device.

Sample EMV Sale Request:

An example of a basic EMV sale transaction request.

```
    _request = [[BPNPaymentRequest alloc]
            initWithAmount:[NSDecimalNumber decimalNumberWithString:@"6.00"]
            tipAmount:nil
            invoiceNumber:@"12345"
            tenderType:TENDER_TYPE_CREDIT
            transactionType:TRANSACTION_TYPE_SALE
            username:@"emvpgtest"
            password:@"57!sE@3Fm"
            merchantCode:@"320000"
            merchantAccountCode:@"320001"
            originalReferenceNumber:nil
            connectionService:nil
            connectionQueue:queue
            cashBackAmount:nil
            PaymentAccountNumber:nil
            ExpirationDate:@""
            shippingAddress:nil
            DeviceType:@"icmp"
            testMode:YES
            withToken:@""];

    _transaction = [[PayGuardianTransaction alloc]
  initWithPaymentRequest:_request];

    [_transaction runOnCompletion: ^(BPNPayment *payment, NSError *error) {
      //process response

      });


    } onStateChanged: ^(PayGuardianTransactionState state) {
      //process state change

      });

    }];
```

Sample EMV Sale Response:

An example of a basic EMV sale transaction response.

```
BPNPayment.BridgeCommResponse:
transactionID:    12345
requestType:     004
responseCode: 0
responseDescription:     Successful Request
token:                   11110000000077149269
expirationDate: 0325
authorizationCode:       471130
originalReferenceNumber:
authorizedAmount:        $6.00
originalAmount:           $6.00
gatewayTransactionID:  214971801
gatewayMessage:          A01 - Approved
internalMessage:          Approved: 471130 (approval code)
gatewayResult:            00000
AVSMessage:
AVSResponse:
CVMessage:
CVResult:
transactionCode:          (null)
transactionDate:          20161020
remainingAmount:          $0.00
ISOCountryCode:           (null)
ISOCurrencyCode:          USD
ISOTransactionDate:       2016-10-20 13:39:46.987
ISORequestDate:           2016-10-20 13:39:46.987
networkReferenceNumber:
merchantCategoryCode:
networkMerchantID:    518564010126944
networkTerminalID:     PPB01.
cardType:          Visa
maskedPAN:      4***********9269
responseTypeDescription:        sale
isCommercialCard:         False
streetMatchMessage:
secondsRemaining:         (null)
merchantCode: (null)
merchantAccountCode:              (null)
merchantName:             (null)
receiptTagData:            4F:A0000000031010;95:8080008000;9F10:06010A03A0000
0;9B:6800;91:;8A:
issuerTagData:
applicationIdentifier:     A0000000031010
terminalVerificationResults:       8080008000
issuerApplicationData:   06010A03A00000
transactionStatusInformation:   6800
```

Sample EMV Sale Receipt

An example of the values returned on a basic EMV Sale receipt.

```
BPNPayment.BPNReceipt:
maskedCardNumber:   4***********9269:
chipCardAID:     A0000000031010
invoice: 12345
seq:      12345
authorizationCode:       471130
entryMethod:  Chip_Read
totalAmount:    6
appLabel:          VISA DEBIT
cardHolderName:          CARDHOLDER/VALUED
networkMerchantId:    518564010126944
networkTerminalId:       PPB01.
cardFirstFour:   4204
cardType:          Visa
requiresSignature:        YES
pinEntered:       NO
```

A. Appendix

A.1. Response Values

A.1.1. Result Codes

The following table contains descriptions of the result codes returned in the **ResultCode** response field from PayGuardian. Please note that when programmatically validating the result of a transaction, you should use this value instead of any other response message describing the result.

Value	Description
0	Approved.
1	Cancelled. Review **ResultTxt** field in **PaymentResponse** to determine why the transaction was not processed.
2	Declined. Review **ResultTxt** field in **PaymentResponse** to determine why the transaction was not processed.

A.1.2. AVS Response Codes

The following table contains the possible descriptions of the response values returned for Address Verification (AVS) in the **AvsResponse** response field from PayGuardian.

Please Note: If the response returned is blank for this specific field tag, it is possible that your selected processor does not support the full range of AVS codes.

Value	Description
00	AVS Error – Retry, System unavailable, or Timed out
40	Address not available (Address not verified)
43	Street address not available (not verified), ZIP matches
44	Address failed
45	Street address and Zip don’t match
46	Street address doesn’t match, 5-digit ZIP matches
47	Street address doesn’t match, 9-digit ZIP matches

4A	Street address or ZIP doesn’t match
4D	Street address matches, ZIP does not
4E	Street address and 5-digit ZIP matches
4F	Street address and ZIP match
53	Account holder name incorrect, billing postal code matches
55	Unrecognized response – Account holder name, billing address, and postal code are all incorrect
5C	Account holder name incorrect, billing address matches
5F	Account holder name incorrect, billing address and postal code match
70	Account name matches
73	Account holder name and billing postal code match
7C	Account holder name and billing address match
7F	Account holder name, billing address and postal code match
80	AVS service not supported by issuer – Issuer doesn’t participate in AVS

A.1.3. CV Response Codes

The following table contains the possible descriptions of the response values returned for a CVV2/CVC2/CID check in the **CvResponse** response field from PayGuardian.

Please Note: If the response returned is blank for this specific field tag, it is possible that your selected processor does not support the full range of CVV response codes.

Value	Description
M	CVV2/CVC2/CID – Match.
N	CVV2/CVC2/CID – No Match.
P	Not Processed.
S	Issuer indicates that the CV data should be present on the card, but the merchant has indicated that the CV data is not present on the card.
U	Unknown. Issuer has not certified for CV or issuer has not provided Visa/MasterCard with the CV encryption keys.
X	Unrecognized reason. Server provided did not respond.

A.2. Errors

A.2.1. iOS App Extension Errors

Result Code	Description
E1001	Invalid invoice number
E1002	Invalid amount.
E1003	Invalid username.
E1004	Invalid password.
E1005	Invalid merchant code.
E1006	Invalid merchant account code.
E1007	Invalid tender type.
E1008	Invalid transaction type.
E1009	Invalid payment request
E1010	Invalid reference number.
E1011	No network connection.
E1012	Device not found.
E1013	Xml deserialization error.
E1014	Xml serialization error
E1015	Xml base64 encode error.
E1016	Xml base64 decode error.
E1017	Unable to process transaction
E1018	Unable to access device.
E1019	Transaction cancelled, Device TimedOut.
E1020	Transaction cancelled.

A.3. Test Cards and Data

A.3.1. Swiped / MSR Test Transaction Info

The following test card numbers are required for use in both integration testing and certification for PayGuardian iOS, but only for **‘NON-EMV TRANSACTIONS’** (a separate test account will be provided for EMV transaction testing):

Test Cards & Bank Accounts

The following accounts will be accepted by the test server's validation mechanism and thus can be used for preliminary testing:

Card Type	**Card Brand**	**Card Number**	**Expiration Date**	**Track Data**
Credit	Visa	4111111111111111	1017	%B4111111111111111^Smith/John^16041011000 1111A123456789012?
Credit	MasterCard	5499740000000057	1017	%B5499740000000057^Smith/John^16041011000 1111A123456789012?
Credit	Discover	6011000991001201	1017	%B6011000991001201^Smith/John^16041011000 1111A123456789012?
Credit	Amex	371449635392376	1017	%B371449635392376^Smith/John^16041011000 1111A123456789012?
Debit	Visa	4217651111111119	1017	%B4217651111111119^Smith/John^16041011234567 440?;4217651111111119=16041011234567440?
Debit	MasterCard	5149612222222229	1017	%B5149612222222229^Smith/John^16041011234567 440?;5149612222222229=16041011234567440?

Encrypted Data

Card Number	Expiration Date	Secure Format	MSRKSN	Track 1 Encrypted	Track 2 Encrypted
5526399000648568	0416	SecureMagV2	310601012B000B40007B	BD12FA1B13FB19BD8897CBCF4109205B69133A78F2B2181F2D3849B484303F4CEB54FD2857748B0D92E72E0BAFB9D9876C431FF24F3712866550DA1D697E84F353F3EEBB8AA42E4F	67E98B7924544F5803BC4A4453834BD13D1929CD7006360A551A1890383011EBBEDBB3442BD71F45
4012000033330026	0416	SecureMagV2	310601012B000B400080	AA06168B0C7DFBF22DFEC13D7B49242906D2639C65E4AAF5364B143ED5DCB9F45AA04BA73F631032B83648311E88E4ED921451F0E9684E866613F35BD74F53802C3AA9E56643863E988DD4BABD4753A5	D011E8820440DE01CD626AE55921ADA5F3D475EB9150E2959DFA1EE6353A3204EF7F69797B97D753
5473500000000014	1225	SecureMag	91070100000000800007	EA02EB27DAFC3BE1AE43C318D96F39E9AB90D55B0978614D7860C895721DD0E9EA358FDDCA4A0FAADE3D1E0E69D91FCB31B70E73B8FB5E49F4E4AF6219AB9B6F75CFFF8A6385DE6645426EC7E9DDD128C7D63E662B2C6E969E3CEE75789F66C48251B69A6B9C79CDC570FD984867ACA2	
4012002000060016	1225	SecureMag	91070100000000800008	C14C88046A89D221FA50EF0DE61B61C4C113095BB6FB2D6185BC5100ABB0ABF6DDA5F7D1F4F3DFA16DFB365EA098CC927195B4A3F9762BF85262D1530C88181B794E3CADBEA469539DEBBD341751599322B1A7A4D2C760044CCC2B311B4D175959FCE0E2DF9B0BB7	

Test amount ranges

As a part of the testing, the amount ranges specified below, can be used to trigger specific response codes from the server. Any valid account number and any properly formatted billing address can be used for the test.

Sales and Authorizations

Amount Range		**Credit Cards**
In dollars	**In cents**	**Response Message**
5.00 – 69.99	500 – 6999	Approved.
70.00 – 79.99	7000 – 7999	Invalid Card Number (Invalid Account Number)
80.00 – 89.99	8000 – 8999	Card reported lost/stolen (Lost/Stolen Card)
90.00 – 99.99	9000 – 9999	Call for Authorization (Referral)*
100.00 – 109.99	10000 – 10999	Hold – Pick up card (Pick Up Card)
110.00 – 119.99	11000 – 11999	CSC is invalid (Decline CSC/CID Fail)
120.00 – 129.99	12000 – 12999	Insufficient Funds
130.00 – 139.99	13000 – 13999	Processing Network Unavailable
140.00 – 149.99	14000 – 14999	Processing Network Error
150.00 – 159.99	15000 – 15999	Partially Approved**

*This range is designated to test voice authorization. Sale, Auth request in the amount between $90.00 – 99.99 will Response in a decline code, however if approval code 012345 is specified, then an approval will be received.

**This range is designated to test partial authorizations. By setting Sale with this amount range the partially approved transaction will be received. Approved amount will be $10 less than the originally requested amount.

Credits

Amount Range (in dollars)	**Amount Range (in cents)**	**Response Message**
5.00 – 69.99	500 - 6999	Credit Posted

A.4. Terms of Use Agreement

1. **ACKNOWLEDGMENT AND ACCEPTANCE OF AGREEMENT**

The Terms of Use Agreement “TOU” is provided by BridgePay to you as an end user “USER” of the information obtained from BridgePay, any amendments thereto, and any operating rules or policies that may be published from time to time by BridgePay, all of which are hereby incorporated by reference. The TOU comprises the entire agreement between USER and BridgePay and supersedes any prior agreements pertaining to the subject matter contained herein.

2. **DESCRIPTION OF SPECIFICATIONS AND INFORMATION**

BridgePay is providing USER with the information concerning the technical requirements or allowing point of sale software to send and receive electronic transaction data to the BridgePay network for authorization and/or settlement. To utilize the Specifications, USER must: (i) provide for USER's own access to the World Wide Web and pay any fees associated with such access, and (ii) provide all equipment necessary for USER to make such connection to the World Wide Web, including a computer, modem and Web browser.

3. **USER'S REGISTRATION OBLIGATIONS**

In consideration of use of the Specifications, USER agrees to: (i) provide true, accurate, current, and complete information about USER as requested on the Registration Form, and (ii) to maintain and update this information to keep it true, accurate, current and complete. This information about a USER shall be referred to as "Registration Data". If any information provided by USER is untrue, inaccurate, not current, or incomplete, BridgePay has the right to terminate USER's access to the Specifications and refuse any and all current or future use of the Specifications.

4. **MODIFICATIONS TO AGREEMENT**

BridgePay may change the TOU from time to time at its sole discretion. Changes to the TOU will be announced and publicly available to all USERs.

5. MODIFICATIONS TO SPECIFICATIONS

BridgePay reserves the right to modify or discontinue, temporarily or permanently, the use of any of the Specifications with or without notice to USER. USER agrees that BridgePay shall not be liable to USER or any third party for any modification or discontinuance of a Specification.

6. USER ACCOUNT, PASSWORD AND SECURITY

USER will receive a password when registering their company (account) to become a Partner. Upon approval, that password will allow USER access into the Partner Portal. USER is responsible for maintaining the confidentiality of the password and account, and is fully responsible for all activities that occur under USER's password or account. USER agrees to immediately notify BridgePay of any unauthorized use of USER's password or account or any other breach of security.

7. LICENSE GRANT

a. Subject to the terms and conditions of this Agreement, BridgePay hereby grants to USER a personal, limited, perpetual, non-exclusive, non-transferable, annual subscription license to make and use the SOFTWARE accompanying this TOU to be installed on CPUs residing on Licensee's premises, solely for Licensee's internal use. BridgePay and its suppliers shall retain title and all ownership rights to the product and this Agreement shall not be construed in any manner as transferring any rights of ownership or license to the SOFTWARE or to the features or information therein, except as specifically stated herein. use and reproduce the following solely to develop, manufacture, test and support the products: (i) in object code form (except as may be agreed by the parties in writing or as otherwise set forth in this Agreement), (ii) the applicable software in object code form, except as may be agreed by the parties in writing or as otherwise set forth in this Agreement, (iii) BridgePay materials which shall include all documentation.

b. Upon the annual anniversary date of the activation of the license, the USER will be responsible the renewal of the subscription of the license either through their RESELLER or directly to BridgePay.

8. TRADEMARKS
USER acknowledges that BridgePay owns exclusive rights in the BridgePay trademarks. USER will not use BridgePay as part of any of its product, service, domain or company names and will not take nor authorize any action inconsistent with BridgePay's' exclusive trademark rights during the term of this Agreement or thereafter. Nothing in this Agreement grants USER ownership or any rights in or to use the BridgePay trademarks except in accordance with this license. USER will use a legend on its website and, where commercially feasible, on all printed materials and products bearing the BridgePay trademarks similar to the following: "(USER name) uses the BridgePay™ mark under express license from BridgePay, LLC."

9. USER OBLIGATIONS

a. USER shall utilize its BridgePay assigned developer ID in each application utilizing the BridgePay specification

b. USER shall not reverse-engineer, reverse-compile or disassemble any BridgePay software or otherwise attempt to derive the source code to any BridgePay software.

c. USER shall have no right to (i) disclose any BridgePay source code or BridgePay source code or BridgePay source code documentation other than as permitted or contemplated by this Agreement. No licenses are granted by BridgePay to USER by implication or estoppels to the BridgePay source code or BridgePay source code documentation.

d. USER shall comply with all applicable card association regulations, applicable federal, state and local statutes and BridgePay required procedures and identified best practices. USER agrees (i) not to use the Specifications for illegal purposes; and (ii) to comply with all applicable laws regarding the transmission of technical data exported from the United States.

10. DISCLAIMER OF WARRANTIES

USER expressly agrees that use of the Specifications is at USER's sole risk. The Specifications are provided on an "as is" basis.

a. BRIDGEPAY EXPRESSLY DISCLAIMS ALL WARRANTIES OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT

b. BRIDGEPAY MAKES NO WARRANTY THAT THE SPECIFICATION WILL MEET USER'S REQUIREMENTS, NOR DOES BRIDGEPAY MAKE ANY WARRANTY AS TO THE RESULTS THAT MAY BE OBTAINED FROM THE USE OF THE SPECIFICATIONS OR AS TO THE ACCURACY OR RELIABILITY OF ANY INFORMATION OBTAINED THROUGH USE OF THE SPECIFICATIONS. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF CERTAIN WARRANTIES, SO SOME OF THE ABOVE EXCLUSIONS MAY NOT APPLY TO YOU.

11. TERMINATION BY BridgePay

USER agrees that BridgePay may terminate USER’s password, account or use of the Specifications:

a. If BridgePay determines in its sole discretion that USER has violated or acted inconsistently with the letter or spirit of the TOU;

b. If the USER has violated the rights of BridgePay, or that USER's continued use of the Specifications poses a material threat to the security, stability or ongoing operation of the System or Specifications.

c. BridgePay may terminate this Agreement for cause at any time upon providing not less than ten (10) business day’s prior written notice to USER. USER acknowledges and agrees that any termination of access privileges to the Specifications under any provision of the Agreement may be effected without prior notice.

d. BridgePay shall in the event that a license has not been renewed and BridgePay has not received and validated payment from either the USER or the RESELLER within 30 days of the anniversary date of the original activation the deactivate or terminate the usage of the license.

12. LIMITATION OF LIABILITY

In no event shall BridgePay be liable to USER for any incidental, consequential or punitive damages related to this Agreement or the use of BridgePay software or specifications. The liability of BridgePay hereunder shall be limited to the fees paid to BridgePay pursuant to this Agreement. USER agrees that BridgePay shall not be liable for any direct, indirect, incidental, special, or consequential damages, resulting from the use or the inability to use the Specifications, including but not limited to, damages for loss of profits, use, data or other intangibles, even if BridgePay has been advised of the possibility of such damages. Some jurisdictions do not allow the limitation or exclusion of liability for incidental or consequential damages so some of the above limitations may not apply to you.
NOTICE: Any notice to USER or to BridgePay shall be made via either e-mail or regular mail. BridgePay may also provide notices of changes to the TOU or other matters by displaying notices to USERs, generally on the Specifications.

13. SPECIAL DAMAGES
In no event will BridgePay be liable to the USER, consequential or punitive damages, including but not limited to, lost profits, even if such party knew of the possibility of such damages.

14. INDEMNIFICATION

a. USER shall be liable to and shall indemnify and hold BridgePay, its employees, representatives, successors and permitted assigns harmless from and against any and all claims, demands by third parties, losses, liability, cost, damage and expense, including litigation expenses and reasonable attorneys' fees and allocated costs for in house legal services, to which BridgePay, its employees, representatives, successors and permitted assigns may be subjected or which it may incur in connection with any claims which arise from or out of or as the result of (i) USER's breach of this Agreement, (ii) the performance by USER of its duties and obligations under this Agreement or (iii) the negligent or willful misconduct of USER, its officers, employees, agents and affiliates in the performance of their duties and obligations under this Agreement.

15. PROTECTION OF CONFIDENTIAL INFORMATION
All information of a business nature relating to the BridgePay specification, software, application interfaces, services, processes, merchant and cardholder data, product or programming techniques of either party shall be deemed confidential ("Confidential Information"). This shall not prohibit each party from disclosing such Confidential Information to persons required to have access thereto for the performance of this Agreement; provided, however, that such persons shall be required to keep such Confidential Information confidential to the same standard that the disclosing party is obligated to keep the Confidential Information confidential.

16. FORCE MAJEURE
In no event shall BridgePay be liable with respect to the failure of its duties and obligations under this Agreement (other than an obligation to pay money) which is attributable to acts of God, war, terrorism, conditions or events of nature, civil disturbances, work stoppages, equipment failures, power failures, fire or other similar events beyond its control.

17. GENERAL

a. The Specifications Agreement and the relationship between USER and BridgePay shall be governed by the laws of the State of Illinois without regard to its conflict of law provisions.

b. The failure of BridgePay to exercise or enforce any right or provision of the TOU shall not constitute a waiver of such right or provision. If any provision of the TOU is found by a court of competent jurisdiction to be invalid, the parties nevertheless agree that the court should endeavor to give effect to the parties' intentions as reflected in the provision, and the other provisions of the TOU remain in full force and effect.

c. USER agrees that regardless of any statute or law to the contrary, any claim or cause of action arising out of or related to use of the Specifications or the Specifications

Agreement must be filed within ninety (90) days after such claim or cause of action arose or be forever barred.

18. SECTION TITLES
The section titles in the TOU are for convenience only and have no legal or contractual effect.