# MobyPay POS - Kiosk Integration Documentation

## 1. Introduction and Behavior

The MobyPay Kiosk system enables seamless integration between kiosk applications and Point of Sale (POS) terminals through TCP/IP communication. The system follows a client-server architecture where the kiosk acts as a TCP server, and the POS terminal connects as a client.

### Key Components

- Kiosk Server: TCP server that handles payment requests
- POS Terminal: Flutter-based mobile application that processes payments
- Security Layer: HMAC-SHA256 message signing with timestamp and nonce validation
- Payment Processing: Supports 4 payment modes with real-time status updates

### System Behavior

- Kiosk starts TCP server and waits for POS connections
- POS terminal connects to kiosk via IP address (manual entry or QR scan)
- Secure bidirectional communication using encrypted message protocol
- Real-time payment processing with acknowledgments and status updates
- Automatic session management

### Connection State Behaviors

#### *When Disconnected*

- POS terminal displays connection interface with QR scanner and manual IP entry tabs
- All kiosk payment functions are disabled
- User can attempt connection via QR code scanning or manual IP input

#### *When Connected*

- Touch interaction is completely disabled on the POS terminal to prevent interference
- POS terminal displays "Connected to KIOSK" status with green indicator
- All payment flows are initiated exclusively from the kiosk server
- POS terminal enters passive listening mode for incoming payment requests
- Navigation is controlled entirely by the kiosk system
- Automatic timeout and cleanup for incomplete transactions
- Connection status monitoring with automatic reconnection attempts

***During Payment Processing***

- POS terminal is locked to the specific payment flow initiated by kiosk
- Cancel operations can only be initiated from the kiosk side
- UI displays payment-specific interfaces (QR codes, card insertion prompts, etc.)
- Real-time status updates sent back to kiosk
- Automatic return to connected state upon completion or cancellation

## 2. Connection to Kiosk

### Connection Flow

```
┌───────────────────┐   TCP Connection   ┌───────────────────┐
│  Kiosk            │◄──────────────────►│ POS Terminal.     │
│ (Server)          │   Port 8080        │  (Client)         │
│                   │                    │                   │
│ Kiosk App         │ ◄── Messages  ──►  │Terminal App       │
└───────────────────┘                    └───────────────────┘
```

### Connection Parameters

| Parameter | Type | Description | Default Value |
|---|---|---|---|
| host | String | Kiosk server IP address | 0.0.0.0 (all interfaces) |
| port | Integer | TCP port number | 8080 |

### Connection Process

1. Kiosk Server Startup:

```
python
# Sample implementation (any language can be used)
sender = TcpSender(port=8080)
sender.start_server()  # Binds to 0.0.0.0:8080
```

2. POS Terminal Connection:

Tap KIOSK button on Home screen of the Terminal app → Enter IP Address that displayed on kiosk app / Scan IP address QR code → Connect

3. Connection Verification:

- Server displays connected client IP and port
- POS terminal receives connection confirmation
- Both sides establish message listeners

**Connection States**

| State | Description | Actions Available |
|---|---|---|
| Disconnected | No active connection | Start connection |
| Connecting | Establishing connection | Cancel connection |
| Connected | Active TCP connection | Send payments, Disconnect |
| Error | Connection failed | Retry connection |

## 3. Payment Flows

### 3.1 Card Payment Flow

Basic Flow:

Kiosk → POS: transaction_request (card)
POS → Kiosk: ack (processing)
POS → Kiosk: transaction_result (success/failed)

**Sample Request:**

```
{"payload": {"type": "transaction_request", "txn_id": "TXN1757492341554", "amount":
10.0, "payment_mode": "card", "kiosk_id": "KIOSK001", "timestamp": "1757492341555",
"nonce": "8915fea5-ad40-4c7c-b0a9-86665c66013e"}, "signature":
"7ee758c7a3182d1b141d998e4a5f92d85141de1537ff51bde0302ccc9927f8e8"}
```

Parameters:

| Parameter | Type | Required | Description |
|---|---|---|---|
| type | String | Yes | "transaction_request" |
| txn_id | String | Yes | Unique transaction ID |
| amount | Double | Yes | Payment amount in RM |
| payment_mode | String | Yes | "card" |
| kiosk_id | String | Yes | Kiosk identifier |
| timestamp | String | Yes | Unix timestamp (ms) |
| nonce | String | Yes | Unique request identifier |
| signature | String | Yes | See 4.1 |

**Sample ACK:**

```
{"payload":{"type":"ack","txn_id":"TXN1757482355464","status":"processing","pos_id":"POS001","timestamp":"1757482354771","nonce":"0voFl1q7/v7ZNT0HO2qSpw=="},"signature":"f187f605b98cfde8f95a62eae2eefe4a79fcccd943245b77101912a58dd40f28"}
```

**Sample Response (Success):**

```
{"payload":{"type":"transaction_result","txn_id":"TXN1757482355464","status":"success","pos_id":"POS001","transaction_id":"000410","amount":2.0,"timestamp":"1757482370549","nonce":"joHQ83Mp9dWbFsREUHDRCg=="},"signature":"fc8d4da75e3930f296d2dae8edaac62ec9a5059a5ea912cebcf8a9433852f761"}
```

**Sample Response (Failure):**

```
{"payload":{"type":"transaction_result","txn_id":"TXN1757482545764","status":"failed","pos_id":"POS001","error_message":"USER_ABORT","amount":80.0,"timestamp":"1757482553481","nonce":"MxMVKtS1wf27M6r1Gpeayw=="},"signature":"0b87d8ad164afdd739a3d519cdd70f478543ff1a15f8dd4f90d6f04914fb2676"}
```

### 3.2 Buy Now Pay Later (BNPL) Flow

Basic Flow:

Kiosk → POS: transaction_request (bnpl)
POS → Kiosk: ack (processing)
[Customer scans QR and completes payment]
POS → Kiosk: transaction_result (success/failed)

Parameters:

| Parameter | Type | Required | Description |
|---|---|---|---|
| type | String | Yes | "transaction_request" |
| txn_id | String | Yes | Unique transaction ID |
| amount | Double | Yes | Payment amount in RM |
| payment_mode | String | Yes | "bnpl" |
| kiosk_id | String | Yes | Kiosk identifier |
| timestamp | String | Yes | Unix timestamp (ms) |
| nonce | String | Yes | Unique request identifier |
| signature | String | Yes | See 4.1 |

### 3.3 DuitNow QR Flow

Basic Flow:

Kiosk → POS: transaction_request (duitnow_qr)
POS → Kiosk: ack (processing)
[Customer scans and pays]
POS → Kiosk: transaction_result (success/failed)

Parameters:

| Parameter | Type | Required | Description |
|---|---|---|---|
| type | String | Yes | "transaction_request" |
| txn_id | String | Yes | Unique transaction ID |
| amount | Double | Yes | Payment amount in RM |
| payment_mode | String | Yes | "duitnow_qr" |
| kiosk_id | String | Yes | Kiosk identifier |
| timestamp | String | Yes | Unix timestamp (ms) |
| nonce | String | Yes | Unique request identifier |
| signature | String | Yes | See 4.1 |

### 3.4 Installment Payment Plan (IPP) Flow

Basic Flow:

Kiosk → POS: transaction_request (ipp)
POS → Kiosk: ack (processing)
POS → Kiosk: transaction_result (ipp_plans)
Kiosk → POS: ipp_plan_selection
POS → Kiosk: ack (plan_received)
POS → Kiosk: transaction_result (success/failed)

Parameters:

| Parameter | Type | Required | Description |
|---|---|---|---|
| type | String | Yes | "transaction_request" |
| txn_id | String | Yes | Unique transaction ID |
| amount | Double | Yes | Payment amount in RM |
| payment_mode | String | Yes | "ipp" |

| | | | |
|---|---|---|---|
| kiosk_id | String | Yes | Kiosk identifier |
| timestamp | String | Yes | Unix timestamp (ms) |
| nonce | String | Yes | Unique request identifier |
| signature | String | Yes | See 4.1 |

**Sample Response (Plans Stage):**

```
{"payload":{"type":"transaction_result","txn_id":"TXN1757492757556","status":"ipp_plans","pos_id":"POS001","plans":[{"planId":"pay-in-full","frequency":"","totalInstallments":0,"installmentDetails":[]}],"amount":100.0,"timestamp":"1757492763490","nonce":"K6LQxCh2CAE2NB8Do3HOrw=="},"signature":"40396385365ebbeea4e750afd71176581d910005e4076d25c70f1ae1e2cd1923"}
```

**Sample Response (ACK – Plans Received)**

```
{"payload":{"type":"ack","txn_id":"TXN1757492757556","status":"plan_received","pos_id":"POS001","timestamp":"1757492832393","nonce":"3KKnbVL/huAu+TmqQ2FMqA=="},"signature":"0cff7d6130760b5b4a16aa697059cfe0797c8996ae706871fda703035909b8d2"}
```

## 4. Security

The MobyPay Kiosk system implements multiple security layers to ensure secure communication and prevent fraud:

### 4.1 Message Security Layer

HMAC-SHA256 Signature:

- All messages signed with shared secret: POS-KIOSK-SECRET-KEY-2024
- Payload and signature transmitted separately
- Signature verification on both ends

```python
python
# Sample implementation (adaptable to any programming language)
def generate_signature(payload):
    json_string = json.dumps(payload, sort_keys=True, separators=(',', ':'))
    signature = hmac.new(
        SHARED_SECRET.encode('utf-8'),
        json_string.encode('utf-8'),
        hashlib.sha256
    ).hexdigest()
    return signature
```

### 4.2 Replay Attack Prevention

Nonce Validation:

- Unique nonce generated for each message
- Server maintains list of used nonces
- Duplicate nonces rejected automatically

```
python
# Sample nonce validation (adaptable to any language)
def validate_nonce(nonce):
  if nonce in used_nonces:
    return False
  used_nonces.add(nonce)
  return True
```

### 4.3 Timestamp Validation

Time Window Security:

- Messages valid within 60-second window
- Prevents replay of old messages
- Synchronized time validation

```
python
# Sample timestamp validation (adaptable to any language)
def validate_timestamp(timestamp):
  request_time = int(timestamp)
  current_time = int(datetime.now().timestamp() * 1000)
  difference = abs(current_time - request_time)
  return difference < 60000  # 60 seconds
```