# QPAY EZ-CONNECT INTEGRATION GUIDE

# Contents


**1 Introduction ..... 4**

1.1 Document Audience ..... 4

1.2 Purpose ..... 4

1.3 Scope ..... 4

1.4 Prerequisites ..... 4

**2 Overview ..... 5**

2.1 Secret Key ..... 5

Data Integrity ..... 5

Authentication ..... 5

Setting Secret Key ..... 5

2.2 Communication Model ..... 5

2.3 Transaction flow ..... 6

Request flow ..... 6

Response flow ..... 6

**3 QPay messages ..... 7**

3.1 Payment ..... 7

3.1.1 Request basic fields ..... 7

3.1.2 Response basic fields ..... 9

3.2 Inquiry ..... 10

3.2.1 Request basic fields ..... 10

3.2.2 Response basic fields ..... 10

3.3 Refund ..... 13

3.3.1 Request basic fields ..... 13

3.3.2 Response basic fields ..... 14

**4 Appendix ..... 16**

4.1 Appendix A: Date Format ..... 16

4.2 Appendix B: Response Codes ..... 16

4.2.1 Payment response codes ..... 16

4.2.2 Inquiry response codes ..... 19

4.3 Appendix C: Merchant Admin - Secret Key Configuration ..... 20

4.4 Appendix D: Secure Hash ..... 21

4.4.1 Payment ..... 21

4.4.2 Inquiry ..... 22

4.4.3 Refund ..... 24

4.5 Appendix E: Sample Code ..... 26
4.5.1 Using PHP ..... 26
4.5.2 Using Java ..... 31
4.5.3 Using C# ..... 37
4.6 Appendix F: Technical Data and Endpoints for QPay ..... 46

# 1 Introduction

## 1.1 Document Audience

This document is intended for merchant system designers & developers planning to integrate with EZ-Connect interface to accept e-commerce through PayOne Payment Gateway. Readers of this document should have the technical/developer knowledge required for web development.

## 1.2 Purpose

This guide is written for merchants who have enrolled, through acquirer banks, in QCB's QPay Payment Gateway system and wish to rely on the EZ-Connect Interface as their integration point. This interface will allow for the handling of electronic transactions (payment, refund, Inquiry) using Qatar debit cards, by using the HTTPS Post as programming interface to perform the transactions. It describes the format for sending transactions and the corresponding received responses.

## 1.3 Scope

This guide describes the EZ-Connect programming interfaces needed to integrate your website along with guidelines and best practices for presenting these payment offerings.

## 1.4 Prerequisites

- Merchant is contracted as an e-commerce QPay merchant
- Merchant is provided with access to the QPay merchant portal to generate the secret key that is required for integration.

# 2 Overview

Merchants will interact, through secure channels, with QCB's QPay system to perform electronic transactions. Merchant websites will construct requests, that consist of pre-defined fields & appropriate values, send these requests, and receive response messages from the PG system.

## 2.1 Secret Key

### Data Integrity

To guarantee request/response data integrity a Secure Hash will be generated for each message. This Secure Hash will be created using all message fields.

Upon receiving a request/response message the receiver will recreate the Secure Hash from received message parameters and compare it with Secure Hash already sent within the message.

### Authentication

For each Merchant a Secret Key will be generated/set, this Merchant Secret Key is used as part of the Secure Hash value, Secret Key will only be known to Merchant application and Payment Gateway, and accordingly message receiver (Merchant or PG) will include Secret Key value when generating the Secure Hash value.

### Setting Secret Key

After accessing the QPay merchant portal, the merchant user can navigate to the Secret Key function within to view/regenerate the Secret Key. This key is securely stored within QPay system & merchant can copy & store this value to be used later in his/her application to be used during communication with the payment gateway. This value is only known to the merchant as it is encrypted from the payment gateway's end.

Merchant's admin/developer must make sure to:

- ✓ Store the secret key as a secure database or file
- ✓ Change the secret key periodically according to the merchant's own security policies
- ✓ Avoid storing the secret key within the source code of any ASP, JSP or other web pages that might have any chance of being accessed or viewed over the web

## 2.2 Communication Model

Interacting with PG system can be done using two types of communication models based on message type. These models are:

- **Redirection communication model:**

  Where the merchant site issues a Http redirection command to the cardholder's browser, who will be redirected to payment gateway's payment site. The customer is then requested to provide some input to complete the cycle. After concluding the payment, the payment gateway will redirect the customer back to the merchant's site based on a predefined merchant URL

- **Back-to-Back communication model:**

  Where the merchant site collects all the needed information from the cardholder & initiates a Http POST call to payment gateway (without using the customer's browser). The payment gateway system would then process the request & provides a response to the merchant. The merchant site would then fetch the needed parameters from the response & inform the customer of the transaction's result. This flow would be transparent for the customer who wouldn't have noticed any communication between the merchant site & payment gateway system .

## 2.3 Transaction flow

For a successful integration, merchant should handle the below:

1. Payment messages
2. Inquiry messages
3. Refund messages

The request & response flows for those messages operate as:

### Request flow

- Merchant site would prepare the request message with the appropriate fields for the message type (payment, inquiry or refund)
- Merchant would generate the secure hash using the request parameters & the set secret key
- The request & secure hash are sent to payment gateway
- Upon receiving the request, payment gateway would retrieve the merchant's secret key
- Payment gateway would regenerate the secure hash using received request parameters & retrieved secret key
- Generated & received secure hashes are the compared. Request would be rejected by the payment gateway in case of mismatch, otherwise the request will continue in the process

### Response flow

- PG System prepares response for merchant request
- PG System generates Secret Hash using response parameters and Merchant Secret Key stored at PG side
- PG sends response and Secure Hash to Merchant
- Merchant upon receiving the response will use response parameter and Secret Key value stored at Merchant application to regenerate Secure Hash for response
- Merchant compares generated Secure Hash with received Secure Hash, if values mismatch. Response will be rejected. Else Merchant application will continue processing response

# 3 QPay messages

## 3.1 Payment

This message is intended to perform a payment using QPay EZ-Connect interface, based on the Redirection communication model.

### 3.1.1 Request basic fields

| Field Name | Description | Mandatory/ Optional | Type | Length | Sample |
|---|---|---|---|---|---|
| Action | Alphanumeric value representing the action to be called; 0 | M | Alphanumeric | 2 | 0 |
| BankID | Provided by acquirer bank's operations team upon merchant enrollment | M | Alphanumeric | 10 | |
| MerchantID | Provided by acquirer bank's operations team upon merchant enrollment | M | Alphanumeric | 10 | |
| CurrencyCode | Numeric ISO Code for currency<br>634 for QAR | M | Numeric | 3 | 634 |
| Amount | Transaction amount, ISO formatted | M | Numeric | 9 | 1050 for 10.5 |
| PUN | Payment Unique Number:<br>- Generated by the merchant<br>-Unique identifier for the transactions<br>-Should not include special characters or spaces | M | Alphanumeric | 20 | |
| PaymentDescription | Narrative description of the payment order | O | Alphanumeric | 255 | |
| MerchantModuleSessionID | Merchant web session id initiating the payment request | O | Alphanumeric | 256 | |
| TransactionRequestDate | ddMMyyyyHHmmss | M | Alphanumeric | 14 | |
| Quantity | Quantity of purchased item<br>Greater than 0 | M | Numeric | 6 | |
| ExtraFields_f14 | Set response page URL that will receive the response from PG.<br>Will allow merchant setting different response pages based on the requested service or any other criteria | M | Alphanumeric | 512 | |
| Lang | Language for the payment description parameter. Supported values are:<br>En & Ar. Default is En. | O | Alphanumeric | 2 | En |

| | | | | | |
|---|---|---|---|---|---|
| NationalID | National id of the customer performing the transaction.<br>If this field is not required to be filled by the PG, it can be set to Empty String:“”.<br>Should not be sent unless merchant business requires this field | O | Alphanumeric | 32 | |
| SecureHash | Generated hex-encoded hash using hashing algorithm SHA-2 (256), by concatenating parameters as a single string starting with the merchant’s secret key, then all parameters | M | Alphanumeric | 64 | |

### 3.1.2 Response basic fields

| Field Name | Description | Mandatory/ Optional | Type | Length | Sample |
|---|---|---|---|---|---|
| AcquirerID | Acquirer ID for | M | Alphanumeric | 10 | |
| Status | Response code.<br>Covers both errors generated by EZ-Connect interface & the payment gateway | M | Alphanumeric | 30 | |
| StatusMessage | Description of received status<br>Will be filled only after a complete execution process. | M | Alphanumeric | 512 | |
| ConfirmationID | Payment gateway confirmation reference | M | Alphanumeric | 27 | |
| MerchantID | Provided by acquirer bank's operations team upon merchant enrollment | M | Numeric | 10 | |
| EZConnectResponseDate | EZ-Connect response date to be used in regenerating the secure hash<br>ddMMyyyyHHmmss | M | Alphanumeric | 14 | |
| Amount | Transaction amount, ISO formatted | M | Numeric | 9 | 1050 for 10.5 |
| CurrencyCode | Numeric ISO Code for currency<br>634 for QAR | M | Numeric | 3 | 634 |
| MerchantModuleSessionID | Merchant web session id initiating the payment request | M | Alphanumeric | 256 | |
| PUN | Payment Unique Number:<br>- Generated by the merchant<br>-Unique identifier for the transactions<br>-Should not include special characters or spaces | M | Alphanumeric | 20 | |
| BankID | Provided by acquirer bank's operations team upon merchant enrollment | M | Alphanumeric | 10 | |
| CardExpiryDate | Card expiry date (yymm) | M | Alphanumeric | 4 | 2308 |
| CardHolderName | Cardholder name | M | Alphanumeric | 32 | |
| CardNumber | Masked card number | M | Alphanumeric | 19 | |
| Lang | Language for the payment description parameter. Supported values are:<br>En & Ar | O | Alphanumeric | 2 | En |
| SecureHash | Generated hex-encoded hash using hashing algorithm SHA-2 (256), by concatenating parameters as a single string starting with the merchant's secret key, then all parameters | M | Alphanumeric | 64 | |

## 3.2 Inquiry

This message is intended to perform an inquiry on the status of a payment using QPay EZ-Connect interface, based on the back-to-back communication model.

This can be used whenever a transaction faces an interruption or an abnormal execution, to retrieve its status.

### 3.2.1 Request basic fields

| Field Name | Description | Mandatory/ Optional | Type | Length | Sample |
|---|---|---|---|---|---|
| Action | Alphanumeric value representing the action to be called; 14 | M | Alphanumeric | 2 | 14 |
| MerchantID | Provided by acquirer bank's operations team upon merchant enrollment | M | Alphanumeric | 10 | |
| BankID | Provided by acquirer bank's operations team upon merchant enrollment | M | Alphanumeric | 10 | |
| OriginalPUN | Payment Unique Number:<br>- Generated by the merchant for the original tranction | M | Alphanumeric | 20 | |
| Lang | Language for the payment description parameter. Supported values are:<br>En & Ar | O | Alphanumeric | 2 | En |
| SecureHash | Generated hex-encoded hash using hashing algorithm SHA-2 (256), by concatenating parameters as a single string starting with the merchant's secret key, then all parameters | M | Alphanumeric | 64 | |

### 3.2.2 Response basic fields

| Field Name | Description | Mandatory/ Optional | Type | Length | Sample |
|---|---|---|---|---|---|
| Status | Response code.<br>Covers both errors generated by EZ-Connect interface & the payment gateway | M | Alphanumeric | 30 | |
| StatusMessage | Description of received status<br>Will be filled only after a complete execution process. | M | Alphanumeric | 512 | |
| OriginalStatus | Representing the response code of the original/inquired transaction | M | Alphanumeric | 10 | |
| OriginalStatusMessage | Description of received status of the original/inquired transactions | M | Alphanumeric | 512 | |

| | | | | | |
|---|---|---|---|---|---|
| | Will be filled only after a complete execution process. | | | | |
| OriginalConfirmationID | Payment gateway confirmation reference for the original/inquired transaction | M | Alphanumeric | 27 | |
| OriginalPUN | Payment Unique Number of original/inquired transaction | M | Alphanumeric | 20 | |
| OriginalExtractStatus | The extract status of the transaction.<br>Extract status to confirm = 0<br>Extract status to extract = 1<br>Extract status not to extract = 2<br>Extract status acknowledged = 3<br>Extract status cutoff = 4 | M | Numeric | 3 | |
| OriginalReversalStatus | Reversal status of the transaction<br>Requires reversal = 0<br>Reversal not required = 1<br>Reversed = 2 | M | Numeric | 3 | |
| TransactionResponseDate | Transaction response date<br>ddMMyyyyHHmmss | M | Alphanumeric | 14 | |
| Amount | Transaction amount, ISO formatted | M | Numeric | 9 | 1050 for 10.5 |
| CurrencyCode | Numeric ISO Code for currency<br>634 for QAR | M | Numeric | 3 | 634 |
| MerchantID | Provided by acquirer bank's operations team upon merchant enrollment | M | Numeric | 10 | |
| EZConnectResponseDate | EZ-Connect response date<br>ddMMyyyyHHmmss | M | Alphanumeric | 14 | |
| BankID | Provided by acquirer bank's operations team upon merchant enrollment | M | Alphanumeric | 10 | |
| SecureHash | Generated hex-encoded hash using hashing algorithm SHA-2 (256), by concatenating parameters as a single string starting with the merchant's secret key, then all parameters | M | Alphanumeric | 64 | |
| Quantity | Quantity of purchased item<br>Greater than 0 | M | Numeric | 6 | |
| ApprovalCode | Code received from payment processor, example VISA | O | Alphanumeric | 12 | |
| CardExpiryDate | Card expiry date (yymm) | O | Alphanumeric | 4 | 2308 |
| CardHolderName | Cardholder name | O | Alphanumeric | 32 | |
| CardNumber | Masked card number | O | Alphanumeric | 19 | |
| PaymentDescription | Narrative description of the payment order using the language specified in the language parameter. | O | Alphanumeric | 255 | |

| | | | | | |
|---|---|---|---|---|---|
| ItemID | ID of purchased item | O | Alphanumeric | 25 | |
| NationalID | National id of the customer performing the transaction.<br>If this field is not required to be filled by the PG, it can be set to Empty String:“”.<br>Should not be sent unless merchant business requires this field | O | Alphanumeric | 32 | |

## 3.3 Refund

This message is intended to perform a refund transaction using QPay EZ-Connect interface, based on the back-to-back communication model.

### 3.3.1 Request basic fields

| Field Name | Description | Mandatory/ Optional | Type | Length | Sample |
|---|---|---|---|---|---|
| Action | Alphanumeric value representing the action to be called; 6 | M | Alphanumeric | 2 | 6 |
| BankID | Provided by acquirer bank's operations team upon merchant enrollment | M | Alphanumeric | 10 | |
| MerchantID | Provided by acquirer bank's operations team upon merchant enrollment | M | Alphanumeric | 10 | |
| Lang | Language for the payment description parameter. Supported values are:<br>En & Ar | M | Alphanumeric | 2 | En |
| TransactionRequestDate | Transaction date<br>ddMMyyyyHHmmss | M | Alphanumeric | 14 | |
| CurrencyCode | Numeric ISO Code for currency<br>634 for QAR | M | Numeric | 3 | 634 |
| PUN_* | Payment Unique Number of the current refund transaction, generated by merchant | M | Alphanumeric | 20 | |
| OriginalTransactionPaymentUniqueNumber_* | Payment Unique Number of the transaction to be refunded, generated by merchant | M | Alphanumeric | 20 | |
| Amount_* | Transaction amount, ISO formatted | M | Numeric | 9 | 1050 for 10.5 |
| SecureHash | Generated hex-encoded hash using hashing algorithm SHA-2 (256), by concatenating parameters as a single string starting with the merchant's secret key, then all parameters | M | Alphanumeric | 64 | |

***The Refund request can handle multiple transactions in one request, up to a maximum of 4.***

***For each transaction to be refunded, the respective field names marked with a *, should be concatenated with the same number.***

### 3.3.2 Response basic fields

| Field Name | Description | Mandatory/ Optional | Type | Length | Sample |
|---|---|---|---|---|---|
| Amount_* | Transaction amount, ISO formatted | M | Numeric | 9 | 1050 for 10.5 |
| OriginalTransactionPaymentUniqueNumber_* | Payment Unique Number of the transaction to be refunded, generated by merchant | M | Alphanumeric | 20 | |
| Status_* | Refund response code. | M | Alphanumeric | 30 | |
| StatusMessage_* | Description of received status<br>Will be filled only after a complete execution process. | M | Alphanumeric | 512 | |
| CurrencyCode_* | Numeric ISO Code for currency<br>634 for QAR | M | Numeric | 3 | 634 |
| ConfirmationID_* | Payment gateway confirmation reference for the refund transaction.<br>This parameter will not be returned in case of an Offline Refund. | M ( For Online refund) | Alphanumeric | 27 | |
| Lang_* | Language for the payment description parameter. Supported values are:<br>En & Ar | M | Alphanumeric | 2 | En |
| TransactionRequestDate_* | Transaction to be refunded request date<br>dd/MM/yyyy/ HH:mm:ss | M | Alphanumeric | 14 | |
| PUN_* | Payment Unique Number of the current refund transaction, generated by merchant | M | Alphanumeric | 20 | |
| EZConnectRequestStatus | EZ-Connect Request validation Status | M | Alphanumeric | 30 | |
| EZConnectStatusMessage | EZ-Connect Status Message | M | Alphanumeric | 30 | |
| EZConnectResponseDate | Request response date to be used in regenerating the secure hash<br>ddMMyyyyHHmmss | M | Alphanumeric | 30 | |

| | | | | | |
|---|---|---|---|---|---|
| SecureHash | Generated hex-encoded hash using hashing algorithm SHA-2 (256), by concatenating parameters as a single string starting with the merchant's secret key, then all parameters | M | Alphanumeric | 64 | |

# 4 Appendix

## 4.1 Appendix A: Date Format

| Letter | Usage | Sample |
|---|---|---|
| yyyy | Year | 2023 |
| MM | Month in year (01-12) | 08 |
| dd | Day in month (01-31) | 21 |
| HH | Hour in day (00-23) | 16 |
| mm | Minute in hour (00-59) | 45 |
| ss | Second in minute (00-59) | 51 |
| S | Millisecond | 8 |

## 4.2 Appendix B: Response Codes

### 4.2.1 Payment response codes

| Status Code | Description | Action to take |
|---|---|---|
| EZConnect-0001 | Missing “Action” Parameter | Provide the missing parameter |
| EZConnect-0002 | Missing “BankID” Parameter | Provide the missing parameter |
| EZConnect-0003 | Missing “PUN” Parameter | Provide the missing parameter |
| EZConnect-0004 | Missing “MerchantID” Parameter | Provide the missing parameter |
| EZConnect-0005 | Merchant is not available | Send the right merchant ID or check with PG operations team. |
| EZConnect-0006 | Merchant is not configured as EZ-Connect Merchant | Contact PG operator for support |
| EZConnect-0007 | Merchant has no configured secret key | The merchant admin should configure a Secret Key and use it in the hashing routine. |
| EZConnect-0008 | Merchant IP is not supported | Contact PG operator for support. |
| EZConnect-0009 | Secure Hash Could not be validated | Check the parameters used to hash and recheck the routine in the request parameters. |
| EZConnect-0010 | Missing “SecureHash” Parameter | Provide the missing parameter. |
| EZConnect-0012 | Missing “Lang” Parameter | Provide the missing parameter |
| EZConnect-0013 | Missing “OriginalTransactionPayment UniqueNumber” Parameter | Provide the missing parameter |
| EZConnect-0014 | Missing “Amount” Parameter | Provide the missing parameter |
| EZConnect-0015 | Missing “CurrencyCode” Parameter | Provide the missing parameter |
| EZConnect-0016 | Invalid amount value received | Provide the ISO formatted amount |
| EZConnect-0017 | Invalid amount format received contains decimal value | Provide the ISO formatted amount |
| EZConnect-0018 | Action type sent is invalid | Provide the right action type for the current request |
| EZConnect-0019 | Missing “TransactionRequestDate” Parameter | Provide the missing parameter. |
| 0000 | Payment was processed successfully | Transaction has been successfully completed. |

| | | |
|---|---|---|
| 2799 | Fraud validation error | Transaction didn’t complete due to validation error on fraud parameters sent from the merchant side. |
| 2810 | Possible duplicate refund. | Does not require confirmation, Correct the PUN value. |
| 2811 | Original transaction for refund is failed. | Does not require confirmation, only success transactions allowed to be refunded. |
| 2812 | Original transaction not confirmed yet. | Does not require confirmation, refund is allowed only for confirmed transactions, confirm transaction. |
| 2992 | Payment Status-Time out. | Transaction did not complete due to network timeouts; the transaction must be submitted again using different transaction ID. |
| 2993 | Merchant not active. | Transaction did not complete due merchant inactive record setup by the gateway, the transaction must be submitted again using different transaction ID after the merchant activation. |
| 2994 | Payment method selected | Transaction did not complete due to customer not completing the payment after selecting a payment method. |
| 2996 | Canceled before payment method selection | Transaction did not complete due to customer cancellation of the payment before selecting a payment method. |
| 2997 | Canceled before login | Transaction did not complete due to customer cancellation of the payment after selecting a payment method |
| 3000 | Payment Failed. | Transaction did not complete successfully due to issuer rejection (such as insufficient amount). |
| 4003 | Payment Failed, please refer to PG administrator. | Transaction did not complete successfully due to blacklist operation; the transaction cannot be submitted again. |
| 4004 | Payment Failed, please refer to PG administrator. | Transaction did not complete successfully due to blacklist operation; the transaction cannot be submitted again. |
| 4100 | Payment has been rejected | Transaction did not complete due to Payment Gateway Risk Rule violation. |
| 5002 | Refund Transaction is pending. | |
| 5003 | Refund Transaction rejected. | Does not require confirmation, Refund did not perform successfully due to backend rejection. |
| 5004 | Original transaction not found for Refund. | Does not require confirmation, Make sure you are refunding an existing transaction with valid data |
| 5006 | The provided currency does not match the original Transaction's currency. | Does not require confirmation, Correct the currency’s value. |
| 5007 | Abnormal error occurred while performing refund | Does not require confirmation, Refund cannot be performed due to abnormal error, and refund must be performed again. |
| 5008 | Invalid refund amount | Does not require confirmation, Correct the amount value, it should be full amount of original transaction to be refunded. |

| | | |
|---|---|---|
| 5011 | Refund rejected, transaction has a pending chargeback. | Does not require confirmation, you can try again later on to refund the original transaction. |
| 5012 | Refund rejected, transaction has already been charged back. | Does not require confirmation, a chargeback has been performed on the original transaction, so it won't be refunded. |
| 5013 | Original transaction for refund requires reversal. | Does not require confirmation, original transactions is require reversal or reversed. |
| 5015 | The transaction is already refunded. | Does not require confirmation. |
| 5018 | The original transaction is not Pay. | Does not require confirmation, refund is allowed only for Pay transactions. |
| 5019 | Payment validation failed. | Transaction did not complete due to send Item Id not predefined for the merchant. |
| 8107 | The payment amount is less than or equal to 0. | Transaction did not complete due to payment amount is less than or equal to 0 and it has not been logged, transaction must be submitted again using different transaction ID with amount greater than 0. |
| 8108 | Transaction currency not supported. | Transaction has been stopped due to currency code not supported, transaction must be submitted again using different transaction ID and correct currency code. |
| 8110 | Invalid data, you can only change card Details for an already used PUN | Transaction did not complete due to send transaction data being for a previous failed transaction with same OUN but also different data other the Card details such as Amount, quantity, etc. |
| 8200 | Current action is not supported for this merchant. | Transaction did not complete due to sale transaction is not supported; transaction must be submitted again using different transaction ID after supporting sale transaction from the gateway. |
| 8201 | Current IP is not valid for this merchant. | Transaction has been stopped due to invalid IP for the merchant, transaction must be submitted again using different transaction ID after adding the merchant IP by the gateway. |
| 8300 | Backend is Inactive. | Does not require confirmation, Refund did not perform successfully due to in active backend, The merchant should perform refund again after activation the backend by the acquirer admin. |
| 9001 | Unknown Backend Error. | Transaction did not complete due to Unknown back-end error, transaction must be submitted again using different transaction ID. |

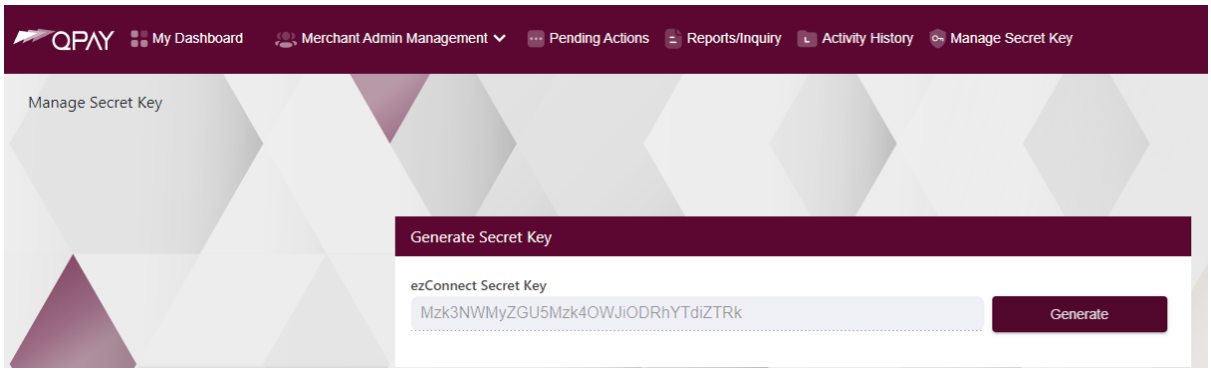
### 4.2.2 Inquiry response codes

| Status Code | Description | Action to take |
|---|---|---|
| 0000 | Inquiry Status-Inquiry Successful. | Transaction has been successfully completed. |
| 0001 | Inquiry Status- Missing “Action” Parameter | Provide the missing parameter |
| Inquiry-0002 | Inquiry Status-Missing “BankID” Parameter | Provide the missing parameter |
| Inquiry-0004 | Inquiry Status- Missing “MerchantID” parameter | Provide the missing parameter |
| Inquiry-0005 | Merchant is not available | Send the right merchant ID or check with PG operations team. |
| Inquiry-0006 | Merchant is not configured as EZ-Connect Merchant | Contact PG System operator for support |
| Inquiry-0007 | Inquiry Status- Merchant has no configured secret key | Contact PG System operator for support |
| Inquiry-0008 | Inquiry Status-Current IP is not valid for this merchant. | Transaction has been stopped due to invalid IP for the merchant, transaction must be submitted again using different transaction ID after adding the merchant IP by the gateway |
| Inquiry-0009 | Inquiry Status-Mismatch secure hash | Check the parameters used to hash, and recheck the routine in the request parameters |
| Inquiry-0010 | Inquiry Status-Missing “SecureHash” Parameter | Provide the missing parameter |
| Inquiry-0013 | Inquiry Status- Missing “OriginalPUN” Parameter | Provide the missing parameter |
| 0014 | Inquiry Status-Missing “Amount” Parameter | Provide the missing parameter |
| 0016 | Inquiry Status-Invalid amount format | Provide the right ISO formatted amount |
| 0018 | Inquiry Status-Action type sent is invalid | Provide the right action type for the current request |
| 19 | Inquiry Status-Missing “TransactionRequestDate” Parameter | Provide the missing parameter |
| 27 | Inquiry Status-Invalid transaction request date format | Provide the right transaction request date format Note: check Date Format for descriptive details about the date format. |
| 31 | Inquiry Status-Current action is not supported for this merchant. | Transaction did not complete due to sale transaction is not supported; transaction must be submitted again using different transaction ID after supporting sale transaction from the gateway |
| 1010 | An abnormal error occurred | Contact PG System operator for support |
| 2993 | Inquiry Status-Merchant not active. | Transaction did not complete due merchant inactive record setup by the gateway, the transaction must be submitted again using different transaction ID after the merchant activation |
| 8106 | Try to Inquiry about unfounded transaction | - |

## 4.3 Appendix C: Merchant Admin - Secret Key Configuration

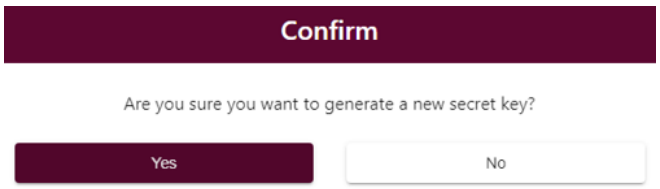
The “Manage Secret Key” page allows users to view/regenerate a secret key for each merchant.

- To access, click **[Manage Secret Key]**




User can then see the merchant’s current secret key or can click on **[Generate]** to generate a new secret key.

Once is clicked, a confirmation screen is pops up for the user to confirm before the process to replace the old secret key with a new one is intiated.



## 4.4 Appendix D: Secure Hash

This section clarifies how secure hash must be generated using all required request parameters and optional parameters if they are provided. Below examples show how to generate secure hash for each message type:

### 4.4.1 Payment

#### *4.4.1.1 Payment Request*

Input request parameters:

- SecretKey: 3aJYQWELpNywSw3O
- Action: 0
- Amount: 20000
- BankID: QIB
- CurrencyCode: 634
- ExtraFields_f14: https://merchantUrl.com/response
- Lang: en
- MerchantID: simulator123
- MerchantModuleSessionID: f859a6e54bd54de4ad3d
- NationalID: 2215275104
- PUN: f859a6e54bd54de4ad3d
- PaymentDescription: SamplePayment
- Quantity: 1
- TransactionRequestDate: 06082023161317

These parameters will be ordered alphabetically based on parameter name.
The input for secure hash generation routine would be:

3aJYQWELpNywSw3O**0**20000**QIB**634**https://merchantUrl.com/response**en**simulator123**f859a6e54bd54de4ad3d**2215275104**f859a6e54bd54de4ad3d**SamplePayment**106082023161317**

The secure hash would be:
2d4cd46694e97d22e3e86d6ec054afbda99e8c1f7ade8d99f4ae07544afa2ab8

#### *4.4.1.2 Payment Response*

Input response message:

- secret Key: 3aJYQWELpNywSw3O
- Response.AcquirerID: 030003
- Response.Amount: 20000
- Response.BankID: QIB
- Response.CardExpiryDate: 3006
- Response.CardHolderName: NOT_CAPTURED

- Response.CardNumber: 421537******3243
- Response.ConfirmationID: 20230806011317449943308335 2
- Response.CurrencyCode: 634
- Response.EZConnectResponseDate: 06082023161626
- Response.Lang: en
- Response.MerchantID: simulator123
- Response.MerchantModuleSessionID: f859a6e54bd54de4ad3d
- Response.PUN: f859a6e54bd54de4ad3d
- Response.Status: 0000
- Response.StatusMessage: Payment Processed Successfully

These parameters will be ordered alphabetically based on parameter name.
The input for secure hash generation routine would be:

3aJYQWELpNywSw3O**030003**20000**QIB**3006**NOT_CAPTURED**421537******3243**20230806011317449943308335 2**634**06082023161626**en**simulator123**f859a6e54bd54de4ad3d**f859a6e54bd54de4ad3d**0000**Payment+Processed+Successfully**

The secure hash would be:
660432ad41745cd716f8b799196e18c2cf84736f80014cc9816e0d53fd718760

### 4.4.2 Inquiry

#### *4.4.2.1 Inquiry Request*

Input response message:

- SecretKey: 3aJYQWELpNywSw3O
- Action=14
- BankID=QIB
- Lang=en
- MerchantID=simulator123
- OriginalPUN=f859a6e54bd54de4ad3d

The order of parameters will be:
Action, BankID, Lang, MerchantID, OriginalPUN

The input to the Secure Hash generation routine would be:

3aJYQWELpNywSw3O**14**QIB**en**simulator123**f859a6e54bd54de4ad3d**

The secure hash would be:
ca49b75c35d42f09fafda9f29d560039e4ae6074fcdcf4db854f39ef107afce1

#### *4.4.2.2 Inquiry Response*

Input response message:

- SecretKey: 3aJYQWELpNywSw3O

- Response.Status=0000
- Response.StatusMessage=Payment processed successfully.
- Response.OriginalStatus=0000
- Response.OriginalStatusMessage=Payment processed successfully.
- Response.OriginalConfirmationID=20230806011317449943308335 2
- Response.OriginalPUN=f859a6e54bd54de4ad3d
- Response.OriginalExtractStatus=
- Response.OriginalReversalStatus=
- Response.TransactionResponseDate=06082023161317
- Response.Amount=20000
- Response.CurrencyCode=634
- Response.MerchantID=simulator123
- Response.EZConnectResponseDate=07082023144016
- Response.BankID=QIB
- Response.Quantity=1
- Response.ApprovalCode=121212
- Response.CardExpiryDate=
- Response.CardHolderName=
- Response.CardNumber=421537******3243
- Response.PaymentDescription= SamplePayment
- Response.ItemID=
- Response.NationalID=2215275104

The order of parameters will be:

Amount, **ApprovalCode**, BankID, **CardExpiryDate**, CardHolderName, **CardNumber**, CurrencyCode, **EZConnectResponseDate**, ItemID, **MerchantID**, NationalID, **OriginalConfirmationID**, OriginalExtractStatus, **OriginalPUN**, OriginalReversalStatus, **OriginalStatus**, OriginalStatusMessage, **PaymentDescription**, Quantity, **Status**, StatusMessage, **TransactionResponseDate**

The input to the Secure Hash generation routine would be:

**3aJYQWELpNywSw3O**20000**121212**QIB**421537******3243**634**07082023144016**simulator123**2 215275104**20230806011317449943308335 2**f859a6e54bd54de4ad3d**0000**Payment+processed +successfully.**SamplePayment**10000**Payment+processed+successfully.**06082023161317**

The secure hash would be:
373c5b679b27eb0bb6f371faa4b60a15223ee7a2c3a95f593a192d8a2d1e4b76

### 4.4.3 Refund

#### *4.4.3.1 Refund Request*

Input request message:

- SecretKey : 3aJYQWELpNywSw3O
- Action: 6
- Amount_1: 1000
- BankID: QIB
- CurrencyCode: 634
- Lang: en
- MerchantID: simulator123
- OriginalTransactionPaymentUniqueNumber_1: f859a6e54bd54de4ad3d
- PUN_1: 62ab9838993a43a48909
- RequestDate:
- TransactionRequestDate: 06082023161932

These parameters will be ordered alphabetically based on parameter name.

The input for secure hash generation routine would be:

**3aJYQWELpNywSw3O**61**1000**QIB**634**en**simulator123**f859a6e54bd54de4ad3d**62ab9838993a43a48909**06082023161932

The secure hash would be:
99d5e09f096eeba95e893e7461554b49ba3d9c9c835a28dcc9db5b045cf38cae

#### *4.4.3.2 Refund Response*

Input response message:

- SecretKey : 3aJYQWELpNywSw3O
- Response.Amount_1=1000
- Response.ConfirmationID_1=
- Response.CurrencyCode_1=634
- Response.EZConnectRequestStatus=
- Response.EZConnectResponseDate=06082023162003
- Response.Lang_1=en
- Response.OriginalTransactionPaymentUniqueNumber_1=f859a6e54bd54de4ad3d
- Response.PUN_1=62ab9838993a43a48909
- Response.StatusMessage_1=Refund+Transaction+is+pending
- Response.Status_1=5002
- Response.TransactionRequestDate_1=06/08/2023 16:19:32

These parameters will be ordered alphabetically based on parameter name.

The input for secure hash generation routine would be:

**3aJYQWELpNywSw3O**1000**634**06082023162003**en**f859a6e54bd54de4ad3d**62ab9838993a43a48909**Refund+Transaction+is+pending**5002**06/08/2023 16:19:32

The secure hash would be:
91183a574d0a0771704c6ac91befe6890b75ab6c044edd2670ab32e3781a586b

## 4.5 Appendix E: Sample Code

### 4.5.1 Using PHP

#### *4.5.1.1 Inquiry*

```
<?php

$secretKey = '2-PGwB1oPGxyW+c8';

// Generate Inquiry Request
$map = generateInquiry($secretKey, '14', 'QIB', 'En', '22092014', 'tRYAowEOdHUVELSUGZdP');
//print_r($map);

$url = 'https://pguat.qcb.gov.qa/qcb-pg/api/gateway/2.0';

// Send Request
$response = sendRequest($url, $map);

handleResponse($response, $secretKey);


function handleResponse($response, $secretKey){

		// parse response into array
		parse_str($response, $queryArray);
   print_r($queryArray);

   $receivedHash = $queryArray['Response.SecureHash'];
   // calculate the Hash
   str_replace(' ', '+', $queryArray['Response.OriginalStatus']);
   str_replace(' ', '+', $queryArray['Response.StatusMessage']);
		$hashString = $secretKey . $queryArray['Response.Amount'] .
$queryArray['Response.ApprovalCode'] . $queryArray['Response.CardExpiryDate'] .
$queryArray['Response.BankID'] . $queryArray['Response.CardHolderName'] .
$queryArray['Response.CardNumber'] . $queryArray['Response.CurrencyCode'] .
$queryArray['Response.EZConnectResponseDate'] . $queryArray['Response.ItemID'] .
$queryArray['Response.MerchantID'] . $queryArray['Response.NationalID'] .
$queryArray['Response.OriginalConfirmationID'] . $queryArray['Response.OriginalExtractStatus'] .
$queryArray['Response.OriginalPUN'] . $queryArray['Response.OriginalReversalStatus'] .
$queryArray['Response.OriginalStatus'] . $queryArray['Response.OriginalStatusMessage'] .
$queryArray['Response.PaymentDescription'] . $queryArray['Response.Quantity'] .
$queryArray['Response.Status'] . $queryArray['Response.StatusMessage'] .
$queryArray['Response.TransactionResponseDate'];

		$digest = hash('sha256', $hashString);

		if($digest != $receivedHash){
			// proceed checking the status
		} else {
			// response should be rejected
		}
}

function sendRequest($url, $map){
   $options = [
```

```
        'http' => [
            'header' => "Content-type: application/x-www-form-urlencoded",
            'method' => 'POST',
            'content' => http_build_query($map),
        ],
    ];

    $context = stream_context_create($options);
    $result = file_get_contents($url, false, $context);
    if ($result === false) {
        /* Handle error */
    }

    var_dump('Result from HTTP CALL: ' .  $result . '</br>');
    return $result;
}

function generateInquiry($secretKey, $action, $bankId, $lang, $merchantId, $originalPun){
    $hash =  hashRequest($secretKey, $action, $bankId, $lang, $merchantId, $originalPun);
    $map = array();
    $map['Action'] = $action;
    $map['BankID'] = $bankId;
    $map['Lang'] = $lang;
    $map['MerchantID'] = $merchantId;
    $map['OriginalPun'] = $originalPun;
    $map['SecureHash'] = $hash;
    return $map;
}

function hashRequest($secretKey, $action, $bankId, $lang, $merchantId, $originalPun){
            $var = $secretKey . $action . $bankId . $lang . $merchantId . $originalPun;
    echo "Hash String: $var";
    echo "\n";
            $digest = hash('sha256', $var);
            echo "Hash Value: $digest";
    return $digest;
}

?>
```

#### 4.5.1.2 Payment

```
<?php

$secretKey = '2-PGwB1oPGxyW+c8';
$request = generatePayWebRequest(
    $secretKey,
    '0',
    '10',
    'QIB',
    '634',
    'http://localhost:8083/apex-merchant-simulator/response',
    'AR',
    '22092014',
    '22092014',
    '1232952100',
    'fUOdYiYrioMGGHwYFacW',
    'Payment desc',
    '1',
    '06082023170908'
```

```
);

// Send Request and forget waiting for response redirection
sendRequest($url, $map);

function onResponseReceived($response){

        // parse response into array
        parse_str($response, $queryArray);
   print_r($queryArray);

   $receivedHash = $queryArray['Response.SecureHash'];
   // calculate the Hash
   str_replace(' ', '+', $queryArray['Response.StatusMessage']);
        $hashString = $secretKey . $queryArray['Response.AcquirerID'] . $queryArray['Response.Amount']
. $queryArray['Response.BankID'] . $queryArray['Response.CardExpiryDate'] .
$queryArray['Response.CardHolderName'] . $queryArray['Response.CardNumber'] .
$queryArray['Response.ConfirmationID'] . $queryArray['Response.CurrencyCode'] .
$queryArray['Response.EZConnectResponseDate'] . $queryArray['Response.Lang'] .
$queryArray['Response.MerchantID'] . $queryArray['Response.MerchantModuleSessionID'] .
$queryArray['Response.PUN'] . $queryArray['Response.Status'] . $queryArray['Response.StatusMessage'] ;

        $digest = hash('sha256', $hashString);

        if($digest != $receivedHash){
                // proceed checking the status
        } else {
                // response should be rejected
        }
}


function sendRequest($url, $map){
   $options = [
      'http' => [
         'header' => "Content-type: application/x-www-form-urlencoded",
         'method' => 'POST',
         'content' => http_build_query($map),
      ],
   ];

   $context = stream_context_create($options);
   $result = file_get_contents($url, false, $context);
   if ($result === false) {
      /* Handle error */
   }

   var_dump('Result from HTTP CALL: ' .  $result . '</br>');
   return $result;
}


function generatePayWebRequest($secretKey, $action, $amount, $bankId, $currencyCode, $extraFields,
$lang, $merchantId, $merchantModuleSessionId,
   $nationalId, $pun, $paymentDescription, $quantity, $transactionRequestDate
) {
   $hash = hashPayWebRequest($secretKey, $action, $amount, $bankId, $currencyCode, $extraFields, $lang,
$merchantId, $merchantModuleSessionId,
   $nationalId, $pun, $paymentDescription, $quantity, $transactionRequestDate
   );
```

```
    $map = array();
    $map['Action'] = $action;
    $map['Amount'] = $amount;
    $map['BankID'] = $bankId;
    $map['CurrencyCode'] = $currencyCode;
    $map['ExtraFields_f14'] = $extraFields;
    $map['Lang'] = $lang;
    $map['MerchantID'] = $merchantId;
    $map['MerchantModuleSessionID'] = $merchantModuleSessionId;
    $map['NationalID'] = $nationalId;
    $map['PUN'] = $pun;
    $map['PaymentDescription'] = $paymentDescription;
    $map['Quantity'] = $quantity;
    $map['TransactionRequestDate'] = $transactionRequestDate;
    $map['SecureHash'] = $hash;


    return $map;
}


function hashPayWebRequest(
    $secretKey,
    $action,
    $amount,
    $bankId,
    $currencyCode
    ,
    $extraFields,
    $lang,
    $merchantId,
    $merchantModuleSessionId,
    $nationalId,
    $pun,
    $paymentDescription,
    $quantity,
    $transactionRequestDate
) {
    $var = $secretKey . $action . $amount . $bankId . $currencyCode . $extraFields . $lang . $merchantId .
$merchantModuleSessionId . $nationalId
        . $pun . $paymentDescription . $quantity . $transactionRequestDate;
    echo "Hash String: $var";
    echo "\n";
    $digest = hash('sha256', $var);
    echo "Hash Value: $digest";
    return $digest;
}
?>
```

#### *4.5.1.3 Refund*

```
<?php

$secretKey = '2-PGwB1oPGxyW+c8';
$amount = '2000';
$originalPun = 'fsfsdf34234234';
$pun = 'sfsdfsdf234234';
$request = generateRefund($secretKey, '6' , $amount, 'QIB', '634', 'En', '22092014', $originalPun, $pun,
'06082023170908');
```

```
$url = 'https://pguat.qcb.gov.qa/qcb-pg/api/gateway/2.0';
$response = sendRequest($url, $request);

handleResponse($response, $secretKey);

function handleResponse($response, $secretKey){

        // parse response into array
        parse_str($response, $queryArray);
   print_r($queryArray);

   $receivedHash = $queryArray['Response.SecureHash'];
   // calculate the Hash
        $hashString = $secretKey . $queryArray['Response.Amount_1'] .
$queryArray['Response.ConfirmationID_1'] . $queryArray['Response.CurrencyCode_1'] .
$queryArray['Response.EZConnectRequestStatus'] . $queryArray['Response.EZConnectResponseDate'] .
$queryArray['Response.Lang_1'] . $queryArray['Response.OriginalTransactionPaymentUniqueNumber_1'] .
$queryArray['Response.PUN_1'] . $queryArray['Response.StatusMessage_1'] .
$queryArray['Response.Status_1'] . $queryArray['Response.TransactionRequestDate_1'];

        $digest = hash('sha256', $hashString);

        if($digest != $receivedHash){
                // proceed checking the status
        } else {
                // response should be rejected
        }
}


function sendRequest($url, $map){
   $options = [
      'http' => [
         'header' => "Content-type: application/x-www-form-urlencoded",
         'method' => 'POST',
         'content' => http_build_query($map),
      ],
   ];

   $context = stream_context_create($options);
   $result = file_get_contents($url, false, $context);
   if ($result === false) {
      /* Handle error */
   }

   var_dump('Result from HTTP CALL: ' .  $result . '</br>');
   return $result;
}


function generateRefund($secretKey, $action, $amount, $bankId, $currencyCode, $lang, $merchantId,
$originalPun, $pun, $transactionRequestDate){
   $hash =  hashRequest($secretKey, $action, $amount, $bankId, $currencyCode, $lang, $merchantId,
$originalPun, $pun, $transactionRequestDate);
   $map = array();
   $map['Action'] = $action;
   $map['Amount_1'] = $amount;
   $map['BankID'] = $bankId;
   $map['CurrencyCode'] = $currencyCode;
   $map['Lang'] = $lang;
```

```
    $map['MerchantID'] = $merchantId;
    $map['OriginalTransactionPaymentUniqueNumber_1'] = $originalPun;
    $map['PUN_1'] = $pun;
    $map['RequestDate'] = '';
    $map['RequestDate'] = '';
    $map['TransactionRequestDate'] = $transactionRequestDate;
    $map['SecureHash'] = $hash;
    return $map;
}


function hashRefundRequest($secretKey, $action, $amounts, $bankId, $currencyCode, $lang, $merchantId,
$originalPun, $pun, $transactionRequestDate){

    $var = $secretKey . $action . $amounts . $bankId . $currencyCode . $lang . $merchantId . $originalPun .
$pun . $transactionRequestDate;
    echo "Hash String: $var";
    echo "\n";
    $digest = hash('sha256', $var);
    echo "Hash: Value: . $digest";
    return $digest;
}

?>
```

## 4.5.2 Using Java

### *4.5.2.1 Payment*

- Step 1: Prepare Payment Request and Send It to Redirect JSP Page (To Send a Post Request):

```
String SECRET_KEY = "3aJYQWELpNywSw3O";// Use Yours, Please Store Your Secret
StringBuilder hashBuilder = new StringBuilder();

hashBuilder.append(SECRET_KEY);

request.setAttribute("Action", "0");
hashBuilder.append("0");
request.setAttribute("Amount=", amount);
hashBuilder.append(amount);
request.setAttribute("BankID=", bankId);
hashBuilder.append(bankId);
request.setAttribute("CurrencyCode", "634");
hashBuilder.append("634");
request.setAttribute("ExtraFields_f14", extraField);
hashBuilder.append(extraField);
request.setAttribute("Lang=", lan);
hashBuilder.append(lan);
request.setAttribute("MerchantID", merchantId);
hashBuilder.append(merchantId);
request.setAttribute("MerchantModuleSessionID", pun);
hashBuilder.append(pun);
request.setAttribute("NationalID", nationalId);
hashBuilder.append(nationalId);
request.setAttribute("PUN", pun);
hashBuilder.append(pun);
request.setAttribute("PaymentDescription", description);
hashBuilder.append(description);
request.setAttribute("Quantity", "1");
hashBuilder.append("1");
```

```
request.setAttribute("TransactionRequestDate", transactionRequestDate);
hashBuilder.append(transactionRequestDate);

String hashBuilderString = hashBuilder.toString();

String secureHash = Hashing.sha256().hashString(hashBuilderString,
Charsets.UTF_8).toString();
request.setAttribute("SecureHash", secureHash);

request.setCharacterEncoding("");
request.setAttribute("PG_REDIRECT_URL", "https:// PG_REDIRECT_URL ");
request.getRequestDispatcher(response.encodeURL("redirect.jsp")).forward(request
, response);
```

- STEP 2: Create JSP Page send Request:

```
<%
// read the paramters from request
String redirectURL = (String) request.getAttribute("PG_REDIRECT_URL");
String action = (String) request.getAttribute("Action");
String amount = (String) request.getAttribute("Amount");
String bankID = (String) request.getAttribute("BankID");
String currencyCode = (String) request.getAttribute("CurrencyCode");
String extraFields_f14 = (String) request.getAttribute("ExtraFields_f14");
String lang = (String) request.getAttribute("Lang");
String merchantID = (String) request.getAttribute("MerchantID");
String merchantModuleSessionID = (String)
request.getAttribute("MerchantModuleSessionID");
String nationalID = (String) request.getAttribute("NationalID");
String pun = (String) request.getAttribute("PUN");
String paymentDescription = (String)
request.getAttribute("PaymentDescription");
String quantity = (String) request.getAttribute("Quantity");
String secureHash = (String) request.getAttribute("SecureHash");
String transactionRequestDate = (String)
request.getAttribute("TransactionRequestDate");
%>
<html>
<body onload="javascript:document.redirectForm.submit();">
<form action="<%=redirectURL%>" method="POST" name="redirectForm">
<input type="hidden" name="Amount" value="<%=amount%>" />
<input type="hidden" name="CurrencyCode" value="<%=currencyCode%>" />
<input type="hidden" name="PUN" value="<%=pun%>" />
<input type="hidden" name="MerchantModuleSessionID"
value="<%=merchantModuleSessionID%>" />
<input type="hidden" name="PaymentDescription" value="<%=paymentDescription%>"
/>
<input type="hidden" name="NationalID" value="<%=nationalID%>" />
<input type="hidden" name="MerchantID" value="<%=merchantID%>" />
<input type="hidden" name="BankID" value="<%=bankID%>" />
<input type="hidden" name="Lang" value="<%=lang%>" />
<input type="hidden" name="Action" value="<%=action%>" />
<input type="hidden" name="SecureHash" value="<%=secureHash%>" />
<input type="hidden" name="TransactionRequestDate"
value="<%=transactionRequestDate%>" />
<input type="hidden" name="ExtraFields_f14" value="<%=extraFields_f14%>" />
<input type="hidden" name="Quantity" value="<%=quantity%>" />
</form>
```

```
</body>
</html>
```

- STEP 3: Backend response validation

```
String SECRET_KEY = "3aJYQWELpNywSw3O";// Use Yours, Please Store Your Secret
Enumeration<String> parameterNames = request.getParameterNames();
// store all response Parameters to generate Response Secure Hash
// and get Parameters to use it later in your Code
Map<String, String> responseParameters = new TreeMap<String, String>();
while (parameterNames.hasMoreElements()) {
	String paramName = parameterNames.nextElement();
	String paramvalue = request.getParameter(paramName);
	responseParameters.put(paramName, paramvalue);
}
// Now that we have the map, order it to generate secure hash and compare it
// with the received
StringBuilder responseOrderdString = new StringBuilder();
responseOrderdString.append(SECRET_KEY);
for (String treeMapKey : responseParameters.keySet()) {
	responseOrderdString.append(responseParameters.get(treeMapKey));
}
System.out.println("Response Orderd String is " +
responseOrderdString.toString());
// Generate SecureHash with SHA256
String generatedsecureHash =
Hashing.sha256().hashString(responseOrderdString.toString(), Charsets.UTF_8)
		.toString();
// get the received secure hash from result map
String receivedSecurehash = responseParameters.get("Response.SecureHash");

if (!receivedSecurehash.equals(generatedsecureHash)) {
	// IF they are not equal then the response shall not be accepted
	System.out.println("Received Secure Hash does not Equal generated Secure
hash");
} else {
	// Complete the Action get other parameters from result map and do
	// your processes
	// Please refer to The Integration Manual to See The List of The
	// Received Parameters
	String status = responseParameters.get("Response.Status");
	System.out.println("Status is :" + status);
}
```

#### *4.5.2.2 Inquiry*

- Function to create inquiry request for a given inputs:

```
	private static String prepareInquiryRequest(String secretKey, String
merchantId, String OriginalPUN, String bankId,
			String lang) {

		StringBuilder hashBuilder = new StringBuilder();
		StringBuilder requestBuilder = new StringBuilder();

		hashBuilder.append(secretKey);
```

```
            requestBuilder.append("Action=14&");
            hashBuilder.append("14");
            requestBuilder.append("BankID=" + bankId + "&");
            hashBuilder.append(bankId);
            requestBuilder.append("Lang=" + lang + "&");
            hashBuilder.append(lang);
            requestBuilder.append("MerchantID=" + merchantId + "&");
            hashBuilder.append(merchantId);
            requestBuilder.append("OriginalPUN=" + OriginalPUN + "&");
            hashBuilder.append(OriginalPUN);

            String hashBuilderString = hashBuilder.toString();
            System.out.println("hashBuilderString: " + hashBuilderString);

            String secureHash = Hashing.sha256().hashString(hashBuilderString,
Charsets.UTF_8).toString();

            System.out.println("SecureHash: " + secureHash);
            requestBuilder.append("SecureHash=" + secureHash);
            System.out.println("Inquiry Request: " + requestBuilder);

            return requestBuilder.toString();
      }
}
```

- Sending Inquiry request and response handling:

```
String SECRET_KEY = "3aJYQWELpNywSw3O";
String requestQuery = prepareInquiryRequest(SECRET_KEY, "simulator123",
"f859a6e54bd54de4ad3d",
            "QIB", "en");
// Send the request
URL url = new URL("https://inquiryrequesturl");
URLConnection conn = url.openConnection();
conn.setDoOutput(true);
OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream(), "UTF-
8");
// write parameters
writer.write(requestQuery.toString());
writer.flush();

// Get the response
StringBuffer output = new StringBuffer();
BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream(), "UTF-8"));
String line;
while ((line = reader.readLine()) != null) {
      output.append(line);
}
writer.close();
reader.close();
// Output the response
System.out.println(output.toString());
// this string is formatted as a "Query String" -
// name=value&name2=value2.......
String outputString = output.toString();
```

```
// To read the output string you might want to split it
// on '&' to get pairs then on '=' to get name and value
// and for a better and ease on verifying secure hash you should put
// them in a TreeMap
String[] pairs = outputString.split("&");
Map<String, String> result = new TreeMap<String, String>();
// now we have separated the pairs from each other
// {"name1=value1","name2=value2",....}
for (String pair : pairs) {
	// now we have separated the pair to {"name","value"}
	String[] nameValue = pair.split("=");
	String name = nameValue[0];// first element is the name
	String value = nameValue[1];// second element is the value
	// put the pair in the result map
	result.put(name, value);
}
// Now that we have the map, order it to generate secure hash and
// compare it with the received one
StringBuilder responseOrderdString = new StringBuilder();
responseOrderdString.append(SECRET_KEY);
for (String treeMapKey : result.keySet()) {
	responseOrderdString.append(result.get(treeMapKey));
}
System.out.println("Response Orderd String is " +
responseOrderdString.toString());
// Generate SecureHash with SHA256
// Using DigestUtils from appache.commons.codes.jar Library
String generatedsecureHash = new
String(DigestUtils.sha256Hex(responseOrderdString.toString()).getBytes());
// get the received secure hash from result map
String receivedSecurehash = result.get("Response.SecureHash");

if (!receivedSecurehash.equals(generatedsecureHash)) {
	// IF they are not equal then the response shall not be accepted
	System.out.println("Received Secure Hash does not Equal generated Secure
hash");
} else {
	// Complete the Action get other parameters from result map and do
	// your processes
	// Please refer to The Integration Manual to See The List of The
	// Received Parameters
	String status = result.get("Response.Status");
	System.out.println("Status is :" + status);
}
```

#### *4.5.2.3 Refund*

- Function to create refund request for a given inputs:

```
private static String prepareRefundRequest(String secretKey, String merchantId,
String OriginalPUN, String amount,
                    String refundPun, String transactionRequestDate, String
bankId, String lang) {

            StringBuilder hashBuilder = new StringBuilder();
            StringBuilder requestBuilder = new StringBuilder();

            hashBuilder.append(secretKey);
```

```
            requestBuilder.append("Action=6&");
            hashBuilder.append("6");
            requestBuilder.append("Amount_1=" + amount + "&");
            hashBuilder.append(amount);
            requestBuilder.append("BankID=" + bankId + "&");
            hashBuilder.append(bankId);
            requestBuilder.append("CurrencyCode=634&");
            hashBuilder.append("634");
            requestBuilder.append("Lang=" + lang + "&");
            hashBuilder.append(lang);
            requestBuilder.append("MerchantID=" + merchantId + "&");
            hashBuilder.append(merchantId);
            requestBuilder.append("OriginalTransactionPaymentUniqueNumber_1="
+ OriginalPUN + "&");
            hashBuilder.append(OriginalPUN);
            requestBuilder.append("PUN_1=" + refundPun + "&");
            hashBuilder.append(refundPun);
            requestBuilder.append("RequestDate=&");
            requestBuilder.append("TransactionRequestDate=" +
transactionRequestDate + "&");
            hashBuilder.append(transactionRequestDate);

            String hashBuilderString = hashBuilder.toString();
            System.out.println("hashBuilderString: " + hashBuilderString);

            String secureHash = Hashing.sha256().hashString(hashBuilderString,
Charsets.UTF_8).toString();
            System.out.println("SecureHash: " + secureHash);
            requestBuilder.append("SecureHash=" + secureHash);
            System.out.println("Refund Request: " + requestBuilder);

            return requestBuilder.toString();
      }
}
```

- Sending Refund request and response handling:

```
String SECRET_KEY = "3aJYQWELpNywSw3O";
String requestQuery = prepareRefundRequest(SECRET_KEY, "simulator123",
"f859a6e54bd54de4ad3d", "1000",
            "62ab9838993a43a48909", "06082023161932", "QIB", "en");

// Send the request
URL url = new URL("https://localhost:9080/PayOneWeb/EZConnectRequestServlet");
URLConnection conn = url.openConnection();
conn.setDoOutput(true);
OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream(), "UTF-
8");
// write parameters
writer.write(requestQuery.toString());
writer.flush();
// Get the response
StringBuffer output = new StringBuffer();
BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream(), "UTF-8"));
String line;
while ((line = reader.readLine()) != null) {
      output.append(line);
```

```
}
writer.close();
reader.close();
// Output the response
System.out.println(output.toString());
// this string is formatted as a "Query String" -
// name=value&name2=value2.......
String outputString = output.toString();
// To read the output string you might want to split it
// on '&' to get pairs then on '=' to get name and value
// and for a better and ease on verifying secure hash you should put
// them in a TreeMap
String[] pairs = outputString.split("&");
Map<String, String> result = new TreeMap<String, String>();
// now we have separated the pairs from each other
// {"name1=value1","name2=value2",....}
for (String pair : pairs) {
	// now we have separated the pair to {"name","value"}
	String[] nameValue = pair.split("=");
	String name = nameValue[0];// first element is the name
	String value = nameValue[1];// second element is the value
	// put the pair in the result map
	result.put(name, value);
}
// Now that we have the map, order it to generate secure hash and
// compare it with the received one
StringBuilder responseOrderdString = new StringBuilder();
responseOrderdString.append(SECRET_KEY);
for (String treeMapKey : result.keySet()) {
	responseOrderdString.append(result.get(treeMapKey));
}
System.out.println("Response Orderd String is " +
responseOrderdString.toString());
// Generate SecureHash with SHA256
// Using DigestUtils from appache.commons.codes.jar Library
String generatedsecureHash = new
String(DigestUtils.sha256Hex(responseOrderdString.toString()).getBytes());
// get the received secure hash from result map
String receivedSecurehash = result.get("Response.SecureHash");
if (!receivedSecurehash.equals(generatedsecureHash)) {
	// IF they are not equal then the response shall not be accepted
	System.out.println("Received Secure Hash does not Equal generated Secure
hash");
} else {
	// Complete the Action get other parameters from result map and do
	// your processes
	// Please refer to The Integration Manual to See The List of The
	// Received Parameters
	String status = result.get("Response.Status_1");
	System.out.println("Status is :" + status);
}
```

### 4.5.3 Using C#

#### *4.5.3.1 Payment*

- Secure Hash Generation

```
static string GenerateSecureHash(String secretKey, String extraField, String
merchantId,
    String pun, String amount, String description, String transactionRequestDate,
String bankId,
    String nationalId, String lan)
{
    StringBuilder hashBuilder = new StringBuilder();
    StringBuilder requestBuilder = new StringBuilder();
    hashBuilder.Append(secretKey);

    requestBuilder.Append("Action=0&");
    hashBuilder.Append("0");
    requestBuilder.Append("Amount=" + amount + "&");
    hashBuilder.Append(amount);
    requestBuilder.Append("BankID=" + bankId + "&");
    hashBuilder.Append(bankId);
    requestBuilder.Append("CurrencyCode=634&");
    hashBuilder.Append("634");
    requestBuilder.Append("ExtraFields_f14=" + extraField + "&");
    hashBuilder.Append(extraField);
    requestBuilder.Append("Lang=" + lan + "&");
    hashBuilder.Append(lan);
    requestBuilder.Append("MerchantID=" + merchantId + "&");
    hashBuilder.Append(merchantId);
    requestBuilder.Append("MerchantModuleSessionID=" + pun + "&");
    hashBuilder.Append(pun);
    requestBuilder.Append("NationalID=" + nationalId + "&");
    hashBuilder.Append(nationalId);
    requestBuilder.Append("PUN=" + pun + "&");
    hashBuilder.Append(pun);
    requestBuilder.Append("PaymentDescription=" + description + "&");
    hashBuilder.Append(description);
    requestBuilder.Append("Quantity=1&");
    hashBuilder.Append("1");
    requestBuilder.Append("TransactionRequestDate=" + transactionRequestDate + "&");
    hashBuilder.Append(transactionRequestDate);

    String hashBuilderString = hashBuilder.ToString();
    Console.WriteLine("hashBuilderString: " + hashBuilderString);

    String secureHash = ComputeSha256Hash(hashBuilderString);
    Console.WriteLine("SecureHash: " + secureHash);
    requestBuilder.Append("SecureHash=" + secureHash);
    Console.WriteLine("Payment Request: " + requestBuilder);

    return requestBuilder.ToString();
}
static string ComputeSha256Hash(string rawData)
{
    // Create a SHA256
    using (SHA256 sha256Hash = SHA256.Create())
    {
        // ComputeHash - returns byte array
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

        // Convert byte array to a string
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < bytes.Length; i++)
        {
            builder.Append(bytes[i].ToString("x2"));
        }
        return builder.ToString();
    }
```

```
}
```

- Redirect Post Request Preparation

```
protected void Page_Load(object sender, EventArgs e, String extraField, String
merchantId,
String pun, String amount, String description, String bankId,
String nationalId, String lan)
{
String secretKey = "3aJYQWELpNywSw3O";
string transactionRequestDate = DateTime.Now.ToString("yyyyMMddHHmmss");

this.Context.Items.Add("Action", "0");
this.Context.Items.Add("Amount", amount);
this.Context.Items.Add("BankID", bankId);
this.Context.Items.Add("TransactionRequestDate", transactionRequestDate);

this.Context.Items.Add("NationalID", nationalId);
this.Context.Items.Add("PUN", pun);
this.Context.Items.Add("MerchantModuleSessionID", pun);
this.Context.Items.Add("MerchantID", merchantId);

this.Context.Items.Add("Lang", lan);
this.Context.Items.Add("CurrencyCode", "634");
this.Context.Items.Add("ExtraFields_f14", extraField);
this.Context.Items.Add("Quantity", "2");
this.Context.Items.Add("PaymentDescription", description);
this.Context.Items.Add("SecureHash", GenerateSecureHash(secretKey, extraField,
merchantId,
pun, amount, description, transactionRequestDate, bankId,
nationalId, lan));
this.Context.Items.Add("PG_REDIRECT_URL", "https://PG_REDIRECT_URL");
/* Now do the redirection */
Server.Transfer("Redirect.aspx", true);
}
```

- Redirect Post Request Submitting

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Redirect.aspx.cs"
Inherits="Redirect" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title></title>
</head>
<%
 string redirectURL = (string)this.Context.Items["PG_REDIRECT_URL"];
 string amount = (string)this.Context.Items["Amount"];
 string currencyCode = (string)this.Context.Items["CurrencyCode"];
 string pun = (string)this.Context.Items["PUN"];
 string merchantModuleSessionID =
(string)this.Context.Items["MerchantModuleSessionID"];
 string nationalID = (string)this.Context.Items["NationalID"];
 string merchantID = (string)this.Context.Items["MerchantID"];
 string bankID = (string)this.Context.Items["BankID"];
 string Lang = (string)this.Context.Items["Lang"];
 string action = (string)this.Context.Items["Action"];
 string secureHash = (string)this.Context.Items["SecureHash"];
```

```
 string transactionRequestDate =
(string)this.Context.Items["TransactionRequestDate"];
 string extraFields_f14 = (string)this.Context.Items["ExtraFields_f14"];
 string quantity = (string)this.Context.Items["Quantity"];
 string PaymentDescription = (string)this.Context.Items["PaymentDescription"];

%>
</html>
<!-- STEP 3: Create HTML Page send Request -->
<body onload="javascript:document.redirectForm.submit();">
    <form action="<%=redirectURL%>" method="POST" name="redirectForm">
        <input type="hidden" name="Amount" value="<%=amount%>" />
        <input type="hidden" name="CurrencyCode" value="<%=currencyCode%>" />
        <input type="hidden" name="PUN" value="<%=pun%>" />
        <input type="hidden" name="MerchantModuleSessionID"
value="<%=merchantModuleSessionID%>" />
        <input type="hidden" name="PaymentDescription"
value="<%=PaymentDescription%>" />
        <input type="hidden" name="NationalID" value="<%=nationalID%>" />
        <input type="hidden" name="MerchantID" value="<%=merchantID%>" />
        <input type="hidden" name="BankID" value="<%=bankID%>" />
        <input type="hidden" name="Lang" value="<%=bankID%>" />
        <input type="hidden" name="Action" value="<%=action%>" />
        <input type="hidden" name="SecureHash" value="<%=secureHash%>" />
        <input type="hidden" name="TransactionRequestDate"
value="<%=transactionRequestDate%>" />
        <input type="hidden" name="ExtraFields_f14" value="<%=extraFields_f14%>" />
        <input type="hidden" name="Quantity" value="<%=quantity%>" />
    </form>
</body>
</html>
```

- Redirect Payment Response

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="PaymenResponse.aspx.cs"
Inherits="PaymenResponse" %>
<%@ Import Namespace="System.Security.Cryptography" %>
<%
 string secretKey = "cd7e786b53fae43147778a8f1a3dcaf8";//store in secure place or in
DB global
class ex. Globals.SECRET_KEY
 NameValueCollection nvCollection = HttpUtility.ParseQueryString(Request.Url.Query);

 /* Loop over the parameters */
 SortedDictionary<string, string=string>
    dictionary = new SortedDictionary<String, String=String>
        (StringComparer.Ordinal);
        foreach (string k in nvCollection) {
        dictionary.Add(k, nvCollection[k]);
        }
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.Append(secretKey);

        foreach(KeyValuePair<string,string>
            kv in dictionary){
            stringBuilder.Append(kv.Value);
            }

            %>
            <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
            <html xmlns="http://www.w3.org/1999/xhtml">
            <head id="Head1" runat="server">
                <title></title>
            </head>
            <body>
                Response order is <%=stringBuilder.ToString()%>
                <%
                SHA256 sha256;
                byte[] bytes, hash;
                StringBuilder hex;
                bytes = Encoding.UTF8.GetBytes(stringBuilder.ToString().ToString());


                sha256 = SHA256Managed.Create();
                hash = sha256.ComputeHash(bytes);
                string hashString = string.Empty;
                foreach (byte x in hash)
                {
                hashString += String.Format("{0:x2}", x);
                }



                string responseHash = Request.Params.Get("Response.SecureHash");
                if (responseHash.Equals(hashString))
                {
                %>
                success
                <% }
                else
                { %>
                Didn't work
                <%
                } %>

            </body>
        </html>
    </string,string>
</String,>
</string,>
```

#### *4.5.3.2 Inquiry*

```
namespace Inquiry
{
    class Inquiry
    {
        static void Main(string[] args)
        {

            Send_Request("3aJYQWELpNywSw3O", "simulator123", "f859a6e54bd54de4ad3d",
"QIB", "en");
            Console.ReadLine();
        }

        private static void Send_Request(String secretKey, String merchantId, String
OriginalPUN, String bankId,
        String lang)
        {

            var client = new RestClient("https://pguat.qcb.gov.qa/qcb-
pg/api/gateway/2.0");
```

```
            var request = new RestRequest("", Method.Post);
            request.AddHeader("Accepts", "application/x-www-form-urlencoded");
            request.AddParameter("Action", "14");
            request.AddParameter("BankID", bankId);
            request.AddParameter("Lang", lang);
            request.AddParameter("MerchantID", merchantId);
            request.AddParameter("OriginalPUN", OriginalPUN);

            String secureHash = GenerateSecureHash(secretKey, merchantId,
OriginalPUN, bankId, lang);

            request.AddParameter("SecureHash", secureHash);
            RestResponse response = client.Execute(request);

            if (response.StatusCode != HttpStatusCode.OK)
            {
                Console.WriteLine("response: " + response.ErrorMessage);
                //response
            } else
            {
                //good response
                //Console.WriteLine("response: " + response.Content);
                String responseDic = response.Content;
                NameValueCollection nvCollection =
HttpUtility.ParseQueryString(responseDic);

                /* Loop over the parameters */
                SortedDictionary<string, string> dictionary = new
SortedDictionary<String,
               String>(StringComparer.Ordinal);
                foreach (string k in nvCollection)
                {
                    dictionary.Add(k, nvCollection[k]);
                }

                StringBuilder stringBuilder = new StringBuilder();
                stringBuilder.Append(secretKey);

                String receivedHash = dictionary["Response.SecureHash"];
                Console.WriteLine("Received Secure Hash " + receivedHash);

                foreach (KeyValuePair<string, string> kv in dictionary)
                {
                    Console.WriteLine(kv.Key);
                    if (kv.Key.Contains("Message"))
                    {
                        stringBuilder.Append(kv.Value.Replace(" ", "+"));
                        continue;
                    }
                    if(kv.Key != "Response.SecureHash")
                        stringBuilder.Append(kv.Value);
                }

                Console.WriteLine("Hash String: " + stringBuilder.ToString());
                String calculatedSecureHash =
ComputeSha256Hash(stringBuilder.ToString());
                Console.WriteLine("Calculated Secure Hash " + calculatedSecureHash);

                if(calculatedSecureHash == receivedHash)
                {
                    // handle the response based on the Status
                } else
                {
```

```
                    // Response should be rejected
                }
            }
        }

        private static String GenerateSecureHash(String secretKey, String
merchantId, String OriginalPUN, String bankId,
        String lang)
        {
            StringBuilder hashBuilder = new StringBuilder();
            hashBuilder.Append(secretKey);
            hashBuilder.Append("14");
            hashBuilder.Append(bankId);
            hashBuilder.Append(lang);
            hashBuilder.Append(merchantId);
            hashBuilder.Append(OriginalPUN);
            String hashBuilderString = hashBuilder.ToString();
            Console.WriteLine("hashBuilderString: " + hashBuilderString);
            String secureHash = ComputeSha256Hash(hashBuilderString);
            return secureHash;
        }

        static string ComputeSha256Hash(string rawData)
        {
            // Create a SHA256
            using (SHA256 sha256Hash = SHA256.Create())
            {
                // ComputeHash - returns byte array
                byte[] bytes =
sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

                // Convert byte array to a string
                StringBuilder builder = new StringBuilder();
                for (int i = 0; i < bytes.Length; i++)
                {
                    builder.Append(bytes[i].ToString("x2"));
                }
                return builder.ToString();
            }
        }
    }
}
```

#### *4.5.3.3 Refund*

```
namespace Refund
{
    class Refund
    {
        static void Main(string[] args)
        {

            Send_Request("3aJYQWELpNywSw3O", "simulator123", "6c506d42139e4fa2ab16",
"1000", "6c506d42139e4fa2ab17",
                    "06082023161932", "QIB", "en");
            Console.ReadLine();
        }

        private static void Send_Request(String secretKey, String merchantId, String
OriginalPUN, String amount,
        String refundPun, String transactionRequestDate, String bankId, String lang)
        {
```

```
            var client = new RestClient("https://pguat.qcb.gov.qa/qcb-
pg/api/gateway/2.0");
            var request = new RestRequest("", Method.Post);
            //request.AddHeader("content-type", "application/x-www-form-
urlencoded");
            request.AddHeader("Accepts", "application/x-www-form-urlencoded");

            StringBuilder requestBuilder = new StringBuilder();

            request.AddParameter("Action","6");
            request.AddParameter("Amount_1", amount);
            request.AddParameter("BankID", bankId);
            request.AddParameter("CurrencyCode", "634");
            request.AddParameter("Lang", lang);
            request.AddParameter("MerchantID", merchantId);
            request.AddParameter("OriginalTransactionPaymentUniqueNumber_1",
OriginalPUN);
            request.AddParameter("PUN_1", refundPun);
            request.AddParameter("RequestDate", "");
            request.AddParameter("TransactionRequestDate", transactionRequestDate);

            String secureHash = GenerateSecureHash(secretKey, merchantId,
OriginalPUN, amount,
        refundPun, transactionRequestDate, bankId, lang);

            request.AddParameter("SecureHash", secureHash);
            RestResponse response = client.Execute(request);

            if (response.StatusCode != HttpStatusCode.OK)
            {
                Console.WriteLine("response: " + response.ErrorMessage);
                //response
            }
            else
            {
                //good response
                //Console.WriteLine("response: " + response.Content);
                String responseDic = response.Content;
                NameValueCollection nvCollection =
HttpUtility.ParseQueryString(responseDic);

                /* Loop over the parameters */
                SortedDictionary<string, string> dictionary = new
SortedDictionary<String,
               String>(StringComparer.Ordinal);
                foreach (string k in nvCollection)
                {
                    dictionary.Add(k, nvCollection[k]);
                }

                StringBuilder stringBuilder = new StringBuilder();
                stringBuilder.Append(secretKey);

                String receivedHash = dictionary["Response.SecureHash"];
                Console.WriteLine("Received Secure Hash " + receivedHash);

                foreach (KeyValuePair<string, string> kv in dictionary)
                {
                    Console.WriteLine(kv.Key);
                    if (kv.Key.Contains("Message"))
                    {
                        stringBuilder.Append(kv.Value.Replace(" ", "+"));
```

```
                        continue;
                    }
                    if (kv.Key != "Response.SecureHash")
                        stringBuilder.Append(kv.Value);
                }

                Console.WriteLine("Hash String: " + stringBuilder.ToString());
                String calculatedSecureHash =
ComputeSha256Hash(stringBuilder.ToString());
                Console.WriteLine("Calculated Secure Hash " + calculatedSecureHash);

                if (calculatedSecureHash == receivedHash)
                {
                    // handle the response based on the Status
                }
                else
                {
                    // Response should be rejected
                }
            }
        }

        private static String GenerateSecureHash(String secretKey, String
merchantId, String OriginalPUN, String amount,
        String refundPun, String transactionRequestDate, String bankId, String lang)
        {
            StringBuilder hashBuilder = new StringBuilder();
            hashBuilder.Append(secretKey);
            hashBuilder.Append("6");
            hashBuilder.Append(amount);
            hashBuilder.Append(bankId);
            hashBuilder.Append("634");
            hashBuilder.Append(lang);
            hashBuilder.Append(merchantId);
            hashBuilder.Append(OriginalPUN);
            hashBuilder.Append(refundPun);
            hashBuilder.Append(transactionRequestDate);
            String hashBuilderString = hashBuilder.ToString();
            String secureHash = ComputeSha256Hash(hashBuilderString);
            return secureHash;
        }

        static string ComputeSha256Hash(string rawData)
        {
            // Create a SHA256
            using (SHA256 sha256Hash = SHA256.Create())
            {
                // ComputeHash - returns byte array
                byte[] bytes =
sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

                // Convert byte array to a string
                StringBuilder builder = new StringBuilder();
                for (int i = 0; i < bytes.Length; i++)
                {
                    builder.Append(bytes[i].ToString("x2"));
                }
                return builder.ToString();
            }
        }
    }
}
```

## 4.6 Appendix F: Technical Data and Endpoints for QPay

| Description | Endpoint/ Portal |
|---|---|
| BankID | <To be shared later> |
| **Staging** | |
| Payment, Inquiry and Refund | https://pguat.qcb.gov.qa/qcb-pg/api/gateway/2.0 |
| Acquirer Portal | https://pguat.qcb.gov.qa/merchant/app/app |
| Merchant Portal | https://pguat.qcb.gov.qa/qcb-portal/login |
| **Production** | |
| Payment, Inquiry and Refund | https://pg-api.qpay.gov.qa/qcb-pg/api/gateway/2.0 |
| Acquirer Portal | https://pg-ui.qpay.gov.qa/merchant/app/app |
| Merchant Portal | https://pg-ui.qpay.gov.qa/merchant/app/app |

**Note:**

As the acquirer and merchant portals are IP Restricted portals, the acquirer is required to share the Static Public IPs from where the user is accessing the portal.