Mobile Wallet

Wallet Integration Document

Document Type : Technical
Audience : Merchants adding Wallet as Payment Option
Version : 1.6

Contents

1. Summary

This document introduces Mobile Wallet. It further details the technical information required to integrate the wallet into a website. The intention of this document is to provide a simple and quick guide to technical team of a merchant who is integrating the Mobile Wallet into the website as a payment option.

2. Introduction

2.1 What is Mobile Wallet

It is an "e-Wallet" solution provided by Mobikwik. Users register and add money into this wallet using Debit/Credit card or Net banking procedure. Once the Wallet has money, it can be used to pay various online merchants, mobile bills, electricity charges and various other list of billers registered with mobikwik. To know a list of billers where a user can use his wallet to pay can be found at https://www.mobikwik.com/

For merchants integrating wallet as payment option will have the user auto registered for a wallet service. The flow is such that the user has quickest and easiest experience of payment. The complete flow is provided later in the document.

2.2 Benefits of using Mobile Wallet

1. **Very Easy and Quick Integration for Merchants**: Mobile Wallet is far easier to integrate than a traditional payment gateway. Just put a small code snippet provided by Mobikwik in your website code and you are ready to go!

2. **Faster and Easier for User**: When payment is made using Mobile Wallet, there is no need to use Card or Net Banking as money is directly deducted from wallet. So payments made using wallet take less time and effort.

3. **Higher Success Rate**: As there is no involvement of any payment gateway or bank during payment, there are fewer chances of failures.

4. **Highly Secure**: All payments are made on HTTPS to provide high level of security to user. After a payment is made, a server to server call is made to verify payment status. Servers to server calls are always more secure than browser based requests.

5. **Repeat Customers:** Once a user has created his wallet and added money, he can use this wallet for future payments on the website without the hassle of payment gateways. We believe this creates repeat customers, as the wallet is easiest one-click payment option.

3. Payment Options provided in Wallet:

3.1 Debit Card: VISA, Master, Diners, Discover

3.2 Credit Card: VISA, Master, Diners, Discover

3.3 Net Banking

3.4 52 Indian Banks registered with us for Net banking

Andhra Bank	Jammu & Kashmir Bank
Allahabad Bank	Karnataka Bank Axis
Bank	Karur Vysya Bank Bank of
Bahrain & Kuwait	Kotak Mahindra Bank Bank of
Baroda Corporate Accounts	Lakshmi Vilas Bank
Bank of Baroda Retail Accounts	Oriental Bank of Commerce
Bank of India	Punjab & Sind Bank
Bank of Maharashtra	Punjab National Bank Corporate Accounts
Bank of Rajasthan	Punjab National Bank Retail Accounts
Catholic Syrian Bank	Royal Bank of Scotland
Canara Bank	Ratnakar Bank
CitiBank	South Indian Bank
City Union Bank	Standard Chartered Bank
Central Bank of India	Shamrao Vitthal Co-operative Bank
Corporation Bank	State Bank of India
Deutsche Bank	State Bank of Mysore
Dena Bank	State Bank of Travancore
DCB Bank	State Bank of Hyderabad
Dhanalaxmi Bank	Syndicate Bank
Development Credit Bank	UCO Bank
Federal Bank	Union Bank of India
HDFC Bank	United Bank of India
ICICI Bank	Vijaya Bank
IDBI Bank	Yes Bank Indian
Bank	IndusInd Bank Indian
Overseas Bank	ING Vysya Bank

4. Wallet Transaction Flows:

4.1 Buyer is NOT already registered at Mobikwik

1. Buyer clicks "Pay With Mobile Wallet" button on merchant's website.

2. A POST request comes from merchant to Mobikwik with all **Request Parameters**.

3. Buyer gets registered automatically at mobikwik.com using the email id sent by merchant in request parameter "email". An email is sent to this email id with a password.

4. Payment page is displayed with different payment options (Card, Net Banking). Buyer chooses a payment option and completes the payment.

5. After payment completion, money is added to merchant's wallet and buyer is redirected to the redirecturl provided by merchant with all **Response** posted to redirecturl.

4.2 Buyer is already registered at Mobikwik

1. Buyer clicks "Pay With Mobile Wallet" button on merchant's website.

2. A POST request comes from merchant to Mobikwik with all request parameters.

3. Buyer is asked to provide his mobikwik.com password or One Time Password (OTP) sent to this mobile phone.

4. If buyer has enough money in his Mobile Wallet, he clicks a button to complete the payment. If enough money is not available in his wallet, he needs to click on "Add to Wallet And Pay" button. Then Payment page is displayed with different payment options (Card, Net Banking). Buyer chooses a payment option and completes the payment.

5. After payment is completed, money is added to merchant's wallet and buyer is redirected to the redirecturl provided by merchant with all response parameters posted to redirecturl.

5. Steps to Integrate Mobile Wallet as payment method

5.1 Register as a Merchant

Below are the steps to register for Mobile Wallet as a Merchant:

1. Go to http://176.58.109.104 and click on Signup with a valid email id.

2. An email will be sent by Mobikwik on above email id. Click on the link given in the email to verify your account.

3. Login and start filling the application. Provide all details and documents required about your company.

4. After application has been completed by merchant, it will be reviewed by Mobikwik team. It will take 1-2 days to review and approve the merchant application.

5. Confirm with your MobiKwik contact point to ensure your merchant application has been approved and your setup/processing fee deposit has been acknowledged.

5.2 Access Merchant Panel

Once the registration is complete you login into merchant panel. Merchant panel helps you support the Mobile wallet Integration and also allows you to manage and track your transactions.

1. Merchant Panel is accessible at http://176.58.109.104.
2. This is the same panel where you registered and uploaded your information for approval.
3. Two new tabs are available now :
    - **a.** Transactions
        - ✓ Here you can view you current balance.
        - ✓ You can also select a date range and check all the transaction made by users to pay to your account via Mobile Wallet.
        - ✓ Click the Excel file icon to download the detail in an excel file.
    - **b.** Technical Configuration
        - ✓ Download your latest wallet Guide. This is the latest version of guide you are currently viewing.
        - ✓ You can upload your logo. This logo will be displayed on Wallet Pages when user navigates to pay using Mobile Wallet.
        - ✓ Here you will also find Secret Key and Merchant ID.
            1. **Secret Key** is a password unique to each merchant. This is strictly not to be shared. While making API calls to Mobile Platform you will be required to use this secret key to generate a checksum. Main purpose of this is to protect your API calls to Mobile Wallet platform.
            2. **Merchant Identifier (merchantId)** is a unique identifier for each merchant. It a **userid** attached to your merchant account with mobile Wallet. While integrating the wallet on your site, you have to set this Merchant Identifier as "mid" in list f Request Parameters.
        - ✓ You can also download the integration code for testing on staging as well as code for production/live deployment.

***** Use dummy test card numbers when integrating with staging server using '**Test Code'**.

***** Use real test card numbers when integrating with Production Server using '**Production Code'**.

5.3 Technical Integration

Below are the steps that merchants need to follow for Technical Integration with Mobile Wallet:

1. When the application has been approved, login to your account at http://176.58.109.104 and click on Technical Configuration.
2. Click on **Download Production Code** button listed in Step -3 (Get Integration Code)
3. Enter the Redirect URL (this is the URL to which MobiKwik will post the transaction response).

4. Click on Generate button and you will get a generate_code.zip file. This file has the HTML form that you need to integrate on your website's payment page.

5. This HTML form has all request parameters (with dummy values) that must be POSTed to wallet api request url. Populate these parameters with actual values. This HTML form also has a submit button which submits all parameters to Mobile Wallet Request url.

   For further support please check Section **"Sample HTML Code for POST Request to mobikwilk"** to insert in your website to send POST parameters.

5.4 Integration Testing on Stage – Test Bed

Mobikwik provides a test bed for merchants to test their Wallet flow and run various test cases before going live on production.
Please follow the steps below to test the Mobile wallet with Mobikwik:

1. Login to the merchant panel
2. Click on Technical Configuration and then click on **Download Test Code** button listed in Step -3 (Get Integration Code)
3. Using the test code above u can test your integration changing POST parameters in the form.
4. Use the following Test cards to complete transaction

Type	CARD NO	CVV	EXPIRY	Name	Month
VISA	4012888888881881	123	2015	TEST	12
VISA	4123450131001381	879	2015	TEST	12
Master	5177194127672001	548	2015	TEST	12
Master	5453010000064154	896	2015	TEST	12

6. Request and Response Parameters

6.1 Request Parameters

Request Method: POST

Request URL: https://www.mobikwik.com/views/proceedtowalletpayment.jsp

Request Parameters:

Parameter name	Parameter Description	Minimum Length	Maximum Length	Characters Allowed	Other Validations/Constraints
email	Buyer's Email Id (May or may not be already registered on mobikwik.com) E.g. amit.gupta111@hotmail.com		50		Any valid and existing email id
amount	Transaction Amount e.g 345			Integer Value (0 to 9) only	No decimal, space, Comma or any other special character allowed. Must be in **Indian Rupees**, Minimum value = Rs. 5
cell	Mobile no of the buyer(May or may not be already registered on mobikwik.com) e.g. 9812398123	10	10	Numeric only (0 to 9)	No special characters allowed. No leading 0 or country code (as +91) allowed.
orderid	Unique id of every transaction, generated by merchant.	8	30	Alphanumeric only(A to Z, a to z, 0 to 9)	No special characters allowed.
mid	Unique Merchant ID given by Mobikwik to merchant. Will remain same for all transactions for a merchant. **Should not be shown to the user.** e.g. MBK1234				
merchantname	Merchant's website or brand name. Displayed to buyer during transaction flow. e.g "snapdeal.com" or "Flipkart"	1	50	A-Z, a-z, 0-9, dot (.), space, hyphen (-)	
redirecturl	URL on merchant's website on which Mobikwik will post response parameters when a transaction is completed (either successful or failed transaction). e.g. http://www.snapdeal.com/MobiKwikpaymentresponse.jsp https://www.flipkart.com/MobiKwikpaymentresponse.jsp				Use the protocol http or https.

showMobile	**If request is coming from Mobile**, this parameter should be sent with value="**true**". **This parameter is not required for Desktop requests**.	4	4		Only one possible value is "true"

6.2 Response Parameters

Mobikwik will POST transaction response on the *redirecturl* provided by merchant in request.

Response Parameters: There are 6 parameters that are sent in response

Parameter name	Parameter Description	Parameter Value
statuscode	Transaction status at Mobikwik	For success=0 For failure please refer response codes table
orderid	Merchant order id, same as passed by merchant in request	
refid	Unique Reference ID generated by Mobikwik for transaction e.g. 1780811131213	
amount	Transaction Amount, same as passed in request by merchant e.g. 345	
statusmessage	Transaction Status Message e.g. "Transaction completed Successfully"	Success or Failure message (refer response codes table)

6.3 Response Codes Table

Response Code	Response Message
0	Transaction completed successfully
10	Merchant secret key does not exist
20	User Blocked
21	Merchant Blocked
22	Merchant does not Exist
23	Merchant not registered on mobikwik
30	Wallet TopUp Failed
31	Wallet Debit Failed
32	Wallet Credit Failed
40	User cancelled transaction at Login page
41	User cancelled transaction at Wallet Top Up page
42	User cancelled transaction at Wallet Debit page
50	Order Id already processed with this merchant.
51	Length of parameter orderid must be between 8 to 30 characters
52	Parameter orderid must be alphanumeric only
53	Parameter email is invalid
54	Parameter amount must be integer only
55	Parameter cell is invalid. It must be numeric, have 10 digits and start with 7,8,9
56	Parameter merchantname is invalid. It must be alphanumeric and its length must be between 1 to 30 characters
57	Parameter redirecturl is invalid
60	User Authentication failed

70	Monthly Wallet Top up limit crossed
71	Monthly transaction limit for this user crossed
72	Maximum amount per transaction limit for this merchant crossed
73	Merchant is not allowed to perform transactions on himself
80	Checksum Mismatch
99	Unexpected Error
91	Orderid is Blank or Null

7. Advanced Integration using Checksum (Mandatory)

7.1 Check Status API

7.1.1 What is Check Status API

This is a **server to server call for transaction reconciliation.** For securing transactions through Mobile Wallet, you can additionally implement the below process to verify a transaction after MobiKwik has sent transaction response to your redirect url.

This API returns transaction status at Mobikwik in more secure way. So after a transaction has completed, merchant can use this API to know transaction status at Mobikwik.

7.1.2 Why Check Status API should be used

1. In a normal browser based call, if browser is closed by user or if internet connection is lost during transaction, response from Mobikwik to merchant will be lost. So merchant will not come to know about the status of transaction. To know the status of such transactions, Check API must be used.

2. It is a server to server call which is more secure than browser based calls.

3. It is Checksum based. So any tampering with the parameter will be detected by matching checksum.

7.1.3 How to use Check Status API

When a response is received back from Mobikwik on a 'redirect URL' , an additional server to server call can be done to confirm the status of transaction at Mobikwik Server.
This server to server call will be checksum protected. The call should be initiated from merchant providing an orderid and Mobikwik will reply back in an XML format the status of the order. Merchant system can use this response to confirm and validate the transaction. Both Request and Response are protected by a checksum. Refer Section "**Sample Codes for Calculating Checksum**" to understand how to calculate the checksum.

7.1.4 Description of Server to Server call

Post URL: https://www.mobikwik.com/wallet.do

7.1.4.1 Request Parameters

Parameter name	Parameter Description	Minimum Length	Maximum Length	Characters Allowed	Other Validations
action	action = gettxnstatus				
mid	Unique Merchant ID given by Mobikwik to merchant. Will remain same for all transactions for a merchant. Should not be shown to the user. e.g. MBK1234				
orderid	Unique id of each transaction, generated by merchant.	8	50	Alphanumeric only(A to Z, a to z, 0 to 9)	No special characters allowed.
checksum	To be calculated using above 3 parameters				

7.1.4.2 Response Parameters

Mobikwik sends a **XML response** with has following tags: statuscode, orderid, refid, amount, statusmessage, ordertype, checksum.

Parameter name	Parameter Description	Parameter Value
statuscode	Transaction status at Mobikwik	For success=0 For failure=1
orderid	Merchant order id, same as passed by merchant in request	
refid	Unique Reference ID generated by Mobikwik for transaction e.g. 1780811131213	

amount	Transaction Amount ,e.g. 345	
statusmessage	Transaction Status Message e.g. "Transaction completed Successfully"	Success or Failure message
ordertype		
checksum		

Sample Success XML response:
If all goes well, the Transaction status URL will return an XML which looks like this:

```
<wallet>
        <statuscode>0</statuscode>
        <orderid>102</orderid>
        <refid>1780811131213</refid>
        <amount>100</amount>
        <statusmessage>success</statusmessage>
        <ordertype>payment</ordertype>
        <checksum>cd345rfdr</checksum>
</wallet >
```

Sample Failure XML response:
The Transaction Status URL will return a failure along with reason if there is some problem.

```
<wallet>
        <statuscode>1</statuscode>
        <orderid>102</orderid>
        <refid>1780811131213</refid>
        <amount>100</amount>
        <statusmessage>failure</statusmessage>
        <ordertype>payment</ordertype>
        <checksum>cd345rfdr</checksum>
</wallet >
```

statuscode in this case will be non-zero and statusmessage tag will have the reason why this failure occurred

7.1.5 How to calculate checksum for Check API?

To calculate checksum, **a secret key will be generated and will be shared with the merchant**. **This secret key must never be made public and must never be shared with any third party**, as it may compromise the security of API communications.

The **process to calculate checksum** is as follows:

a. Create a concatenated string of all raw data values that are to be passed in the API. While creating the concatenated String, **surround each parameter with single quotes**.
b. The order in which the data must be kept to make the concatenated string is the order in which the parameters are listed. This is important to preserve the order of data.

c. Calculate checksum over the concatenated string using **HMAC SHA-256** algorithm.

d. **After receiving the response, calculate and match the response checksum**.

e. Attach the calculated checksum in the request as the data for a parameter named "checksum".

Example:

For calculating checksum for getting transaction status request:
Request Checksum: Create a concatenated string with variables in following order:
""+"gettxnstatus"+""+""+mid+""+""+orderid+""
Calculate check sum on this concatenated string using the secret key provided. Send the value of this calculated check sum in a parameter named "checksum" in the post request.
Response Checksum: After receiving response, merchant must calculate checksum using response parameters and match it with the response checksum received. Create a concatenated string with variables in following order:
""+statuscode+""+orderid+""+refid+""+amount+""+statusmessage+""+ordertype+""

If response checksum calculated by merchant does not match with checksum sent by Mobikwik in response, this call's response must be discarded and no changes must be done by merchants at their end on the basis of this call's response.

7.2 Refund API

7.2.1 What is Refund Status API

Refund API is a public API that a merchant can use to refund money to user's mobile wallet. Refund API is usually used by merchant when the product or services are declined by the user and he is claiming for a refund of money.

Please not that money will only be refunded into user's mobile wallet and not into his/her bank account. User can use this wallet balance to buy services from mobikwik.com or buy product/services from any merchant using mobile wallet as payment option.

7.2.2 How to use Refund API

Like Check Status API, Refund API also is a URL with specific Request and Response parameters. The request parameter has a checksum protected by the secret provided to you.

POST URL: https://www.mobikwik.com/wallet.do

7.2.2.1 Request Parameters

Parameter name	Parameter Description	Minimum Length	Maximum Length	Characters Allowed	Other Validations
action	action = **refundtouser**				
mid	Unique Merchant ID given by Mobikwik to merchant. Will remain same for all transactions for a merchant. Should not be shown to the user. e.g. MBK1234				
txid	Unique id of each transaction, generated by merchant. It is the orderId that was provided by merchant.	8	50	Alphanumeric only(A to Z, a to z, 0 to 9)	No special characters allowed.
email	Email Id of the user to whom refund has to be done.				A valid email id uniquely identifying the user who had done the transaction being refunded.
amount	The amount paid by user to merchant in this particular transaction				The amount should be exactly same as the amount of transaction, otherwise refund will be declined.
checksum	To be calculated using above 3 parameters				

7.2.2.2 Response Parameters

Mobikwik sends a **XML response** which has following tags: statuscode, status, refid, txid, statusmessage.

Parameter name	Parameter Description	Parameter Value
statuscode	Transaction status code at Mobikwik	For success=0 For failure=1
status	Transaction status text at Mobikwik	For success="success", for failure ="failure"
txid	Merchant transaction id, same as passed by merchant in request	
refid	Unique Reference ID generated by Mobikwik for refund transaction e.g. 1780811131213	
statusmessage	Transaction Status Message in case of failure only e.g. "Merchant does not have sufficient balance"	Success or Failure message

Sample Response XML:
Success:

```
<wallet>
        <status>success</status>
        <statuscode>0</statuscode>
        <statusmessage> Transaction Completed Successfully </statusmessage>
        <txid>abcde1234557</txid>
       <refId>13625666142170</refId>
</wallet>
```

Failure:

```
<wallet>
        <status>failure</status>
        <statuscode>1</statuscode>
        <statusmessage> Merchant does not have sufficient balance </statusmessage>
        <txid>1234AC1</txid>
        <refId>124224209A</refId>
</wallet>
```

7.2.3 How to calculate checksum for Refund API?

To calculate checksum, **a secret key will be generated and will be shared with the merchant in the merchant panel**. **This secret key must never be made public and must never be shared with any third party**, as it may compromise the security of API communications.

The **process to calculate checksum** is as follows:

a. Create a concatenated string of all raw data values that are to be passed in the API. While creating the concatenated String, **surround each parameter with single quotes**.
b. The order in which the data must be kept to make the concatenated string is the order in which the parameters are listed. This is important to preserve the order of data.

c. Calculate checksum over the concatenated string using **HMAC SHA-256** algorithm.

d. Attach the calculated checksum in the request as the data for a parameter named "checksum".

Example:

For calculating checksum for getting transaction status request:
Create a concatenated string with variables in following order:

"'"+"refundtouser"+"'"+mid+"'"+txid+"'"+amount+"'"+email+"'"

Calculate check sum on this concatenated string using the secret key provided.
Send the value of this calculated check sum in a parameter named "checksum" in the post request.

8. Sample HTML Code for POST Request to Mobikwik

Please use this sample code in your web page to insert a form which will submit POST parameters required by Mobikwik to process a payment for the order.
All you have to do is to replace the dummy values of parameters such as email, amount etc. with the actual values.
You can also save the following code in a file, save it with any name for e.g. as sampleWallet.html. Now just open this file from any browser and see the working.

```
<html>
<head>
 <script src="http://code.jquery.com/jquery-latest.js"></script>
 <script type="text/javascript" src="http://jzaefferer.github.com/jquery-validation/jquery.validate.js"></script>
<style>
.paywith {
background: url("http://merchant.mobikwik.com/static/paywith.png") no-repeat center center;
width: 260px;
height: 73px;
color: transparent;
background-color: #DDD;
}
</style>
</head>
<body>
<h1>Checkout with Mobikwik on merchant website</h1>
<script>
$(document).ready(function(){
   $("#demo_form").validate();
 });
</script>
        <form id="demo_form" action="https://176.58.117.32/mobikwik/views/proceedtowalletpayment.jsp"
method="POST">            <div style="display:block" id="datafields" >
                                <div id="mkdiv1">
                                </br><span>Email: </span><input class="required email" size="50" name="email" type="text"
id="email">
                                </div>
                                <div id="mkdiv2">
                                </br><span>Amount: </span><input class="required number" size="50" name="amount" type="text"
id="amt">
                                </div>
                                <div id="mkdiv3">
                                </br><span>Cell no: </span><input class="required number" size="50" maxlength="10" minlength="10"
name="cell" type="text" id="cell">
                                </div>
                                <div id="mkdiv4">
                                </br><span>Order ID: </span><input  size="30" name="orderid" type="text" id="orderid">

                                </div>
                                <div id="mkdiv4" style="padding-top:30px;">
                                <input class="paywith" type="submit" value="Proceed">
                                </div>    </div>
                                <input type="hidden" name="mid" value="MBK9002">
                                <input type="hidden" name="merchantname" value="XYZ.com">
                                <input type="hidden" name="redirecturl"
value="https://176.58.117.32/mobikwik/views/demoreturnmerchant.jsp">
</form>  <body>  </html>
```

9. Additional Features for Mobile Wallet

9.1 Your logo in the payment flow

You can now upload your logo in your merchant panel. During the user's transactions your logo will be shown to the user at various steps. Please login to the merchant panel and click on "Technical configuration" and then follow "Step 2" to customize your integration.

9.2 More Secure Payments (https)

All payments through the wallet payment system now take place over a more secure https connection increasing the security and user trust while making the payments.

9.3 Reduced number of steps to make a payment

The new flow has at least 1-2 pages lesser that the user goes through thereby improving success rate and making user experience better.

9.4 Card collection on page

Now, choosing credit cards and most debit cards as a payment option doesn't redirect the customer to the payment gateway page; instead card data is entered on the wallet page itself.

9.5 OTP for repeat users

Repeat users who do not remember their wallet password can simply get an OTP (one time password) sent to their registered mobile phone and continue their payment.

9.6 Forgot password

The "Forgot password" flow has been improved to enhance user experience. The forgot password option now directly sends the user a new password link via email, where they can click and reset the password.

9.7 Remember Me

Revisiting customers can enable "Remember Me" so they don't have to put their password when they come back next time to pay using the wallet.

10. Sample Codes for Calculating Checksum

Below are some **sample codes** to calculate checksum in various popular web programming languages:

10.1 PHP

```
<?php
$action = "gettxnstatus"; // fixed value
$merchantId = "MBK1234"; // Merchant ID given by Mobikwik
$orderId = "123123"; // merchant order id
// Merchant Secret Key given by Mobikwik
$secretKey = "ask43jcl78rersa5cvcc212";
$algo = 'sha256';
$checksum_string = "'{$action}''{$merchantId}''{$orderId}'";
$checksum =  hash_hmac($algo, $checksum_string, $secretKey);
?>
```

10.2 Ruby

```
require 'openssl'
@action = "gettxnstatus"; # fixed value
@merchantId = "MBK1234"; # Merchant ID given by Mobikwik
@orderId = "123123"; # merchant order id
# Merchant Secret Key given by Mobikwik
@secretKey = "ask43jcl78rersa5cvcc212"
@algo = "sha256"
@checksum_string = "'"+@action+"''"+@merchantId+"''"+
@orderId+"'"
@ checksum = OpenSSL::HMAC.hexdigest(@algo, @secretKey,
@checksum_string)
```

10.3 Python

```
import hmac
from hashlib import sha256;
secretKey = 'ask43jcl78rersa5cvcc212'; # Merchant Secret Key
given by Mobikwik
action = 'gettxnstatus'; # fixed value
merchantId = 'MBK1234' # Merchant ID given by Mobikwik
orderId = '123123';  # merchant order id
checksum_string =
"'{}''{}''{}'".format(action,merchantId,orderId);
a = hmac.new(secretKey, checksum_string, sha256) ;
checksum = a.digest().encode('hex');
```

10.4 Perl

```
use Digest::SHA qw(sha256);
use Digest::HMAC qw(hmac_hex);

$action = "gettxnstatus"; # fixed value
$merchantId = "MBK1234"; # Merchant ID given by Mobikwik
$orderId = "123123"; # merchant order id
# Merchant Secret Key given by Mobikwik
```

```
$secretKey = "ask43jcl78rersa5cvcc212";
$algo = "sha256";
$checksum_string = join '', "'",$action,"'",$merchantId,"'",
$orderId,"'";
$checksum = hmac_hex($checksum_string, $secretKey, \&sha256);
```

10.5 C#

```
using System;
using System.Security.Cryptography;
using System.Text;
public class HMACSHA256example {
 public static void Main(string[] Fileargs) {
   const string action = "gettxnstatus"; // fixed value
// Merchant ID given by Mobikwik
   const string merchantId = "MBK1234";
   const string orderId = "123123";     // merchant order id
// Merchant Secret Key given by Mobikwik
const string secretKey = "ask43jcl78rersa5cvcc212";
   const string checksum_string =
"'"+action+"'"+merchantId+"'"+ orderId+"'";

   byte[] secretkeyArr = Encoding.ASCII.GetBytes(secretKey);
   byte[] checksum_stringArr =
Encoding.ASCII.GetBytes(checksum_string);
   using (HMACSHA256 hmac = new HMACSHA256(secretkeyArr)) {
                    byte[] hashValue =
hmac.ComputeHash(checksum_stringArr);
               StringBuilder sb = new StringBuilder();
               foreach (byte b in hashValue)
                   sb.Append(b.ToString("X2"));
// Final Checksum
               string checksum = sb.ToString();
                    Console.WriteLine(checksum);
        }
    }  //end main
} //end class
```

10.6 Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class HMACSHA256example {
   public static void main(String[] args) {
           try {
                   String action = "gettxnstatus"; // fixed
value
// Merchant ID given by Mobikwik
String merchantId = "MBK1234";
                   String orderId = "123123"; // merchant order
id

// Merchant Secret Key given by Mobikwik
String secretKey = "ask43jcl78rersa5cvcc212";
                   String check_sum_string = "'" + action +
"'" + merchantId + "'"+ orderId + "'";
```

```
                    SecretKeySpec secret = new
SecretKeySpec(secretKey.getBytes(),"HmacSHA256");
                    Mac mac = Mac.getInstance("HmacSHA256");
                    mac.init(secret);
                    byte[] digest =
mac.doFinal(check_sum_string.getBytes());
                    String checksum = "";
                    for (byte b : digest) {
                            checksum = checksum +
String.format("%02x", b);
                    }
                    System.out.println(checksum);
            } catch (Exception e) {
                    e.printStackTrace();
            }
    }
}
```

10.7 JavaScript

Download Google library CryptoJS from

http://code.google.com/p/crypto-js/downloads/list and import hmac-sha256.js. You can also directly download or import hmac-sha256.js from this url:

http://crypto-js.googlecode.com/svn/tags/3.1.2/build/rollups/hmac-sha256.js

```
<script src="hmac-sha256.js"></script>
<script>
var action = "gettxnstatus"; // fixed value
// Merchant ID given by Mobikwik
var merchantId = "MBK1234";
var orderId = "123123"; // merchant order id
// Merchant Secret Key given by Mobikwik
var secretKey = "ask43jcl78rersa5cvcc212";
var checksum_string = "'"+action+"''"+merchantId+"''"+
orderId+"'";
      var checksum= CryptoJS.HmacSHA256(checksum_string,
secretKey);
</script>
```