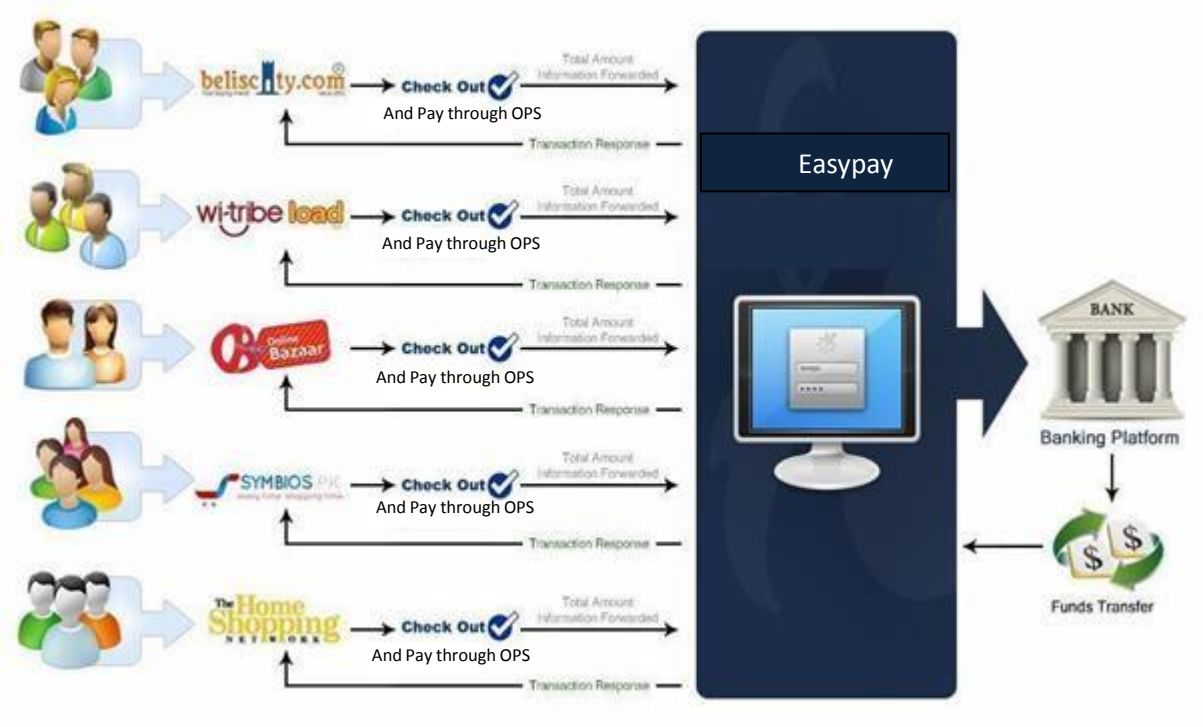
Easypay
Merchant Integration Guide

Table of Contents

Contents

1. Introduction

Easypay Solution is an electronic payment solution that enables internet users to make financial transaction online. It can easily and seamlessly be integrated with any online website and/or shopping cart and enable users to pay online through Easypay. It allows users to buy online using the monetary value available in their respective accounts. Easypay Solution provides a platform to local content, software/app developers, e-commerce merchants a viable payment method that they can use online. It is similar to PayPal which allows purchasing online, sending and receiving money and paying through user's PayPal accounts.



Easypay Solution is built upon latest state of the art technologies and toolset that leverages many features a secure payment solutions system must have. The users require no special technology or business license, only a valid email address to have an account on Easypay solution.

Apart from this, Easypay provides an Administrative/CRM solution which can be used by agents to administer the system and to help customers when they need. Different roles and responsibilities are defined to keep the system manageable and easy.

1.1. Intended Audience and Reading Suggestions

Reader	Description
Business	Any role within the organization that uses or defines the current software systems.
Development	Any role within the organization that creates and/or designs the current software Systems. This includes programmers, architects, and database administrators.
Quality Assurance	Any role within the organization that confirms the accuracy of the current or Requested functionality.
Business Analysis	Any role within the organization that analyzes the current functionality and business needs to propose the new or enhanced functionality.
Program and Project Management	Any role within the organization that determines and manages the program and project plans.
Merchants	Any merchant willing to integrate 'Easy Pay' payment method in his/her website

2. Plug-in Integration

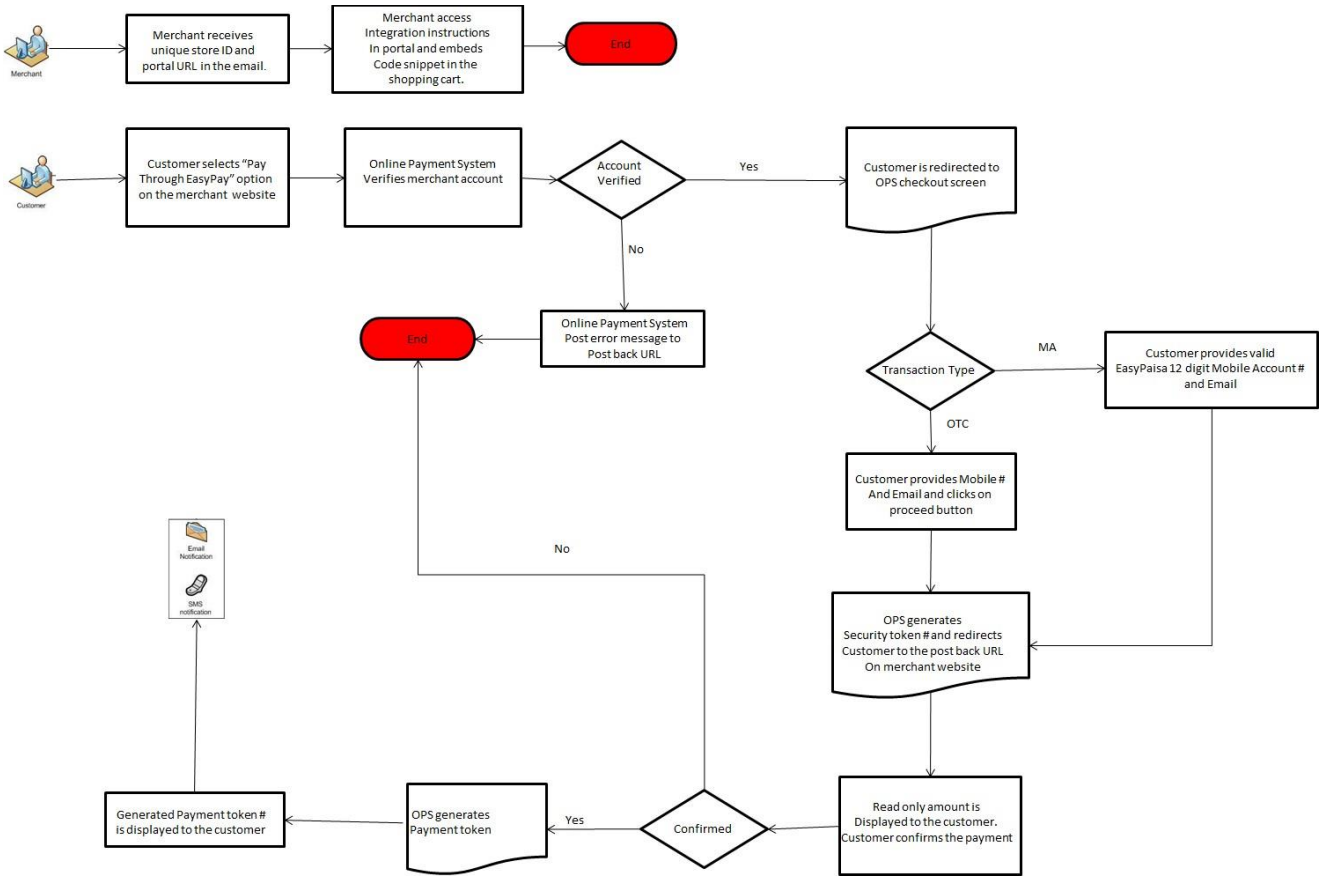
A plug-in is provided to Merchants enabling them to integrate Easypay Solution in their shopping sites. The plug-in is simple and seamless to integrate and it is compatible with all types of shopping carts/websites across different browsers and platforms.

Any Merchant registered can use this plug-in in their online retail sites. Easypay solution provides registered merchants a unique ID. By using that unique Id/store Id, merchants can easily integrate the plug-in with their online stores.

By integrating with any shopping cart/website the plug-in brings forth the authentication form and amount input field that the customer wants to pay to the merchants.

2.1 Plug-in Integration Process:

The below demonstrates the complete flow that is triggered for the online payments via the Plug-in.



2.1.1. Merchant Account Verification:

Customer selects “Pay through Easy Pay” payment method on merchant website. Easypay authenticates Merchant account, and then verifies store information whether or not store can be used for online transactions (valid Store Id, Valid Easypay Account). If Merchant’s account is not verified, an error message and description is sent to postBackURL.

2.1.2. Redirection to Secure Checkout Screen

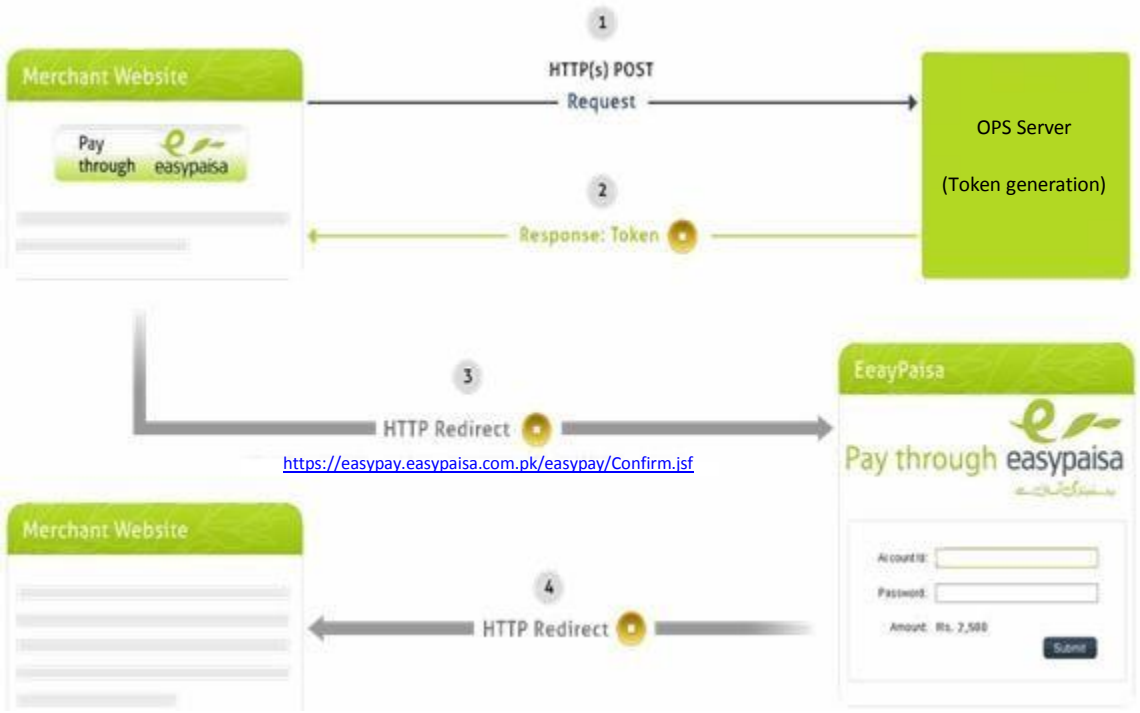
After successful merchant verification customer is redirected to Easypay secure checkout screen displaying following controls

- Your Mobile # - input field
- Your Email – input field
- Amount – Label displaying product(s) amount from merchant website
- Captcha
- Proceeds button

The integration allows merchant website to send unique Store ID, Amount, and Post Back URL, Order Reference Number and Expiry Date as post parameters on the Easypay URL upon redirection to PG secure checkout screen.

2.1.3. Generation of security/Authentication token by Easy Pay

On Easypay secure checkout screen when customer clicks on Proceed button, number of validations take place and if any of the validations fail, an error message along with description is generated. In case of success message a security token is generated and sent to Post Back URL. If no validations fail, a confirmation box opens displaying read-only amount to the customer.



2.1.4. Handling of security/auth_token by Merchant

Following is an example of how security Token can be handled at merchant website.

http://www.my-online-store.com/transaction/TokenHandler

```
protected void doPost(HttpServletRequest request, HttpServeletResponse response) throws
ServletException, IOException{
String token=request.getParameter("auth_token");
If(!token.equals("")){String postBackURL= http://wwww.my-online-
store.com/transaction/TransactionStatus.jsp;
try{response.getWriter().write(postBackURL);}
catch(Exception e){e.printStackTrace();}}}
```

2.1.5. Second redirection with security/authentication token to Easypay

Customer receives the security token and responds with another post back URL on confirmation of payment. As customer confirms the payment, merchant sends the same security token back to the Easypay establishing a handshake of trust between the two systems.

Easypay receives the new Post Back URL and sends the Success / Error message to the new URL. Easypay validates the token and following parameters are sent to the postBackURL.

- status
- desc
- orderRefNumber

Easypay processes the confirmed Payment, maintains transaction log, debits customer's account and credits Merchant's account. Following is an example of how these parameters will be handled by merchants on their websites.

```
protected void doPost(HttpServletRequest request,
response) throws ServletException, IOException {
String status = request.get.Parameter("status"); String desc = request.getParameter("desc");
Response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<h1> Showing Post Back Page Response On New URL</h1>;
out.println("<br/><br/>");
out.println("<br/><br/>");
out.println("--------------------------------------------------------------------------------");
out.println("<br/>");
out.println("<b>STATUS:</b>") <span style='color:red'>"+status+"</span>");
out.println("<br/><br/>");
out.println("<b>DESCRIPTION</b> <span style ='color:red'>"+desc+"</span>"); out.println("<br/>");
out.println("--------------------------------------------------------------------------------");
```

2.1.6. Checkout Confirmation by Customer:

Easypay sends notification to customer on provided mobile# and email id after receiving confirmation response from the customer.

2.2. Plug-in Integration Steps:

Following process will be followed by merchants to embed the 'Easypay' Plug-in in their stores:

- Merchant acquires an account through Easypay agents. A Welcome email containing unique Store ID and URL is sent to the Merchant after successful registration.

- Merchant logins the Easy Pay portal and access 'Guide to Integration' menu where Merchant is presented with step by step instructions to integrate the Easypay plug- in to their shopping cart/online retail shop.

Following is the sample of flow merchant should find after logging into the Easypay portal.

Merchants having unique store ID embed Easypay plug-in on checkout page of their online stores/ websites. This will integrate "Pay through Easypay" as a payment solution in their websites. The integration of Easypay plug-in is a simple two-step process:

1. The merchant needs to POST following parameter to the Easypay on the URL https://easypay.easypaisa.com.pk/easypay/Index.jsf:

    - amount
    - storeId
    - postBackURL
    - orderRefNum
    - expiryDate (Optional)
    - merchantHashedReq (Optional)
    - autoRedirect (Optional)
    - paymentMethod (Optional)

Parameter	O/M	Name	Explanation	Possible Values
Amount	M	amount	Total amount of the transaction in PKR	Numeric
Store Id	M	storeId	Id of the store as provided by the Telenor POC	Numeric
Post back URL	M	postBackURL	The first post back URL for confirmation	Character
Order Reference Number	M	orderRefNum	Merchant Generated order reference number	Character
Expiry Date	O	expiryDate	Merchant provided expiry date for the particular transaction	YYYYMMDD HHMMSS
Merchant Hashed Request	O	merchantHashedReq	Hash value for the request	Character
Auto Redirect	O	autoRedirect	If merchant wants to redirect to final post back URL and the end of transaction	0/1
Payment Method	O	paymentMethod	If merchant wants the customer to land on specific payment page	OTC_PAYMENT_METHOD/ MA_PAYMENT_METHOD/ CC_PAYMENT_METHOD

After successful redirection the customer would land on the Easypay Checkout Screen where there is a form to be filled regarding the transaction information.

2. After completing the form in Step 1 the customer will be pressing the Proceed Button and lands back on the merchant website on the same URL given in postbackURL variable in the first step. This will be a confirmation screen on merchant's website to perform a handshake between Easypay and merchant's website. The Easypay sends back a parameter named auth_token to the postbackURL which is sent as a GET parameter. Now the merchant needs to post back following two parameters again to the URL https://easypay.easypaisa.com.pk/easypay/Confirm.jsf:

    - auth_token
    - postBackURL

After this redirection the Easypay authenticates the auth_token sent by merchant with the one it has in the previous step, and upon successful authentication it will make customer land on the successful checkout screen sending back following two variables to the second postBackURL:

- status
- desc
- orderRefNumber

2.3. Plug-in Integration Code Samples:

Following Code Snippet can be used as a reference for the redirection performed in Step 1:

```
<form action=" https://easypay.easypaisa.com.pk/easypay/Index.jsf " method="POST" target="_blank">
        <! -- Store Id Provided by Easypay-->
        <input name="storeId" value="43" hidden = "true"/>
        <! -- Amount of Transaction from merchant's website -->
        <input name="amount" value="10" hidden = "true"/>
        <! – Post back URL from merchant's website -- >
        <input name="postBackURL" value=" http://www.my.online-store.com/transaction/MessageHandler" hidden = "true"/>
        <! – Order Reference Number from merchant's website -- >
        <input name="orderRefNum" value="1101" hidden = "true"/>
        <! – Expiry Date from merchant's website (Optional) -- >
        <input type =”hidden” name=”expiryDate” value=”20140606 201521”>
        <! – Merchant Hash Value (Optional) -- >
        <input type =”hidden” name=”merchantHashedReq” value=”askldjflaksdjflkasdf======asdfas dfkjaskdf”>
        <! – If Merchant wants to redirect to Merchant website after payment completion (Optional) -- >
        <input type =”hidden” name=”autoRedirect” value=”0”>
        <! – If merchant wants to post specific Payment Method (Optional) -- >
        <input type =”hidden” name=”paymentMethod” value=”0”>
        <! – This is the button of the form which submits the form -- >
        <input type = “image” src=”checkout-button-with-logo.png border=”0” name= “pay”>
</form>
```

Output:

After successful implementation, the below button would appear on the website of the merchant.

Following Code Snippet can be used as a reference for the redirection performed in Step 2:

```
<form action=" https://easypay.easypaisa.com.pk/easypay/Confirm.jsf " method="POST" target="_blank">
        <input name="auth_token" value="<?php echo $_GET['auth_token'] ?>" hidden = "true"/>
        <input name="postBackURL" value=" http://www.my.online-store.com/transaction/MessageHandler1" hidden =
"true"/>
        <input value="confirm" type = "submit" name= "pay"/>
</form>
```

Following Code Snippet can be used as a reference to retrieve the transaction status sent on the fly by Easypay:

```
        <input name="auth_token" value="<?php echo $_GET['status'] ?>"/>
        <input name="postBackURL" value="<?php echo $_GET['desc'] ?>"/>
        <input name="orderRefNumber" value="<?php echo $_GET['orderRefNumber'] ?>"/>
```

3. Open API

Easypay provides the capability for B2B integrations by exposing core services to external partners to reuse their existing interfaces in order to integrate with Easypay. There will be no redirection to Easypay checkout page and external partner's system will directly invoke Easypay APIs for initiating and inquiring the transaction.

Following two APIs are in-scope:

1. Initiate a Transaction
2. Inquire Transaction Status
3. Initiate Credit Card Transaction

The communication protocol supported is SOAP over HTTPs. External Systems should have the capability to perform SSL based communication with the Easypay Load Balancer.

The web service may be integrated using the following link:
https://easypay.easypaisa.com.pk/easypay-service/PartnerBusinessService/META-INF/wsdl/partner/transaction/PartnerBusinessService.wsdl

Integrate via Open API

Following are the steps that required for integration via Open API:

1. Generate a WSDL client using the WSDL URL (https://easypay.easypaisa.com.pk/easypay-service/PartnerBusinessService/META-INF/wsdl/partner/transaction/PartnerBusinessService.wsdl)

2. After the client generation three SOAP operations would be exposed as follows:

    a. initiateTransaction – This operation can be used to initiate OTC and MA transactions only.
    b. inquireTransaction – This operation can be used to inquire about the status of any transaction.
    c. initiateCCTransaction – This operation can be used to initiate Credit Card transactions only.

Testing Open API Operations

SOAP UI tool v.5 or above can be used in order to test open APIs.

3.1. Initiate Transaction:

Request Parameters

Field Name	Description	Mandatory(M)/O	Data
Username	This will be provided by Easypay using "Manage Partner Accounts" screen.	M	String
password	Encrypted password generated using "Manage Partner Accounts" screen.	M	String
channel	Channel through which transaction is initiated. (E.g. Internet, JPOS Device, etc.)		String
orderId	Merchant's system generated Order Id	M	String
storeId	Store ID generated during merchant registration in Easypay	M	Integer
transactionAmoun	Total Transaction Amount	M	String
transctionType	Type of transaction. Possible values are: OTC, MA, CC	M	String
msisdn	Customer's MSISDN	M in case of OTC O in case of MA M in case of CC	String
mobileAccountNo	Customer's Mobile Account #	O in case OTC M in case of MA	String
emailAddress	Customer's Email Address	O	String

Response Parameters

Field Name	Description	Data Type
responseCode	Easypay generated response code. Possible values are: 0000 - Success 0001 - System Error 0002 - Required field is missing 0003 - Invalid Order ID 0004 - Merchant Account does not exist 0005 - Merchant Account is not active 0006 - Store does not exist 0007 - Store is not active	String
orderId	Merchant's system generated Order Id	String
storeId	Store ID generated during merchant registration in Easypay	Integer
paymentToken	Token generated in case of OTC	String
transactionId	Transaction Id of FUNDAMO System for MA transactions only.	String
transactionDateTime	Transaction Date Time	Datetime
paymentTokenExiryDateTime	Token Expiry Date time	Datetime

Sample Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dto="http://dto.transaction.partner.pg.systems.com/" xmlns:dto1="http://dto.common.pg.systems.com/">
   <soapenv:Header />
   <soapenv:Body>
      <dto:initiateTransactionRequestType> <dto1:username>pg-
         systems</dto1:Username>
         <dto1:password>9b01234324vxddc0b</dto1:Password>
         <channel>Internet</channel>
         <orderId>000000001</orderId> <storeId>465</storeId>
         <transactionAmount>5000.00</transactionAmount>
         <transactionType>OTC</transactionType>
         <msisdn></msisdn>
         <mobileAccountNo>034632401722</mobileAccountNo>
         <emailAddress>abc@test.com</emailAddress>
      </dto:initiateTransactionRequestType>
   </soapenv:Body>
</soapenv:Envelope>
```

Sample Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns3:initiateTransactionResponseType
      xmlns:ns2="http://dto.common.pg.systems.com/"
xmlns:ns3="http://dto.transaction.partner.pg.systems.com/">
         <ns2:responseCode>0000</ns2:responseCode>
         <orderId>000000001</orderId>
         <storeId>465</storeId>
         <paymentToken>064463</paymentToken> <transactionDateTime>2014-04-
         25T19:59:37.981+05:00</transactionDateTime> <paymentTokenExiryDateTime>2014-04-
         26T19:59:37.968+05:00</paymentTokenExiryDateTime>
      </ns3:initiateTransactionResponseType>
   </soapenv:Body>
</soapenv:Envelope>
```

3.2. Inquire Transaction:

Request Parameters

Field Name	Description	Mandatory (M)	Data Type
username	This will be provided by Easypay using “Manage Partner Account” screen	M	String
password	Encrypted password generated by “Manage Partner Account” screen	M	String
orderId	Merchant’s system generated Order Id	M	String
accountNum	Merchant’s Account Number registered with Easypay	M	String

Response Parameters

Field Name	Description	Data Type
responseCode	Easypay generated response code. Possible values are: 0000 - Success 0001 - System Error 0002 - Required field is missing 0003 - Invalid Order ID 0004 - Merchant Account does not exist 0005 - Merchant Account is not active 0006 - Store does not exist 0007 - Store is not active	String
orderId	Merchant’s system generated Order Id	String
storeId	Merchant Account # registered in Fundamo	Integer
accountNum	Merchant Account No registered with Easypay	String
storeName	Merchant Store Name	String
paymentToken	Token generated in case of OTC	String
transactionId	Easypay generated unique Transaction Id	String
transactionStatus	Transaction Status possible Values are: REVERSED, PAID,	String
transactionAmount	Total transaction Amount	Double
transactionDateTime	Transaction Datetime	Datetime
paymentTokenExiryDateTime	Token expiration date time in case of OTC	Datetime
transactionPaidDateTime	Transaction Paid Date Time	Datetime
Msisdn	Customer MSISDN	String
paymentMode	Mode of payment (OTC, MA, ATM)	TransactionType

Sample Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dto="http://dto.transaction.partner.pg.systems.com/" xmlns:dto1="http://dto.common.pg.systems.com/">
   <soapenv:Header />
   <soapenv:Body>
      <dto:inquireTransactionRequestType> <dto1:username>pg-
         systems</dto1:Username>
         <dto1:password>9b01234324vxddc0b</dto1:Password>
         <orderId>0000001</orderId>
         <accountNum>9999999999999999</accountNum>
      </dto:inquireTransactionRequestType>
   </soapenv:Body>
</soapenv:Envelope>
```

Sample Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
      <ns3:inquireTransactionResponseType
      xmlns:ns2="http://dto.common.pg.systems.com/" xmlns:ns3="http://dto.transaction.partner.pg.systems.com/">
         <ns2:responseCode>0000</ns2:responseCode>
         <orderId>000001</orderId>
         <accountNum>9999999999999999</accountNum>
         <storeId>1001</storeId> <storeName>Test
         Store</storeName>
         <paymentToken>668551</paymentToken>
         <transactionId>100591</transactionId>
         <transactionStatus>PAID</transactionStatus>
         <transactionAmount>10000.00</transactionAmount> <transactionDateTime>2014-04-
         25T19:59:37.981+05:00</transactionDateTime> <paymentTokenExiryDateTime>2014-04-
         25T19:59:37.981+05:00</paymentTokenExiryDateTime> <transactionPaidDateTime>2014-04-
         25T19:59:37.981+05:00</transactionPaidDateTime> <paymentTokenExiryDateTime>2014-04-
         26T19:59:37.968+05:00</paymentTokenExiryDateTime> <msisdn>03463240172</msisdn>

         <paymentMode>OTC</paymentMode>
      </ns3:inquireTransactionResponseType>
  </soapenv:Body>
</soapenv:Envelope>
```

3.3 Initiate Credit Card Transaction

Request Parameters

Field Name	Description	Mandatory(M)/O	Data
Username	This will be provided by Easypay using "Manage Partner Accounts" screen.	M	String
password	Encrypted password generated using "Manage Partner Accounts" screen.	M	String
orderId	Merchant's system generated Order Id	M	String
storeId	Store ID generated during merchant registration in Easypay	M	Integer
transactionAmoun	Total Transaction Amount	M	String
transctionType	Type of transaction. Possible values are: OTC, MA, CC	M	String
msisdn	Customer's MSISDN	M	String
emailAddress	Customer's Email Address	O	String
cardType	Type of Credit Card in case of Credit Card Transaction	M	String
pan	Personal Account Number of the customer	M	String
expiryYear	Year of expiry of Credit Card	M	String
expiryMonth	Month of expiry of Credit Card	M	String
cvv2	CVV code of credit card at the back side	M	String

Response Parameters

Field Name	Description	Data Type
responseCode	Easypay generated response code. Possible values are: 0000 - Success 0001 - System Error 0002 - Required field is missing 0003 - Invalid Order ID 0004 - Merchant Account does not exist 0005 - Merchant Account is not active 0006 - Store does not exist 0007 - Store is not active	String
orderId	Merchant's system generated Order Id	String
storeId	Store ID generated during merchant registration in Easypay	Integer
paymentToken	Token generated in case of OTC	String
transactionId	Transaction Id of FUNDAMO System for MA transactions only.	String
transactionDateTime	Transaction Date Time	Datetime
paymentTokenExiryDateTime	Token Expiry Date time	Datetime

Sample Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dto="http://dto.transaction.partner.pg.systems.com/" xmlns:dto1="http://dto.common.pg.systems.com/">
   <soapenv:Header />
   <soapenv:Body>
      <dto:initiateCCTransactionRequestType>
        <dto1:username>pg-systems</dto1:username>
        <dto1:password>?</dto1:password>
        <orderId>1101</orderId>
        <storeId>23</storeId>
        <transactionAmount>10</transactionAmount>
        <transactionType>CC</transactionType>
        <msisdn>03323685741</msisdn>
        <emailAddress>test@test.com</emailAddress>
        <cardType>Mastercard</cardType>
        <pan>5313581000123430 </pan>
        <expiryYear>17</expiryYear>
        <expiryMonth>05</expiryMonth>
        <cvv2>123</cvv2>
      </dto:initiateCCTransactionRequestType>
   </soapenv:Body>
</soapenv:Envelope>
```

Sample Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns3:initiateTransactionResponseType
      xmlns:ns2="http://dto.common.pg.systems.com/"
xmlns:ns3="http://dto.transaction.partner.pg.systems.com/">
         <ns2:responseCode>0000</ns2:responseCode>
         <orderId>000000001</orderId>
         <storeId>465</storeId>
         <paymentToken>064463</paymentToken> <transactionDateTime>2014-04-
         25T19:59:37.981+05:00</transactionDateTime> <paymentTokenExiryDateTime>2014-04-
         26T19:59:37.968+05:00</paymentTokenExiryDateTime>
      </ns3:initiateTransactionResponseType>
   </soapenv:Body>
</soapenv:Envelope>
```

4. Instant Payment Notification

4.1. Purpose

Instant payment notification message is used to notify merchants about the details of any particular transaction made by customers using Easypay channel. IPN message is self-configured in Easypay for each merchant. It is customized as to what details merchant requires in response for any particular transaction. The IPN message service sends merchant a notification whenever Easypay transaction is created (with 'Paid' status) or is updated to 'Paid'.

IPN Handler URL is the Merchant's IPN listener URL where Merchant will be expecting the response from Easypay. IPN message will contain the selected parameters configured for merchant using below mentioned screen in merchant portal.

4.2. Payments Flow

1- Transaction is initiated by the customer through merchant's site on check out page.

2- Available Payment Method selected from the Easypay site

a. Payment Mode - OTC:

- OTC token is generated by PG and shown to customer.
- Customer pays the amount against the OTC token at authorized shop.
- UBPS updates the transaction status in Easypay.

b. Payment Mode - MA:

- Easypay invokes Fundamo API to initiate a transaction
- Fundamo sends a confirmation message to the customer.
- Customer confirms the transaction.
- Fundamo sends the response back to Easypay and system creates a transaction with paid status.

c. Payment Mode - CC:

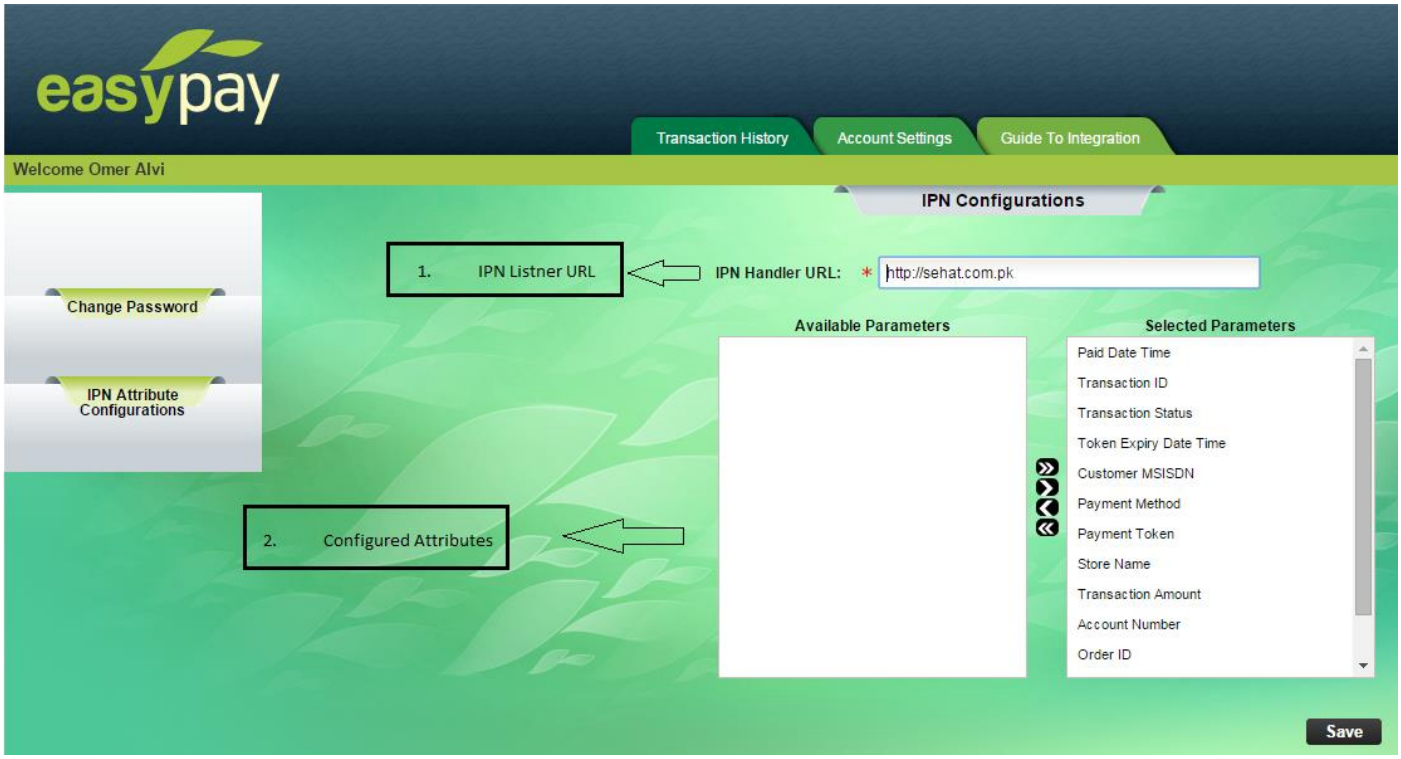
- Customer fills in the card details on Easypay forum
- Easypay invokes MCB checkout mechanism for authentication and transaction capturing/completion.
- MCB sends back the response of the transaction and Easypay updates the transaction accordingly.

3- Easypay will send a IPN REST URL to the IPN listener that is configured by the merchant via Easypay merchant portal

4- Merchant will then hit the received IPN URL and fetch the configured attribute.

4.3 What Merchant need to do?

1. Configure the listener URL on Easypay merchant portal.
2. Configure the desired IPN attributes from the Easypay merchant portal
3. Create a listener that will receive in GET request variable named 'url'.
4. The received URL would be that of REST API which will return the IPN attributes.



Following is the format for URL:

MerchantURL?url=**Rest API URL/Merchant Account ID/Order ID**
Where url = is the reserved word added by the system.

For Example:
Merchant URL = http://www.TestMerchant.com
Rest API URL = https://easypay.easypaisa.com.pk/easypay-service/rest/v1/order-status Merchant Account ID = 00001
OrderID = 998877
IPN Message:
https://www.TestMerchant.com?url= https://easypay.easypaisa.com.pk/easypay-service/rest/v1/order-status/1000223344/998877

5. Encryption Algorithm

In order to mitigate parameter tempering/modification while transfer and posting of data, merchant can encrypt the request using the hash key provided by Telenor POC. This encrypted request is sent along with the main request, which then reconciled at OPS end to detect if parameter is changed or not. The encryption can be done using following algorithm:

1. Create map of all the fields that are part of the request

```
Map<String, String> fields = new HashMap<String, String>();
fields.put("amount", "10");
fields.put("storeId", "28");
fields.put("orderRefNum", "11001");
fields.put("expiryDate", "20150101 151515");
fields.put("postBackURL", "http://localhost:9081/local/status.php");
```

2. Get the list of field name from the map created in the first step

```
List fieldNames = new ArrayList(fields.keySet());
```

3. Sort the map fields based on map key in alphabetical order

```
Collections.sort(fieldNames);
```

4. Create a string in following format:

```
amount=10&expiryDate=20150101
151515&orderRefNum=11001&postBackURL=http://localhost:9081/local/status.php&storeId=28
```

5. Use AES/ECB/PKCS5Padding algorithm to encrypt with the key and string produced in the previous step

```
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
SecretKeySpec   secretKey = new SecretKeySpec(key.getBytes(), "AES");
cipher.init(Cipher.ENCRYPT_MODE, secretKey);
encryptedValue = new String(Base64.encodeBase64(cipher.doFinal(value.getBytes())));
```

6. Appendix

1. POST 5 variables to the URL https://easypay.easypaisa.com.pk/easypay/Index.jsf with following name as follows:

    - **storeId**
    - **amount**
    - **postBackURL**
    - **orderRefNum**
    - **expiryDate**
    - **merchantHashedReq**
    - **autoRedirect**
    - **paymentMethod**

2. Receive the **auth_token** from the **postBackURL** in the previous step and post this token along with another **postBackURL** to the URL https://easypay.easypaisa.com.pk/easypay/Confirm.jsf , this time POST two following two variables:
    - **auth_token**
    - **postBackURL**

3. Retrieve transaction response on the fly from the **postBackURL** in previous step, two of the variables will be sent by Easypay:
    - **status**
    - **desc**

5.1.1 Sample Code Snippet for .NET

For the first redirection:

```
            using (var client = new HttpClient())
            {
               var values = new List<KeyValuePair<string, string>>();
               values.Add(new KeyValuePair<string, string>("storeId", "43"));
               values.Add(new KeyValuePair<string, string>("amount", "10"));
               values.Add(new KeyValuePair<string, string>("postBackURL", "http://www.my.online-
store.com/transaction/MessageHandler"));
               values.Add(new KeyValuePair<string, string>("orderRefNum", "1101"));
               values.Add(new KeyValuePair<string, string>("expiryDate", "20140606 201521"));

               values.Add(new KeyValuePair<string, string>("merchantHashedReq",
               "as;dlkjfaslk==asdfasdfasdf"));
               values.Add(new KeyValuePair<string, string>("autoRedirect", "0"));
               values.Add(new KeyValuePair<string, string>(" paymentMethod", "
               OTC_PAYMENT_METHOD"));

               var content = new FormUrlEncodedContent(values);
```

For the second redirection:

```
            using (var client = new HttpClient())
            {
               var values = new List<KeyValuePair<string, string>>();
               values.Add(new KeyValuePair<string, string>("auth_token", Request.Querystring["auth_token"]));
               values.Add(new KeyValuePair<string, string>("postBackURL", "http://www.my.online-
store.com/transaction/MessageHandler1"));
               var content = new FormUrlEncodedContent(values);
               var response = await client.PostAsync("https://easypay.easypaisa.com.pk/easypay/Confirm.jsf", content);
               var responseString = await response.Content.ReadAsStringAsync();
            }
```

5.1.2 Sample Code Snippet for Perl

For the first redirection:

```
use LWP::UserAgent;

my $ua = LWP::UserAgent->new;
my $server_endpoint = "https://easypay.easypaisa.com.pk/easypay/Index.jsf";

my $req = HTTP::Request->new(POST => $server_endpoint);
$req->header('content-type' => 'application/json');
$req->header('x-auth-token' => 'kfksj48sdfj4jd9d');

my $post_data = '{
        "storeId"           : "43",
        "amount"            : "10",
        "postBackURL"       : "http://www.my.online-store.com/transaction/MessageHandler",
        "orderRefNum"       : "1101",
        "expiryDate"        : "20140606 201521",
        “merchantHashedReq” : “asdfasdjklfja;lsdkjfa;lskdjflasdf===”,
        “autoRedirect”      : “0”,
        “paymentMethod”     : “OTC_PAYMENT_METHOD”
}';

$req->content($post_data);
```

For the second redirection:

```
use LWP::UserAgent;

my $ua = LWP::UserAgent->new;
my $server_endpoint = "https://easypay.easypaisa.com.pk/easypay/Confirm.jsf";
my $req = HTTP::Request->new(POST => $server_endpoint);
$req->header('content-type' => 'application/json');
$req->header('x-auth-token' => 'kfksj48sdfj4jd9d');
my $auth_token = $_Get['auth_token'];
my $post_data = '{
        "auth_token"        : ' + $auth_token + ',
        "postBackURL"       : "http://www.my.online-store.com/transaction/MessageHandler1"
}';

$req->content($post_data);
my $resp = $ua->request($req);
```

5.1.3 Sample Code Snippet for Ruby

For the first redirection:

```
require 'uri'
require 'net/http'
uri = URI.parse("https://easypay.easypaisa.com.pk/easypay/Index.jsf")
http = Net::HTTP.new(uri.host, uri.port)
http.use_ssl = true
http.verify_mode = OpenSSL::SSL::VERIFY_NONE

request = Net::HTTP::Post.new("/v1.1/auth")
request.add_field('Content-Type', 'application/json')
request.body = {
        'storeId'           => '43',
        'amount'            => '10',
        'postBackURL'       => 'http://www.my.online-store.com/transaction/MessageHandler',
        'orderRefNum'       => '1101',
        'expiryDate'        => '20140606 201521',
        'merchantHashedReq' => 'aasldfjlaksdjf;laksjdf;asdf--=====',
        'autoRedirect' => '0',
        'paymentMethod' => 'OTC_PAYMENT_METHOD'

}
response = http.request(request)
```

For the second redirection:

```
require 'uri'
require 'net/http'

uri = URI.parse("https://easypay.easypaisa.com.pk/easypay/Confirm.jsf")

http = Net::HTTP.new(uri.host, uri.port)
http.use_ssl = true
http.verify_mode = OpenSSL::SSL::VERIFY_NONE

cgi = CGI.new

cgi.params.each do |key, val|
  auth_token = val
end

request = Net::HTTP::Post.new("/v1.1/auth")
request.add_field('Content-Type', 'application/json')
request.body = {
        'auth_token'        => auth_token,
        'postBackURL'       => 'http://www.my.online-store.com/transaction/MessageHandler1'
}

response = http.request(request)
```

5.1.4 Sample Code Snippet for Python

For the first redirection:

```
import httplib, urllib

host = 'https://easypay.easypaisa.com.pk'
url = '/easypay/Index.jsf'
values = {
        'storeId'           : '43',
        'amount'            : '10',
        'postBackURL'       : 'http://www.my.online-store.com/transaction/MessageHandler',
        'orderRefNum'       : '1101',
        'expiryDate'        : '20140606 201521',
        'merchantHashedReq' => 'aasldfjlaksdjf;laksjdf;asdf--=====',
        'autoRedirect' => '0',
        'paymentMethod' => 'OTC_PAYMENT_METHOD'
}
headers = {
   'User-Agent': 'python',
   'Content-Type': 'application/x-www-form-urlencoded',
}
values = urllib.urlencode(values)
conn = httplib.HTTPSConnection(host)
conn.request("POST", url, values, headers)
response = conn.getresponse()

data = response.read()
```

For the second redirection:

```
import httplib, urllib
host = 'https://easypay.easypaisa.com.pk'
url = '/easypay/Confirm.jsf'
auth_token = reponse.GET['auth_token']
values = {
        'auth_token'        : auth_token,
        'postBackURL'       : 'http://www.my.online-store.com/transaction/MessageHandler1',
}
headers = {
   'User-Agent': 'python',
   'Content-Type': 'application/x-www-form-urlencoded',
}
values = urllib.urlencode(values)
conn = httplib.HTTPSConnection(host)
conn.request("POST", url, values, headers)
```

6 Sand Box Environment

For the ease of the merchants Easypay have exposed a testing environment which will be acting like a sandbox for the merchants to integrate, test and monitor their transactions. The sand box is basically an environment which simulates that of production, it is exposed over the internet. Merchants may use it to integrate for testing purposes and then while taking the project live they can just change the URLs.

Following are the public URLs for sand box:

- Merchant URL: http://202.69.8.50:9080/easypay-merchant/faces/pg/site/Login.jsf

- Plugin Index Page: http://202.69.8.50:9080/easypay/Index.jsf
- Plugin Confirmation Page: http://202.69.8.50:9080/easypay/Confirm.jsf

- WSDL URL for Open API: http://202.69.8.50:9080/easypay-service/PartnerBusinessService/META-INF/wsdl/partner/transaction/PartnerBusinessService.wsdl

- Rest API URL for getting configured IPN Attributes: http:// 202.69.8.50:9080/easypay-service/rest/v1/order-status/1000223344/998877