Network

International Payment Solutions

PAYMENT GATEWAY INTEGRATION QUERY API SAMPLE CODE ORDER STATUS C#

USER MANUAL

VERSION: 1.0.0

20-JUNE-2017

Corporate Office: Abzer DMCC, 3304-1, Mazaya Business Avenue BB1, Jumeirah Lake Towers, Dubai, UAE PO Box: 211860, +971 4 363 7757-137, info@abzer.com, www.abzer.com

Contents

Introduction

Query API Web Service is used perform transactions related queries, much easily and quickly as compared to doing so through the 'Merchant Portal'. You can send the transaction details to the payment gateway using the Web Service API containing certain parameters as defined in the individual transaction message structures.

Prerequisites

Requires .Net framework: 4.0 or above.

Visual Studio IDE

Aes.DLL version 1.0.0.0

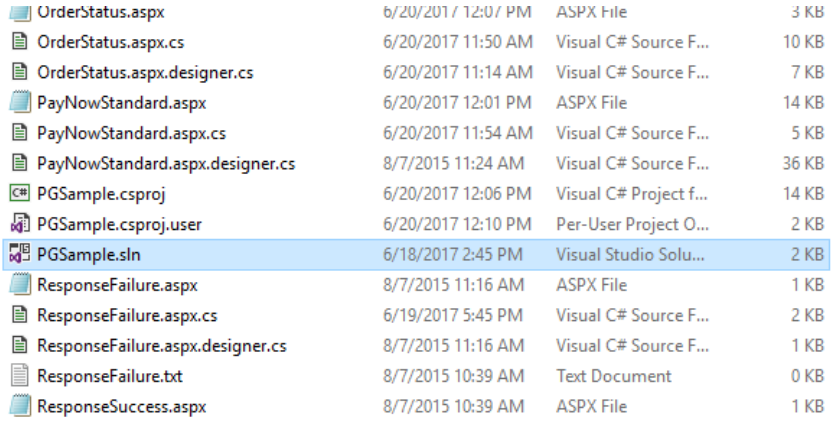
Installation

Extract the zip folder and copy the folder to your local folder. Once the copying is done, if needed you can remove the documentation file`s like (pdf`s, doc, .txt files).

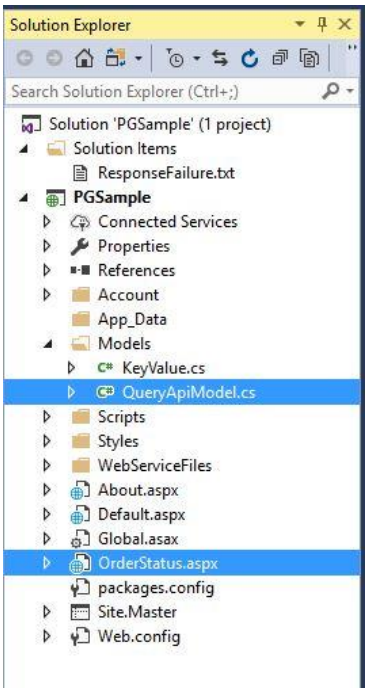
Name	Date modified	Type	Size
AES_dll	6/28/2017 12:05 PM	File folder	
PGSample	6/28/2017 12:24 PM	File folder	
InvokePMTBitMapWebService_live.xml	6/12/2017 9:05 PM	XML File	6 KB
InvokePMTBitMapWebService_test.xml	6/12/2017 9:05 PM	XML File	6 KB
Query API_for Status User Manual_Cshar...	6/28/2017 12:00 PM	Microsoft Word D...	538 KB

Settings & Executing the file

Once the installation is done please open the solution using PGSample.sln file in Visual studio IDE. Please go to Models folder inside solution and open QueryApiModel.cs file assign the following values like **merchantKey merchantId** with KEY & MID which you have. Along with this values please assign the **merchantOrderNumber** (i.e. the order number to which you are querying) to get the transaction details.



OrderStatus.aspx	6/20/2017 12:07 PM	ASPX File	3 KB
OrderStatus.aspx.cs	6/20/2017 11:50 AM	Visual C# Source F...	10 KB
OrderStatus.aspx.designer.cs	6/20/2017 11:14 AM	Visual C# Source F...	7 KB
PayNowStandard.aspx	6/20/2017 12:01 PM	ASPX File	14 KB
PayNowStandard.aspx.cs	6/20/2017 11:54 AM	Visual C# Source F...	5 KB
PayNowStandard.aspx.designer.cs	8/7/2015 11:24 AM	Visual C# Source F...	36 KB
PGSample.csproj	6/20/2017 12:06 PM	Visual C# Project f...	14 KB
PGSample.csproj.user	6/20/2017 12:10 PM	Per-User Project O...	2 KB
PGSample.sln	6/18/2017 2:45 PM	Visual Studio Solu...	2 KB
ResponseFailure.aspx	8/7/2015 11:16 AM	ASPX File	1 KB
ResponseFailure.aspx.cs	6/19/2017 5:45 PM	Visual C# Source F...	2 KB
ResponseFailure.aspx.designer.cs	8/7/2015 11:16 AM	Visual C# Source F...	1 KB
ResponseFailure.txt	8/7/2015 10:39 AM	Text Document	0 KB
ResponseSuccess.aspx	8/7/2015 10:39 AM	ASPX File	1 KB



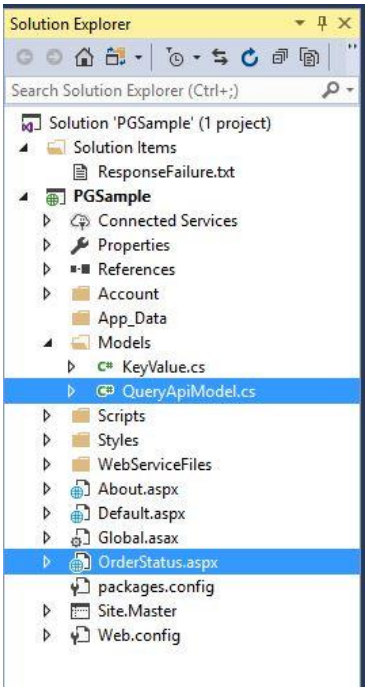
Data can be inserted in two ways.

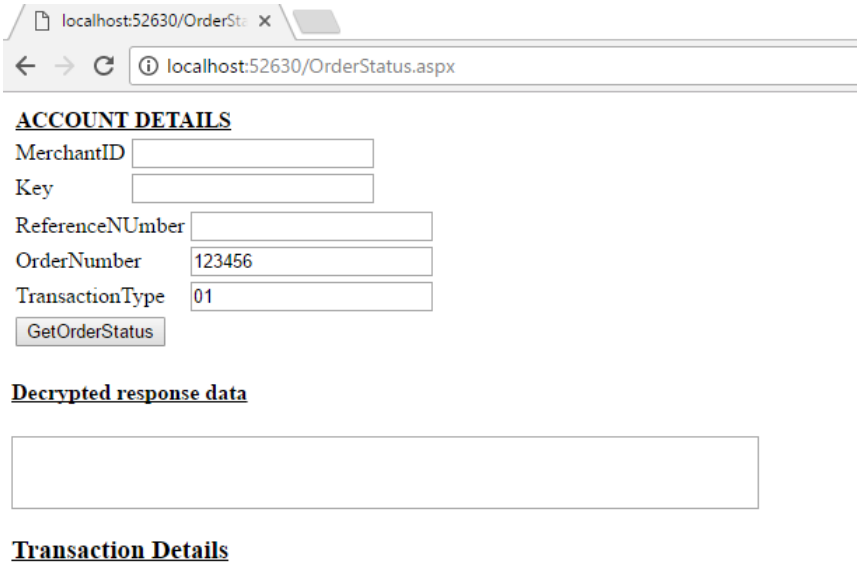
1. You can set the default values using the above constructor in QueryApiModel.cs class
2. You can use the below screen to enter values where Order number is mandatory.

```
ServiceReferencePMTBitMapWebService_test.InvokePMTBitMapWebServiceClient client
public string merchantId = "201711111111111111";
public string merchantKey = "12313124414zzzzgggg";
public string collaboratorId = "NI";
string dataBlockString = "";
```

```
public QueryApiModel()
{
    block_Existence_Indicator[0] = new Dictionary<string, bool>();
    field_Existence_Indicator_Transaction[0] = new Dictionary<string, string>();
    field_Existence_Indicator_Transaction[1] = new Dictionary<string, string>();
    field_Existence_Indicator_Transaction[2] = new Dictionary<string, string>();
    block_Existence_Indicator[0].Add("transactionDataBlock", true);// Transaction Data Block  ==> Tl
    field_Existence_Indicator_Transaction[0].Add("ReferenceID", "id12345");
    field_Existence_Indicator_Transaction[1].Add("merchantOrderNumber", "123456");
    field_Existence_Indicator_Transaction[2].Add("transactionType", "01");
}
```

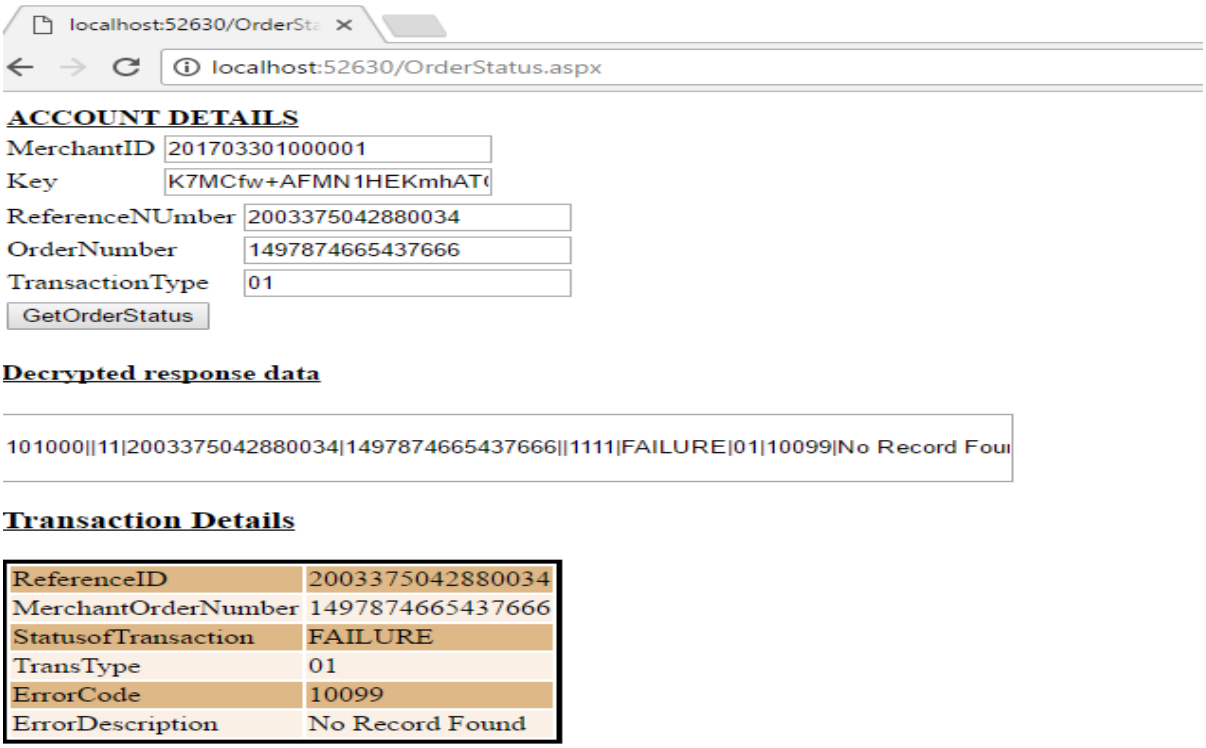
To get the below form, please set OrderStatus.aspx page as start page and run the solution





URL ex: http://localhost:port/OrderStatus.aspx

You will be getting the following results: This UI is provided for the ease of testing and displaying the decoded response in a better understandable format.



Adding .ASMX files as reference

First change the extension of the required XML file to .asmx. and add service reference.

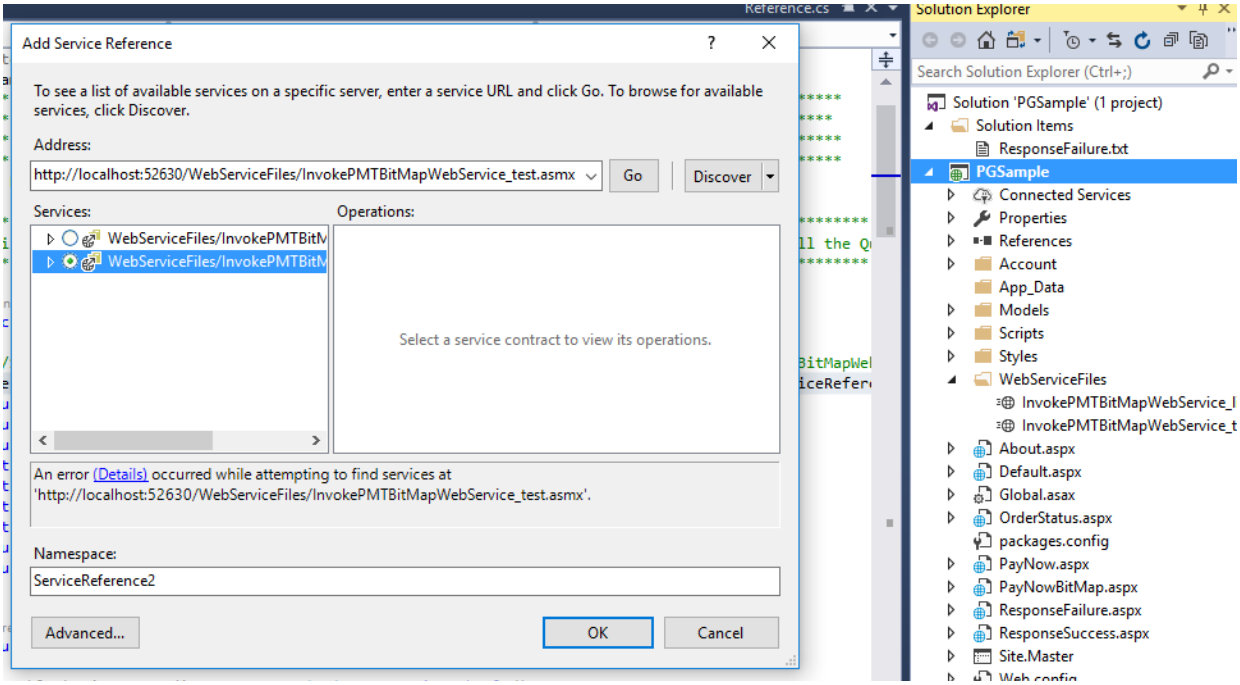
Give meaningful name space name and click ok which will create a proxy class and we can use it as below.

WSDL files
InvokePMTBitMapWebService_live.xml
InvokePMTBitMapWebService_test.xml

One file is for live environment and the other is for test environment which will handle the SOAP calls.

```
ServiceReferencePMTBitMapWebService_test.InvokePMTBitMapWebServiceClient client =
        new ServiceReferencePMTBitMapWebService_test.InvokePMTBitMapWebServiceClient();

ServiceReferencePMTBitMapWebService_live.InvokePMTBitMapWebServiceClient client =
        new ServiceReferencePMTBitMapWebService_live.InvokePMTBitMapWebServiceClient();
```



Note:- This is already added to the solution only required when not added.

Query API Web-Service Details

To get web service methods we have generated a proxy class given below which contains methods to pull result from the web service.

```
7 references
public partial class InvokePMTBitMapWebServiceClient : System.ServiceModel.ClientBase<PGSample.ServiceReferencePMTBitMapWel

    1 reference
    public InvokePMTBitMapWebServiceClient() {
    }

    0 references
    public InvokePMTBitMapWebServiceClient(string endpointConfigurationName) :
            base(endpointConfigurationName) {
    }

    0 references
    public InvokePMTBitMapWebServiceClient(string endpointConfigurationName, string remoteAddress) :
            base(endpointConfigurationName, remoteAddress) {
    }
```

You can pass the encrypted string (requestData) web service using below code where client is an object of proxy class.

string msg = client.invokeQueryAPI(requestData);

requestparameters = Merchant ID||Collaborator ID||fieldBitmap||transactionData

transactionData = FieldBitmap|ReferenceID|MerchantOrderNumber|TransactionType

requestparameters = Merchant ID||Collaborator ID||fieldBitmap||transactionData
transactionData = FieldBitmap|ReferenceID|MerchantOrderNumber|TransactionType

Example

beforeEncryptionS = 1||010|145000197

EncryptedString = wQObTItnT/1duFces4c7BS7GV/xSpi095GtDLeYwcAM=

postingString => 201607211000001||NI||wQObTItnT/1duFces4c7BS7GV/xSpi095GtDLeYwcAM=

Data will be present in the string only if the corresponding Field existence block is present as 1. If it is zero then that data will not be present. Below is an example of calculating the data

```
1 reference
public string Calculate()
{

    foreach (KeyValuePair<string, bool> data in blockExistenceIndicator)
    {
        switch (data.Key)
        {
            case "transactionDataBlock":
                blockExistenceIndicatorData += data.Value ? 1 : 0;
                if (data.Value)
                {
                    CalculateFieldExistenceIndicatorTransaction();
                    finalData += fieldExistenceIndicatorTransactionData + "|" + dataBlockString;
                    dataBlockString = "";
                }

                break;
        }
    }
    finalData = finalData.Remove(finalData.Length - 1);
    finalData = blockExistenceIndicatorData + "||" + finalData;
    return finalData;
}
1 reference
```

To simply we have customized keys into string arrays and are combined with corresponding data from response parameters to generate a keyValue pair

```
string[] referenceKeys = { "ReferenceID", "MerchantOrderNumber" };
string[] currencyKeys = { "Amount", "Currency" };
string[] statusKeys = { "StatusofTransaction", "TransType", "ErrorCode", "ErrorDescription" };
string[] merchantKeys = { "PayModeType", "CardType", "CardEnrollmentResponse", "ECI_Values", "Card Number", "Auth Code" };
string[] fraudKeys = { "FraudDecision", "FraudReason" };
string[] dccKeys = { "DCC_Converted", "DCC_ConvertedAmount", "DCC_ConvertedCurrency", "DCC_Exchange Rate", "DCC_Margin_Rate" };
```

Eg: code

```
public void DecodeFields(string data, string[] keyArray)
    {
      List<KeyValue> valuesDecrypted = new List<KeyValue>();
      String fieldExistenceBlock = data.Substring(0, data.IndexOf("|", 0));
      char[] charArr = fieldExistenceBlock.ToCharArray();
      string[] fieldData;
      fieldData = data.Substring(data.IndexOf("|", 0) + 1).Split(new[] { "|" }, StringSplitOptions.None);
      int j = 0;
      for (int i = 0; i < charArr.Length; i++)
      {  if (charArr[i].ToString().Equals("1"))
        {
          valuesDecrypted.Add(new KeyValue(keyArray[i], fieldData[j]));
          j++;
        }
        else
          valuesDecrypted.Add(new KeyValue(keyArray[i], ""));
        }
```