Network International Payment gateway integration

This Document include all the information needed for building an integration (Direct Http post request & Transactions query webservice) between Salesforce and Network International Payment Gateway

A. What do you need from NI side ?

1. Test environment
2. Test Merchant ID
3. Test Encryption Key
4. Test URL - currently: https://uat-neo.network.ae/direcpay/secure/PaymentTxnServle
5. Test WSDL Endpoint: https://uat-neo.network.ae:443/direcpay/secure/InvokePMTBitMapWebService
6. WSDL file : you can find the current on here
7. Setting there programming language to PHP
8. There latest developer guide, currently you can find it here

B. What to build from you side ?

1. Custom MetaData Type : **Payment Gateway Setting** - this object will be used to save the data related to test and live integration

Field Name	**Description**
Merchant Id	Text - Provided by NI
collaborator Id	Text - Currently: NI
Encryption Key	Text - Provided by NI
Initialization Vector	Text - Currently: 0123456789abcdef
Payment Currency	Text - AED
Transaction Mode	Text - INTERNET
Transaction Type	Text - 01
Return Page	Text - The name of the page your user will be redirect to after payment
Active	Checkbox - indicator of which record to use in

	integration
URL	Text - URL used for http post requests
WSDL Endpoint	Text - URL used in Webservices calls

- You will be creating two records under “Payment Gateway Setting” one for test environment integration and another for live integration can be easily switched between by checking and unchecking the “Active” field.

2. For Direct http post request do the following
    - Build your payment page with a “PageReference” pay button
    - Build a method to return URL to generate the “PageReference”
    - Read the developer guide and build your request parameters in the way you need
    - Here is a working sample code that you can directly use if it meet your requirements

```
public static string doPayment(case caseObj,decimal totalAmount){

string returnURL = '';
string parameters = '';

Payment_Gateway_Setting__mdt gateWayParams = [select Id, Active__c, collaborator_Id__c,
Encryption_Key__c, Initialization_Vector__c,  Merchant_Id__c, Payment_Currency__c,
Return_Page__c, Public_Return_Page__c, Transaction_Mode__c, Transaction_Type__c,
URL__c
 from Payment_Gateway_Setting__mdt where Active__c = true];

string testUrl = gateWayParams.URL__c;
string MerchantID = gateWayParams.Merchant_Id__c;
string EncryptionKey = gateWayParams.Encryption_Key__c;
string collaboratorId = gateWayParams.collaborator_Id__c;
string orderNumber = caseObj.Id;
string Iv = gateWayParams.Initialization_Vector__c;
string returnPage = gateWayParams.Return_Page__c ;
string successUrl =  URL.getSalesforceBaseUrl().toExternalForm() + '/apex/'+ returnPage;
string failureUrl =  URL.getSalesforceBaseUrl().toExternalForm() + '/apex/'+ returnPage;
string TransactionMode = gateWayParams.Transaction_Mode__c;
string TransactionType = gateWayParams.Transaction_Type__c;
string paymentCurrency = gateWayParams.Payment_Currency__c;

parameters = '1100100||11111111|' +
orderNumber+'|'+totalAmount+'|'+successUrl+'|'+failureUrl+'|'+TransactionMode+'|'+'CC'+'|'+Tra
nsactionType+'|'+paymentCurrency+'||'

+'1111111111000|'+accountObj.Authorize_Manager__r.firstName+'|'+accountObj.Authorize_M
anager__r.lastName+ address +accountObj.Authorize_Manager__r.PERSONEMAIL+'|'+
accountObj.Authorize_Manager__r.PHONE+'||'

+'1000000000|'+accountObj.External_License_Number__c;

Blob data = Blob.valueOf(parameters);
Blob Key = Encodingutil.base64Decode(EncryptionKey);
Blob networkIv = Blob.valueOf(Iv);
Blob encrypted = Crypto.encrypt('AES256', key, networkIv ,data);
String b64=EncodingUtil.base64Encode(encrypted);
string paramData = MerchantID+'||'+collaboratorId+'||'+b64;
returnURL = testUrl+'?requestParameter='+EncodingUtil.urlEncode(paramData,'UTF-8');
return returnURL;
}
```

- Now build the Return page where you can get the return response and do the needed action and shown the success or failure message based on the response parameters
- Here is a sample code of a payment page controller you can use directly if it meet your requirements

```
public class Payments_ctrl {

    public case caseObj {get;set;}
    public string responseParameter {get;set;}
    public string returnFlag {get;set;}
    public string returnMessage {get;set;}
    public string caseId {get;set;}
    public decimal amountPaid {get;set;}
    public string returnCode {get;set;}
    public string tryAgainUrl {get;set;}
    public string referenceNumber {get;set;}
    public decimal amountToPay {get;set;}
    public boolean paymentDone {get;set;}


    public Payments_ctrl (){
      Payment_Gateway_Setting__mdt gateWayParams = [select Id, Active__c,
collaborator_Id__c, Encryption_Key__c, Initialization_Vector__c,
                                          Merchant_Id__c, Payment_Currency__c, Return_Page__c,
Transaction_Mode__c, Transaction_Type__c, URL__c
                                          from Payment_Gateway_Setting__mdt where Active__c =
true];

      caseId = '';
      string EncryptionKey = gateWayParams.Encryption_Key__c;
      string Iv = gateWayParams.Initialization_Vector__c;
      Blob Key = Encodingutil.base64Decode(EncryptionKey);
      Blob networkIv = Blob.valueOf(Iv);

      Map<String, String> getParameters = ApexPages.currentPage().getParameters();
      string merchantId = getParameters.get('merchantId');
      responseParameter = getParameters.get('responseParameter');
      string responseparams = getParameters.get('responseparams');
      string txnErrMsg = getParameters.get('txnErrMsg');

      if(responseParameter != null){
         string data = responseParameter.split('\\|\\|')[1];
         blob decodedData = EncodingUtil.base64Decode(data);
         Blob decryptedData = Crypto.decrypt('AES256', key, networkIv, decodedData);
         string stringDecryptedData = decryptedData.toString();

         // 1111010||
         // transactionResponse       111111|Test-001|AED|1000.00|CC|VISA|01||
         // transactionResponseInfo   1111101|2001930499445426|01-Aug-2018 11:13:33
AM|ENROLLED|Fully Secure|5331076308466201704008|831000||
         // transactionResponseStatus 111|SUCCESS|00000|No Error.||
         // returnData               1000000000|0060||
         // farud                 ----
         // DCCconverted             10000|NO
         // Additional               ----


         list<string> DecryptedDataList = stringDecryptedData.split('\\|\\|');
         System.Debug('DecryptedDataList ' + DecryptedDataList);

         boolean transactionResponse, transactionResponseInfo, transactionResponseStatus,
               returnData, farud, DCCconverted, Additional;
         list<string> dataKey = new list<string>();
         map<integer,string> dataMap = new map<integer,string>();
```

```
        dataKey = DecryptedDataList.get(0).split('');
        if(dataKey.get(0) == '1') transactionResponse = true;        else transactionResponse =
false;
        if(dataKey.get(1) == '1') transactionResponseInfo = true;     else
transactionResponseInfo = false;
        if(dataKey.get(2) == '1') transactionResponseStatus = true;   else
transactionResponseStatus = false;
        if(dataKey.get(3) == '1') returnData = true;                  else returnData = false;
        if(dataKey.get(4) == '1') farud = true;                       else farud = false;
        if(dataKey.get(5) == '1') DCCconverted = true;                else DCCconverted = false;
        if(dataKey.get(6) == '1') Additional = true;                  else Additional = false;

        list<integer> dataNumber = new list<integer>{1,2,3,4,5,6,7};
        integer emptyNumber = 1;
        for(integer item : dataNumber){
           if(item == 1){
              dataMap.put(item,decryptedDataList.get(emptyNumber));
              emptyNumber++;
           }
           else if(item == 2){
              if(transactionResponseInfo){
                 dataMap.put(item,decryptedDataList.get(emptyNumber));
                 emptyNumber++;
              }else dataMap.put(item,'');
           }
           else if(item == 3){
              if(transactionResponseStatus){
                 dataMap.put(item,decryptedDataList.get(emptyNumber));
                 emptyNumber++;
              }else dataMap.put(item,'');
           }
           else if(item == 4){
              if(returnData){
                 dataMap.put(item,decryptedDataList.get(emptyNumber));
                 emptyNumber++;
              }else dataMap.put(item,'');
           }
           else if(item == 5){
              if(farud){
                 dataMap.put(item,decryptedDataList.get(emptyNumber));
                 emptyNumber++;
              }else dataMap.put(item,'');
           }
           else if(item == 6){
```

```
          if(DCCconverted){
            dataMap.put(item,decryptedDataList.get(emptyNumber));
            emptyNumber++;
          }else dataMap.put(item,'');
        }
        else if(item == 7){
          if(Additional){
            dataMap.put(item,decryptedDataList.get(emptyNumber));
            emptyNumber++;
          }else dataMap.put(item,'');
        }
     }

     for(integer item : dataMap.keyset()){
        system.debug('item ' + item +' : '+ dataMap.get(item));
     }

     caseId = dataMap.get(1).split('\\|')[1];
//   caseId= caseId.substring(0, caseId.length() - 1); for testing only
     system.debug('caseId '+ caseId);

     returnFlag = dataMap.get(3).split('\\|')[1];
     system.debug('returnFlag '+ returnFlag); //SUCCESS FAILURE PENDING

     if(returnFlag == 'SUCCESS')
       referenceNumber = dataMap.get(2).split('\\|')[1];
     system.debug('referenceNumber '+ referenceNumber );

     amountPaid = decimal.valueOf(dataMap.get(1).split('\\|')[3]);

     returnCode = dataMap.get(3).split('\\|')[2];
     system.debug('returnCode '+ returnCode);

     returnMessage = dataMap.get(3).split('\\|')[3];
     system.debug('returnMessage '+ returnMessage);

     string accountLicenseNumber = dataMap.get(4).split('\\|')[1];
     system.debug('accountLicenseNumber '+ accountLicenseNumber);

}
             if(returnFlag == 'SUCCESS'){
               //do some actions
             }else{
               //do other actions
```

```
                  }
   }
}
```

3. For Webservice Transactions queries do the following
    - First generate the apex class from WSDL file you already have
    - Build a class for bulk queries webservice call
    - Here is a sample code you can you directly if it meet your requirements

```
global class NetworkInternationalQuery_Class {

    public static string testEncryptedResponse;  // for test class use

    Webservice static void NetworkInternationalQuery(list<case> caseList){
        system.debug('NetworkInternationalQuery start >> caseList '+ caseList);
            //NetworkInternationalQuery_WSDL is the apex class created from the WSDL file we
have from NI
        NetworkInternationalQuery_WSDL.InvokePMTBitMapWebServicePort  newCLass =
new NetworkInternationalQuery_WSDL.InvokePMTBitMapWebServicePort();

        Payment_Gateway_Setting__mdt gateWayParams = [select Id, Active__c,
collaborator_Id__c, Encryption_Key__c, Initialization_Vector__c,
                                        Merchant_Id__c, Payment_Currency__c, Return_Page__c,
Public_Return_Page__c, Transaction_Mode__c, Transaction_Type__c, URL__c
                                        from Payment_Gateway_Setting__mdt where Active__c =
true];

        string MerchantID = gateWayParams.Merchant_Id__c;
        string EncryptionKey = gateWayParams.Encryption_Key__c;
        string Iv = gateWayParams.Initialization_Vector__c;
        string collaboratorId = gateWayParams.collaborator_Id__c;
        string TransactionType = gateWayParams.Transaction_Type__c;

            //lets say you are quering list of cases to get the transactions results
       //loop the cases and get the transactions details
        for(case caseObj : caseList){
           string caseId = caseObj.Id;

           string parameters ='1||011|' + caseId +'|'+ TransactionType;

           Blob data = Blob.valueOf(parameters);
system.debug('data ' + data);
```

```
             Blob Key = Encodingutil.base64Decode(EncryptionKey);
system.debug('key ' + key);
             Blob networkIv = Blob.valueOf(Iv);                         system.debug('networkIv
' + networkIv);
             Blob encrypted = Crypto.encrypt('AES256', key, networkIv ,data);
             String b64=EncodingUtil.base64Encode(encrypted);
             string paramData = MerchantID+'||'+collaboratorId+'||'+b64;

             string encryptedResponse = newCLass.invokeQueryAPI(paramData);
             if(Test.isRunningTest())
               encryptedResponse = testEncryptedResponse;
             system.debug('encryptedResponse '+ encryptedResponse);

             blob decodedData = EncodingUtil.base64Decode(encryptedResponse);
             Blob decryptedData = Crypto.decrypt('AES256', key, networkIv, decodedData);
             string response = decryptedData.toString();
             System.Debug('response ' +  response );

                // Sample results
          //  111111||11|2001983452990392|5009E0000096noiQAA ||11|90|AED
||1011|Settled|00000|No Error.||111111|CC|VISA|ENROLLED|Fully
Secure|xxxxxxxxxxxx1111|831000||11|NA|NA||10000|NO
          //  111111||11|2001348990207496|5009E0000096ouaQAA ||11|4110.00|AED
||1011|FAILURE|20002|General decline of the card. No other information was provided by the
issuing bank.||111110|CC|VISA|ENROLLED|Fully
Secure|xxxxxxxxxxxx1111||11|NA|NA||10000|NO
          //  111111||11|2001672280111794|5009E0000096nh3QAA ||11|575|AED
||1011|Settled|00000|No Error.||111111|CC|VISA|ENROLLED|Fully
Secure|xxxxxxxxxxxx1111|831000||11|NA|NA||10000|NO

            list<string> DecryptedDataList = response.split('\\|\\|');
            System.Debug('DecryptedDataList ' + DecryptedDataList);

            boolean transactionData, amountData, statusData, paydetailsData, fraudData,
DCCData;
            list<string> dataKey = new list<string>();
            map<integer,string> dataMap = new map<integer,string>();

            dataKey = DecryptedDataList.get(0).split('');
            if(dataKey.get(0) == '1') transactionData = true;        else transactionData = false;
            if(dataKey.get(1) == '1') amountData = true;             else amountData = false;
            if(dataKey.get(2) == '1') statusData = true;             else statusData = false;
            if(dataKey.get(3) == '1') paydetailsData = true;         else paydetailsData = false;
            if(dataKey.get(4) == '1') fraudData = true;              else fraudData = false;
            if(dataKey.get(5) == '1') DCCData = true;            else DCCData = false;

            list<integer> dataNumber = new list<integer>{1,2,3,4,5,6};
            integer emptyNumber = 1;
            for(integer item : dataNumber){
               if(item == 1){
                 if(transactionData){
                   dataMap.put(item,decryptedDataList.get(emptyNumber));
                   emptyNumber++;
                  }else dataMap.put(item,'');
               }
               else if(item == 2){
                 if(amountData){
                   dataMap.put(item,decryptedDataList.get(emptyNumber));
                   emptyNumber++;
                 }else dataMap.put(item,'');
               }
               else if(item == 3){
                 if(statusData){
                   dataMap.put(item,decryptedDataList.get(emptyNumber));
                   emptyNumber++;
                 }else dataMap.put(item,'');
               }
               else if(item == 4){
                 if(paydetailsData){
                   dataMap.put(item,decryptedDataList.get(emptyNumber));
                   emptyNumber++;
                 }else dataMap.put(item,'');
               }
               else if(item == 5){
                 if(fraudData){
                   dataMap.put(item,decryptedDataList.get(emptyNumber));
                   emptyNumber++;
                 }else dataMap.put(item,'');
               }
               else if(item == 6){
                 if(DCCData){
                   dataMap.put(item,decryptedDataList.get(emptyNumber));
                   emptyNumber++;
                 }else dataMap.put(item,'');
               }
            }

            for(integer item : dataMap.keyset()){
```

```
                system.debug('item ' + item +' : '+ dataMap.get(item));
            }

            string returnFlag = dataMap.get(3).split('\\|')[1];
            system.debug('returnFlag '+ returnFlag); //SUCCESS FAILURE Settled

            string returnCode = dataMap.get(3).split('\\|')[3];
            system.debug('returnCode '+ returnCode);

            string returnMessage = dataMap.get(3).split('\\|')[4];
            system.debug('returnMessage '+ returnMessage);

            decimal amountPaid = 0;
            string referenceNumber = '';
            if(returnFlag == 'Settled' || returnFlag == 'SUCCESS'){
               amountPaid = decimal.valueOf(dataMap.get(2).split('\\|')[1]);
                 system.debug('amountPaid '+ amountPaid);
               referenceNumber = dataMap.get(1).split('\\|')[1];
                 system.debug('referenceNumber '+ referenceNumber );
            }

            try{
               if(returnFlag == 'Settled' || returnFlag == 'SUCCESS'){
                      //do some actions
                  }else{
                     //do other actions
                  }
              }catch(Exception e){
                  system.debug('MyException >>> ERROR AT LINE : ' + e.getLineNumber() + '
CAUSE BY : ' + e.getStackTraceString() + '  -  MESSAGE : ' + e.getMessage() );
                  //create service log for the fail payment
                  Service_Log__c serviceLogObj = new Service_Log__c();
                  if(serviceLogMap.containsKey(caseId))
                     serviceLogObj = serviceLogMap.get(caseId);
                  serviceLogObj.where__c = 'NetworkInternationalQuery_Class  >>
NetworkInternationalQuery ';
                  serviceLogObj.error_details__c = 'MyException >>> ERROR AT LINE : ' +
e.getLineNumber() + ' CAUSE BY : ' + e.getStackTraceString() + '  -  MESSAGE : ' +
e.getMessage();
                  serviceLogObj.Case__c = caseId ;
                  serviceLogList.add(serviceLogObj);
              }

          }
```

```
    }
}
```

- Then you can create a batch to call this class and a schedulable class to run the batch in a certain times

C. Finally after contacting NI team with the result of your testing they will activate the live environment and provide

1. Live Merchant ID
2. Live Encryption Key
3. Live URL - currently: https://neo.network.ae/direcpay/secure/PaymentTxnServle
4. Live WSDL Endpoint: - currently:

https://neo.network.ae:443/direcpay/secure/InvokePMTBitMapWebService

Best Regards..