NTT DaTa
Payment Services

Info type: Confidential
Company: NTT Data Payment Services India Limited
Info. owner: Product Management Group

Transaction API (Non-Seamless)

NTT Data Payment services Ltd.
www.nttdatapay.com

CONFIDENTIALITY DISCLAIMER

A. Document Information

Document Attributes	Information
Document Name	Transaction API
Document Version	1.01
Owner	PMG
Author	Nikhil Dayma
Approved	Pavan Nikumbh

B. Revision History

This chart contains a history of this document's revisions.

Version	Author	Description of Version	Date	Reviewed By
1.0	Nikhil Dayma	Transaction API (Non-Seamless)	07-04-2022	Pavan Nikumbh
1.01	Aviral Tripathi	Response Signature	23-09-2022	Pavan Nikumbh

CONTENTS

1. Introduction

This document will provide overview of Transaction API process flow. Transaction API is an API in which NTT Data payment services system receives request from merchant with transaction details provided to redirect end-user to the NTT Data payment page for selecting payment options and once the end-user completes the transaction sends the status of the transaction to the merchant so that he can take action accordingly.

2. General process flow:

The general process flow of the transaction API is as follows:

- The end customer after purchasing product/service on merchant websites opts for payment, he selects the payment option, and the API gets called.
- In case of non-seamless transaction, the merchant redirects the user to NTT Data payment service page whereas in case of seamless transaction he calls our own API.
- Once the transaction is completed by the end-user the NTT data payment service sends the transaction status to merchant of the payment made

3. Pre-requisite:

Merchant's IP and domain URL need to be whitelisted and configured at NTT data payment system. Also, the NTT Data's payment server needs to be whitelisted at their end.

4. Request posting parameters:

The parameters that are to be posted fall under 4 categories: Header, Merchant validation, Account validation, Channel information and customer information. These categories have been highlighted as bold in sample request.

Specifications of the parameters of API Request:

Parameter Name	Data Type	Length	Static/Dynamic	Prefilled [Static]	Mandatory/Optional/CM	Description
version	string	7	Static	OTSv1.1	M	API Version
payMode	string	5	Static	SL	M	API Mode
channel	string	10	Static	ECOMM	M	Payment Channel
Api	string	10	Static	SALE	M	API Type
platform	string	10	Static	WEB	M	Supported Platform
Stage	string	1	Static	1	M	Transaction Stage
merchId	int	15	Static	Provided by NTT Data payment services	M	Merchant Unique Reference Number shared by NTT Data payment services
Password	string	45	Static	Provided by NTT Data payment services	M	Password shared by NTT Data payment services
merchantTxnId	string	50	Dynamic	-	M	Txn Id pertaining to merchant
mccCode	string	7	Static	Provided by NTT Data payment services	M	Master Category Code provided by NTT Data payment services
merchType	string	15	Static	"M"- Broker Merchant "R" - Regular Merchant	M	Type of Merchant
merchTxnDate	datetime	Datetime	Dynamic	-	M	Transaction Date
prodName	string	50	Static	Pennydrop	M	Name of the Product

prodAmount	double	12, 2	Dynamic	-	M	Penny Drop Amount
amount	double	12, 2	Dynamic	-	M	Amount
surchargeAmount	double	12, 2	Static	NA	O	Not to be passed
totalAmount	double	12, 2	Dynamic	-	M	Total Amount
custAccNo	string	45	Dynamic	-	M	Bank Account Number of end user
custAccIfsc	string	45	Dynamic	-	M	IFSC of end user
custEmail	email	100	Dynamic	-	M	
clientCode	string	45	Dynamic	-	M	Unique Client Code passed by Merchant
txnCurrency	string	5	Static	INR	M	Curreny used for transaction
remarks	string	20	Dynamic	-	M	Remark passed by Merchant
Signature	string	256	Dynamic	-	M	Signature generated
subChannel	string	45	Static	PD	M	Sub Channel Id
custFirstName	string	80	Dynamic	-	M	First Name of User
custLastName	string	80	Dynamic	-	M	Last Name of User
custEmail	email	100	Dynamic	-	M	Email of end user
custMobile	string	20	Dynamic	-	M	Mobile of End User
custAddr1	string	100	Dynamic	-	M	Address of End User
custAddr2	string	100	Dynamic	-	O	Address of End User
custCountry	string	45	Dynamic	-	O	Country of End User
custCity	string	30	Dynamic	-	O	City of End User
custState	string	30	Dynamic	-	O	State of End User
custZipCode	Integer	10	Dynamic	-	M	ZipCode of End User

Sample Decrypted Response (Open Data):

```
{
  "payInstrument":{
    "headDetails":{
      "title":"ATOM",
      "payMode":"RD",
      "channel":"ECOMM",
      "api":"SALE",
      "platform":"WEB",
      "stage":1
    },
    "merchDetails":{
      "merchId":9135,
      "password":"Test@123",
      "merchTxnId":"OTS77",
      "mccCode":7011,
      "merchType":"R",
      "merchTxnDate":"2022-05-02 18:43:44"
    },
    "payDetails":{
      "prodDetails":[
        {
          "prodName":"Mangeshtest",
          "prodAmount":3.00
        }
      ],
      "amount":3.00,
      "surchargeAmount":0.00,
      "totalAmount":3.00,
      "custAccNo":"123456789",
      "clientCode":"32454",
      "txnCurrency":"INR",
      "remarks":"OTS",

"signature":"d6f7a7884c0fef8e84f70eb0e1638970e50714ba56a7f00fb5310bbd1e7a0011451dece7c501
76a63bbc17b18f42dd7c47a3be39e1647b7581bf8a74a14e6336",
      "standInstr":false
    },
    "responseUrls":{
      "returnUrl":"https://pgtest.atomtech.in/paynetzclient/ResponseParam.jsp"
    }
  }
}
```

Note:

a) 'custAccNo' is mandatory in case of broker transactions using NetBanking or UPI Collect modes.

b) In case of prod details, merchant can specify multiple products if he has configured them and multiple amounts across those products, but the total amount across all the products should be equal to that paid by the end-user.

various products across which the payment is to be distributed. An example of the same is as follows:

```
"prodDetails": [
{
"prodName": "Mangeshtest",
"prodAmount": 11
},
{
"prodName": "DHARAM_TEST",
"prodAmount": 12
}
],
```

One aspect to be considered here is sum of all prodAmount should equal amount to be paid

Sample encrypted request format is as follows:

Key for encryption is as follows:

MerchId	**encResKey**
9135	7813E3E5E93548B096675AC27FE2C850

Parameter Name	**Mandatory**	**Data Type & Max Length**	**Sample Value**	**Content/ Remarks**
encResKey	Mandatory	String(32)	7813E3E5E93548B096675AC27FE2C850	Need to be saved per MID as shared by ATOM

Transaction Sample Request (Encrypted Request):

```
https://caller.atomtech.in/ots/payment/txn?merchId=9135&encData=70191D14B11D5F62EA63D4D681
C42478599654D8EDEB00CE19AE51822A3F9375F201AEE786FAF86EF04ADED336A69A53809C946A2BC3
A9DDDED65674A9D504D2EEF83F95269A7FA945839A32FF9949D098909B8F756FC6AEF6C7FB432108E4
86157C33C67902B43BC245B4A143F9B5B884F90F56CC8B74803F80A392E5555C783C166E6B9691DFF3
```

D178950238E447C8E51B7BCC2783512A162F548C8EE136DBC03C529E338C807CCD2ADAB8E81F29ECFB
60157068295DA13007BA8E457DB74D637B872AD3AA8CDF2AE9FC6332DDB90EA98DCAE37ACCDC0499
3340D15437568F3D8B9017C1B0A0AF5181610F2EE80D4EEA4CB5CDD3FB363120C4C9E7C5761A79B5C0
FE990677CF8BB69978851DC2A775867367553608 78D09AE7F584572C3E2247DC7C7CA770C7F68AA96B
CE5AB721D32B1F6C3601678BAE2CE96A64BDC7135817D9468E09FBAC248D4BE90FC9F5D36F2BD1A85
EF8F7D6CF7C74443590669C1E6855D48EB7B38D461DC9D14A271B6980659CD0763FC8F5F620287B6A7
70126D2E57DADC1FCAB583463748A4895BCC8730AC3D47432E487A7249F1D9D7E1802D2FD365EAE75
7C5BFA221D7F4A693A022C87F241E5D0B681D3C00815D9F84C910F365D61338FC1D7821038F8F7624E
2A5DF20ED37DF072E77F208B3D82B62E710199A427314009EF3843D5FEA3861D77270458C113510631
DF21D11B3B0201E239F48286CD62D1E4626300C86EE996AEC216903A49812558C820278AD2E0A36379
7356B22BFB1CAB0DA56AC3E778FA5F29BED0F6081F388A77978964DF21B35E8137064E3E525D7D13F1
4F5C879557CCBC9DA8376DA1746436967C4ED4BA4CBB3D57F0F3273414BF611284FA398679DF925D7
B42EF43F400361E4F76F7EBCA68B6F3C5426F387EA1E3A833C22CDAC40D2E5DB58EE1B12CEBD42FF87
D81FEC60DBE6A6F0DA144B542B7175144FAC5C69C3702368D487E17C8820961AC80B6C85388E3255A7
072B7DC1B524644E8BAFFBE573912EA3299346BB4DA043F7303CC7CB8457AECD5C9720D2F59EA15552
73F347DF23E9E113DE5BC6DD0611EC6ED72130F26B55EE67917A5DE822EC58198AD1B435268A24A9AE
ADCCF6A40909690C392D66C58C46E600089A73D602DB975B386D3C85F3A24CB03F25BA5500E2F76D0
5996B53846DD23F06B64C21D77679698432B2DBB29BFD527E29F008A045A1A8110AB7AD1BE7E59554
765CAFCD0ACE07A1E5C4F2F2965D54D4BC8CA2C692FA0F5D463CDEFE3924895CF8C786716BE844DCE7
F94DE20A10AA9CCE07D81BE6D0C372DFEC5E3AB0D74BB670ABF889244A00A9A29888C87A2823A1CF4
A1E58A9608B722283F3A12E8EAF66FC4A15F17DE683A7586D8508BEB30364A0957635759B45FD77F5B
6EECC8162A634A5EE344CC8CD36395F730B4EC3970CA81B7FCF35EE470E1EE7F54688DA89DB55F2ED7
5B980778ACD8C770E00AB79354A776248BE831F3E7CED81EC249927E8B335BE494BF0C70E1913374E4
2DA62BB2720F8407C6B0DA3FEA069D64E98086000949DDD18CE31047FE905BA9295049F56023BC271C
9DCC77866E6FB3F5E95400C803734E54EE0F3CEF

5. Response parameters for Merchant from NTT data payment services:

The key for decryption is as follows:

MerchId	encResKey
9135	7813E3E5E93548B096675AC27FE2C850

Parameter Name	Mandatory	Data Type & Max Length	Sample Value	Content/ Remarks
encResKey	Mandatory	String(32)	7813E3E5E93548B096675AC27FE2C850	Need to be saved per MID as shared by ATOM

Sample Encrypted Response Data:

```
merchId=9135&encData=C702DD1A871AEEDB89D4E23C1100E2B7D225881F7B7A3B46C75A2C167D651
1C2FFA94E5B8B4D467CF3850B4B2EC5ECF1C0FA2EB50A6534400B4BFF92AA3D8E8D78231F9DBF4F747
7AE69E87A05C8C3ACD7FF6DBD655BAAF561496A5447884EF426DF65275AC22B7F998C0213C45B5C8C
DF919956188FDB8634741697D187EEEB0C340AFE303D04CD5CAADFA4A52372801CB5C55D7A4CE9B9F
DCBE95AD8AACCB52CE94738495572D7C7A40ECE6A8F9B0FF88CF1413EC3CA4344B546A49B859492C6
6D470BCFD418B40154A35D729DE1A5A6CA05AC2973E6A965C39579D490DBB4810F57DA38B84D6D94
55936009733B1B91BBAC32D1BB9E69E60AB4617D3D7CCB0751E7FCF612FA1115EB02F4579ADE096F7F
EFC13DDC602C89FC1F7F68CCE1EEB51487DA971135ABE040622E0F4CCBBEDABB6FBCD61B850EE03D81
35E73B9DAA5DC7DC1E0C37ABE55614320C65E404BA3AC93D5220837C2118D2E42ABBF41472CCE9D22
270F93B6B1197AC7FD903B30A254262FBF33A602171EF1524CF10ECF604540AF1E8EB283355C5A29EB9
3ABBBAF8217E49CC6CFF92B376BDE3F4B4DEFB4BEC358DC1B3FA58D1A513481FD42DAF6CA66441E57
C0331A127B3C00E81D71BDEA349E37822161B5B05899396AC4A346D4DE44F442E2B675B9321A487BD
D930C2E36180D9FF1660E9D6724967148FFEC127AFDE7651EA7B75DE9A681169877663E3ED110E0EB4
63D5A42A941B9DFF7B58736B6848EB8A3E07C6AFCBD8EA73045AE8FC4AEE44F806CC25A8AB3C7DA14
337DD51A82BAEACFF7AC3A9C4190B98D5D56F8FF9458358815979ABBC6947AF6B94C6748FE2E966E2A
3F27B77D67534EB2E27F5E3A8024D6CFE6260721EB66FD2018BA317CA1C2712326A0E72D06C4975C08
DCB1D76DB45F720D9528604E468D76A697AD3A85FFAD4212E69DC173C68B1310C007B37D23CA79C15
EF8CA865576852A476DEFD08A41CFACB863B5E1A3AE27BAA1F6097AD634889C9DD2A6CF3BE85E0DAF
ECA243700D03742E741DC67456F974E58693DC098A61EF07E5DF306913E0B70A76C705A8DE9F003ECF
4F4EC4E0B5D80456A8E65B54B31349DAFDFD3D3784DD4C6298C6C4B48FEBBA33D344D25F9474052AF
598195880A310E7C7143849DF7D6067517F10B6438743D869220577DE17537AAC02D016633A9F5E39D
116FA80A3FDDD6D3202A092BDE2271351933572837036841669E7A68DA3DADC5336037795E0DB85F2
21EDF0AA7AAE1CA1C1331B413072070E764C8B32F9FB00150B28A9F95C44B27A13996D6A3BA95E6482
870D5ED8FFA9A2A7262C4FA50D8F208EC1EF15EC9A3CF9968F1D8C940F28505A042D61892117AEEEC0
4D49FECA964167AE770F66694A05338A9D6802B8942CA179F01E66B19E0AF3A05DD795831D7AE12E84
54026C9171E89256EABD0680E06C12033E54831AE89A937FCD3559F161DCA93205E4D5AFCA3774CF1A
09C48DE43E8C883468A2FC72CCBCEACB3EFA72F1258EC7E7E8671CE4081CEBB14EB831258A81B2BFF6
D3A5449FC49F51C5F4E69F9DF7CDB83134E447990BF6695A21349A4ED21B630425AEAEF7AED0D278BD
F304464D8E61AE2747790096E966955A3C2D963B17D34A8930C9E431965A607D18EE814C9A126773E1
15875A402B6C14F9626528436ACB81B881C4300491D744B788A01023D04566F502288D06922140F8CB
8CE49B4CA8687A818AD03B6CF07A82636DC3085B4BF2E80976A0AD6DE7C658AE85AC8F6F1724D62A4
76D9FB5E8AF53ED613EA7271E814664B4699C5B6EE139DCD002B3461C92DBDE1B1EBC28D12A27792C
94880A6D2DED466D4AB065A9AC0355EFC65B3B90317B73228203DB3038146ABC308C7532F46D8C2E9
98EFD2167993FB6A68949DA0F61C8143504569737DFA8340EABF0E01E8E7A42DC00D476814E07119C1
E1B973EB126DD164FC23813C6B618FD58D58045334A2D765A0DCE2457EE5F99E
```

Specifications of the parameters of API Response:

Parameter Name	Data Type	Length	Description
merchId	Int	15	Merchant Unique Reference Number shared by NDPS
Userid	String	45	Password shared by NDPS
merchantTxnId	String	50	Txn Id pertaining to merchant
mccCode	String	7	Master Category Code provided by NDPS
merchTxnDate	datetime	datetime	Transaction Date
atomTxnID	Int	20	NDPS transaction ID
prodName	String	50	Name of the Product
prodAmount	Double	12,2	Penny Drop Amount
Amount	Double	12,2	Amount
surchargeAmount	Double	12,2	Not to be passed
totalAmount	Double	12,2	Total Amount
custAccNo	String	45	Bank Account Number of End User
custAccIfsc	String	45	IFSC of end user
clientCode	String	45	Unique Client Code passed by Merchant
txnCurrency	String	5	Currency used for transaction
Remarks	String	20	Remark passed by Merchant
Signature	String	256	Signature generated
txnInitDate	datetime	datetime	NDPS Txn Initiated Date
txnCompleteDate	datetime	datetime	Transaction Complete Date
Subchannel	String	45	Sub Channel Id
otsBankId	String	10	Bank ID stored at Bank's end
bankTxnId	String	30	Bank Reference Id
authId	String	10	authId
otsBankName	String	60	Sponsor Bank Name
cardType	String	20	CC/DC
cardMaskNumber	String	20	Card Number
qrString	String	650	Qr String
Extras	String	50	Extras
custFirstName	String	80	Name received in response from Issuing Bank
custLastName	String	80	Last Name of User
custEmail	Email	100	Email of end user
custMobile	String	20	Mobile of End User
billingInfo	String	30	billing info
statusCode	String	30	Status Code of Transaction
Message	String	30	Success/Failure Message
Description	String	50	Description of Transaction

Sample Decrypted Response (Open Data):

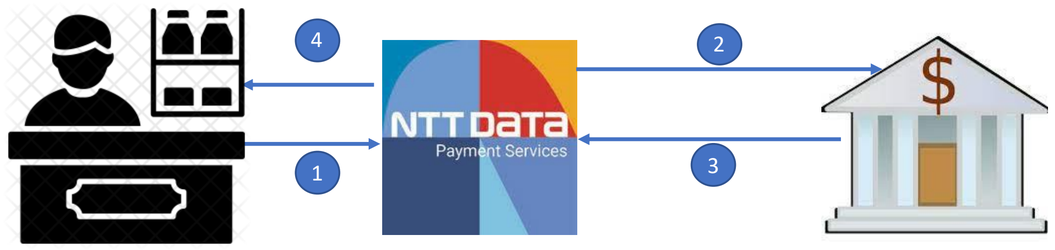
```
{
  "payInstrument":{
    "merchDetails":{
      "merchId":9135,
      "merchTxnId":"OTS77",
      "merchTxnDate":"2022-05-02T18:43:44"
    },
    "payDetails":{
      "atomTxnId":11000000219503,
      "prodDetails":[
        {
          "prodName":"Mangeshtest",
          "prodAmount":3.0
        }
      ],
      "amount":3.00,
      "surchargeAmount":0.43,
      "totalAmount":3.43,
      "custAccNo":"123456789",
      "clientCode":"32454",
      "txnCurrency":"INR",

"signature":"89292927e10f23244d165054ba5ff5cd0ca705a3831abdd7a2ce72d70ea274a1e4c8376d41c
a01a2bcff034150ea46b9dcc9e6a29bd457fe1c2bbf7dc9745ca9",
      "txnInitDate":"2022-05-02 18:43:56",
      "txnCompleteDate":"2022-05-02 18:44:00"
    },
    "payModeSpecificData":{
      "subChannel":[
        "NB"
      ],
      "bankDetails":{
        "otsBankId":1,
        "bankTxnId":"FueLffDEmCnahpAyBw2s",
        "otsBankName":"State Bank of India"
      }
    },
    "extras":{

    },
    "custDetails":{
      "custEmail":"atomdev-paynetz@atomtech.in",
      "custMobile":"5000000001",
      "billingInfo":{
        }
      },
      "responseDetails":{
        "statusCode":"OTS0000",
        "message":"SUCCESS",
        "description":"TRANSACTION IS SUCCESSFUL."
      }
    }
  }
```

6. Flow of Transaction API:

Transaction API



1. NTT receives payment request from merchant of the transaction carried out by end customer through transaction API in encrypted form
2. NTT DPS forwards this request to bank
3. NTT receives response from the bank of the transaction being successful/failed
4. NTT posts the status of the transaction at the merchant’s end in encrypted form. The merchant decrypts the same and publishes infront of the end user.

7. Response codes:

Error Code	Message	Description
OTS0000	SUCCESS	TRANSACTION IS SUCCESSFUL
OTS0101	CANCEL	TRANSACTION IS CANCELLED BY USER ON PAYMENT PAGE
OTS0201	TIMEOUT	TRANSACTION IS TIMEOUT
OTS0401	NODATA	NO DATA
OTS0451	INVALIDDATA	INVALID DATA
OTS0501	INVALIDDATA	INVALID DATA
OTS0600	FAILED	TRANSACTION IS FAILED
OTS0301	INITIALIZED	TRANSACTION IS INITIALIZED
OTS0351	INITIATED	TRANSACTION IS INITIATED
OTS0551	PENDING	TRANSACTION IS PENDING

8. Java code for transaction API implementation

The API involves encryption and decryption of data. Encryption is done while the request is being sent by merchant to NTT Data payment service. This encrypted data is decrypted by NTT data payment service.

In case of response being sent to merchant, encrypted response is sent to the merchant. Merchant decrypts the response and acts accordingly on the transaction.

Class code for encryption and decryption:

```
import java.util.logging.Logger;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;

public class AtomEncryption {
static Logger log = Logger.getLogger(AtomEncryption.class.getName());

private static int pswdIterations = 65536;
private static int keySize = 512;
private static final byte[] ivBytes = {
        0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
};
```

```
public static String encrypt(String plainText, String key) {
        try {
                byte[] saltBytes = key.getBytes("UTF-8");

                SecretKeyFactory                    factory                    =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");
                PBEKeySpec spec = new PBEKeySpec(key.toCharArray(), saltBytes, pswdIterations,
keySize);

                SecretKey secretKey = factory.generateSecret(spec);
                SecretKeySpec secret = new SecretKeySpec(secretKey.getEncoded(), "AES");

                IvParameterSpec localIvParameterSpec = new IvParameterSpec(ivBytes);
                Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                cipher.init(1, secret, localIvParameterSpec);

                byte[] encryptedTextBytes = cipher.doFinal(plainText.getBytes("UTF-8"));

                return byteToHex(encryptedTextBytes);
        } catch (Exception e) {
                log.info("Exception while encrypting data:" + e.toString());
        }

        return null;
}

public static String decrypt(String encryptedText, String key) {
        try {
                byte[] saltBytes = key.getBytes("UTF-8");
                byte[] encryptedTextBytes = hex2ByteArray(encryptedText);

                SecretKeyFactory                    factory                    =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");
                PBEKeySpec spec = new PBEKeySpec(key.toCharArray(), saltBytes, pswdIterations,
keySize);

                SecretKey secretKey = factory.generateSecret(spec);
                SecretKeySpec secret = new SecretKeySpec(secretKey.getEncoded(), "AES");

                IvParameterSpec localIvParameterSpec = new IvParameterSpec(ivBytes);
                Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                cipher.init(2, secret, localIvParameterSpec);

                byte[] decryptedTextBytes = (byte[]) null;
                decryptedTextBytes = cipher.doFinal(encryptedTextBytes);

                return new String(decryptedTextBytes);
        } catch (Exception e) {
                log.info("Exception while decrypting data:" + e.toString());
        }

        return null;
}

private static String byteToHex(byte[] byData) {
			StringBuffer sb = new StringBuffer(byData.length * 2);

			for (int i = 0; i < byData.length; ++i) {
				int v = byData[i] & 0xFF;
				if (v < 16)
					sb.append('0');
				sb.append(Integer.toHexString(v));
			}

			return sb.toString().toUpperCase();
		}

		private static byte[] hex2ByteArray(String sHexData) {
			byte[] rawData = new byte[sHexData.length() / 2];
			for (int i = 0; i < rawData.length; ++i) {
				int index = i * 2;
				int v = Integer.parseInt(sHexData.substring(index, index + 2), 16);
				rawData[i] = (byte) v;
			}

			return rawData;
		}

		public static void main(String[] args) {

			try {

				String		encryptedData		=		AtomEncryption.encrypt("1235",
		"ASWKLSLLFS4sd4g4gsdg");
				System.out.println("encryptedData : " + encryptedData);
			} catch (Exception e) {
				// TODO: handle exception
			}
		}

}
```

Class code for signature element:

Signature is an element of the parameters being passed. It is part of the Category account validation. Signature needs to be created every time a new transaction is initiated, as well as at transaction response when the signature is matched.

a) Request Signature -

The code for the same is as follows:
The generateSignature method shown below needs to have two parameters -one hashKey and other Param. The param field consists of following elements – merchantId + TxnPassword + MerchantTxnID + Payment Mode + Total amount + currency + stage. The plus sign denotes concatenate operator without space.

Eg: Merchant Id: 1191
TxnPassword: ABC@123
MerchantTxnID: OTS77
Payment Mode:
Total amount: 1000
Currency: INR
Stage:1

Therefore, the param field will be of format: **1191ABC@123OTS771000INR1**

b) Response Signature –

The generateSignature method shown below needs to have two parameters - one hashKey and other Param. The param field consists of following elements – [Merchant ID + Atom Txn ID + Merch Txn ID + Total Amount + Product + Txn Status Code + Subchannel + Bank Txn ID] encrypted using Response Hash Key. The plus sign denotes concatenate operator without space.

Eg: Merchant Id: 1191
Atom Txn ID: 11000000219503
MerchantTxnID: OTS77
Total amount: 1000
Product: Mangeshtest
Txn Status Code: OTS0000
Subchannel: NB
Bank Txn ID: FueLffDEmCnahpAyBw2s

Therefore, the param field will be of format:
119111000000219503OTS771000MangeshtestOTS0000NBFueLffDEmCnahpAyBw2s

There are two types of hashkey- request and response hash keys. For encryption, the end-user will use request key and for decryption of response they will be using the response key. The request and response key are merchant specific and has following characteristics.

Parameter Name	Mandatory	Data Type & Max Length	Sample Value	Content/ Remarks
reqHashKey	Mandatory	String(20)	KEY1234567234	Need to be save per MID on boarded in MW in Merchant Table

respHashKey	Mandatory	String(20)	KERESPY1234567234	Need to be save per MID on boarded in MW in Merchant Table

Java code for the same is as follows:

```
import java.io.PrintStream;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class NDPSSignature
{
 public static String generateSignature(String hashKey, String[] param)
 {
  String resp = null;

  StringBuilder sb = new StringBuilder();
  for (String s : param) {
   sb.append(s);
  }

  try
  {
   System.out.println("String =" + sb.toString());
   resp = byteToHexString(encodeWithHMACSHA2(sb.toString(), hashKey));
  }
  catch (Exception e)
  {
   System.out.println("Unable to encocd value with key :" + hashKey + " and input :" + sb.toString());
   e.printStackTrace();
  }
  return resp;
 }
 private static byte[] encodeWithHMACSHA2(String signatureKey, String text)
  throws NoSuchAlgorithmException, InvalidKeyException, UnsupportedEncodingException
 {
  Key sk = new SecretKeySpec(keyString.getBytes("UTF-8"), "HMACSHA512");
  Mac mac = Mac.getInstance(sk.getAlgorithm());
  mac.init(sk);
  byte[] hmac = mac.doFinal(text.getBytes("UTF-8"));

  return hmac;
 }
public static String byteToHexString(byte byData[])
        {
                StringBuilder sb = new StringBuilder(byData.length * 2);

                for(int i = 0; i < byData.length; i++)
                {
                        int v = byData[i] & 0xff;
                        if(v < 16)
                                sb.append('0');
                        sb.append(Integer.toHexString(v));
                }
                return sb.toString();
        }
```

UAT environment details:

The UAT environment details are as follows:
13.127.25.237

The above is the IP address of the UAT server for callback scenarios.

UAT server: The UAT server details to be provided are that of the

The UAT server needs to be whitelisted at the merchant’s end so that we can post on the merchant side.

Request parameters to be received from NTT Data payment services.

- MCC code: This will be provided to merchant based on his business operations.