NTT DATA
Payment Services

Info type: Confidential
Company: NTT Data Payment Services India Limited
Info. owner: Product (PMG)

Transaction Status (Requery) API

CONFIDENTIALITY DISCLAIMER

The information included in this document is confidential information relating to the business of NTT Data Payment Services, India(NDPS). It is being presented to you based on the understanding that it will not be used for any reason other than consideration of a commercial relationship with NDPS and, will not be used in connection with any decision to trade in securities of NDPS. Please be advised that any disclosure of the information contained in this document/presentation to any other person, or any use of this information in connection with the trading of NDPS securities, may be a violation.

This document, or any part of it, may not be reproduced, copied, circulated and/or distributed nor quoted without prior written approval from NDPS.

A. Document Information

Document Attributes	Information
Document Name	Transaction Status (Requery) API
Document Version	1.02
Owner	PMG
Author	Aviral Tripathi
Approved	Pavan Nikumbh

B. Revision Chart

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

Version	Primary Author	Description of Version	Date Completed	Reviewed By
1.0	Aviral Tripathi	Transaction Status (Requery) API	07/06/2022	Pavan N.
1.01	Aviral Tripathi	Status Codes Updated	09/02/2023	
1.02	Aviral Tripathi	Change in Status Code description in Requery API response with details added in 'Point to be Noted' section	05/06/2023	

NTT DATA
Payment Services

Contents

A. Document Information 2

B. Revision Chart 2

1. Description 4

Transaction Status Tracking Process: 4

2. Request Format 4

Transaction Status Sample Request (Open Request-JSON): 4

Transaction Status Sample Request (Encrypted Request): 5

Sample Encrypted Request Data: 5

Specifications of the parameters of API Request: 6

3. Response Format: 6

Sample Encrypted Response Data: 6

Sample Encrypted Data from obtained Response to Decrypt: 7

Decryption of Response: 7

Sample Decrypted Response – Open Data: 8

Specifications of API Response: 9

Response Codes: 9

4. AES Encryption Logic: 10

AES Encryption Java Code: 10

Signature Generation Logic: 12

Signature Generation Java Code: 12

UAT environment details: 13

1. Description

This API is provided to the merchant to track the status of any transaction.

Transaction Status Tracking Process:

- Merchant can track the transaction initiated/completed (Success/Fail) by end user via Transaction Status (Requery) API. Merchant sends the MID credentials as provided by NDPS in login parameter along with the transaction details as encrypted data [Pg. 5] pertaining to transaction details of the transaction/refund of which the status is enquired.
- On initiating the Requery API, merchant will receive the status in encrypted response [Pg. 6]. Merchant will decrypt this response through decryption method [Pg. 7] .
- Merchant must incorporate the encryption logic [Pg. 9] provided by NDPS at their end, to send the encrypted data in request and to decrypt the encrypted response.

Note*: This API is a **Server-to-Server** Call.

2. Request Format

Transaction Status Sample Request (Open Request-JSON):

- Transaction Status API request UAT URL: https://caller.atomtech.in/ots/payment/status?
- Production URL : https://payment1.atomtech.in/ots/payment/status?
- Request and Response of Transaction Status (Requery) API will be encrypted using AES 512.

Request Parameters are to be shared in the format illustrated below:

```
{
"payInstrument" : {
"headDetails" : {
"api" : "TXNVERIFICATION",
"source" : "OTS"
},
"merchDetails" : {
"merchId" : 9135,
"password" : "Test@123",
"merchTxnId" : "250420221",
```

"merchTxnDate" : "2022-04-25"
},
"payDetails" : {
"amount" : 1.00,
"txnCurrency" : "INR",
"signature" :
"abaf4b4011b6813c0a16896302a6fab404035df377d3b25e60b8a6766dffb6383891a7443f603fc99b643e2bf4049d34eccc74e3253
3c742c25580f60e17ab2a"
}
}
}

Transaction Status Sample Request (Encrypted Request):

https://caller.atomtech.in/ots/payment/status?merchId=9135&encData=400E949F8F951060A21D462EC57CB03341BBB5627A23973D22BB39E312AA7491788FE15430FC7019851C8EC737B613F70740488E050EBA67B08083103AB5C0D139A35E47FE9C41A36C44C9EB181DA931FED63BE0F68C82B26F791DC805395AB0580AB7B7CF942061980C94AEE5AC50DD01225B6135A5F3F7AFA5646130D3CEFFC241AB316CBC2ECD34AD757B05CC8EFCBC0F91961325604CD71DFDE35AFD48F0070FD94136986DFE1573ABDF2F1DD1027B1581BA59FB6D635FF8DF1FDC70C27A011197E80E069F2D473493C614B000FEB951DA47EF3433ED1B3E00F8A5625379290CE453DBC792BDDEF825F11224BFDB5FF02BCDCD0B08901E97D7010F394BC50DD1E891BE655768B2E34CEBCDBA52B5423E079E9ADFF5F64E6560F86499DC1E6124242F46DC077C128A6A8574EEE6148177B7EBB0F8CC09CD7CDA85505C390CD49C10F79191680FCF1D048DEF669CF1DD42D0F2E66A7AA81E56D537796C7DF271BFC181D32C71425B26FE0F2574DA406250854CDC8045CE9AE5E06309BDA8F1885EC40CB5E5B9FC2210079232F08E0DF2B5A88DE5B0AA8B433AFD2DFBEE616C3EC946D5A48517E4F6AEC3CA9172CE3F

Sample Encrypted Request Data:

400E949F8F951060A21D462EC57CB03341BBB5627A23973D22BB39E312AA7491788FE15430FC7019851C8EC737B613F70740488E050EBA67B08083103AB5C0D139A35E47FE9C41A36C44C9EB181DA931FED63BE0F68C82B26F791DC805395AB0580AB7B7CF942061980C94AEE5AC50DD01225B6135A5F3F7AFA5646130D3CEFFC241AB316CBC2ECD34AD757B05CC8EFCBC0F91961325604CD71DFDE35AFD48F0070FD94136986DFE1573ABDF2F1DD1027B1581BA59FB6D635FF8DF1FDC70C27A011197E80E069F2D473493C614B000FEB951DA47EF3433ED1B3E00F8A5625379290CE453DBC792BDDEF825F11224BFDB5FF02BCDCD0B08901E97D7010F394BC50DD1E891BE655768B2E34CEBCDBA52B5423E079E9ADFF5F64E6560F86499DC1E6124242F46DC077C128A6A8574EEE6148177B7EBB0F8CC09CD7CDA85505C390CD49C10F79191680FCF1D048DEF669CF1DD42D0F2E66A7AA81E56D537796C7DF271BFC181D32C71425B26FE0F2574DA406250854CDC8045CE9AE5E06309BDA8F1885EC40CB5E5B9FC2210079232F08E0DF2B5A88DE5B0AA8B433AFD2DFBEE616C3EC946D5A48517E4F6AEC3CA9172CE3F

Specifications of the parameters of API Request:

Parameter Name	Conditional/ Optional/ Mandatory	Data Type & Max Length	Sample Value	Content/ Remarks
api	Mandatory	String (20)	TXNVERIFICATION	For Transaction verification "TXNVERIFICATION" fixed
source	Mandatory	String (3)	It has to be only "OTS"	It's static, only OTS
merchId	Mandatory	int(15)	9135	Unique ID assign by NDPS to merchant
password	Mandatory	String (50)	Password provided by NDPS	Password Provided by NDPS
merchTxnId	Mandatory	String(50)	1234567890	Unique transaction ID provided by merchant
merchTxnDate	Mandatory	Date	2022-04-25	Transaction date must be in yyyy-mm-dd
signature	Mandatory	String (256)	eced0d634d6fa6637 6d40cbc1c5a812c5f0 fbc9c2fe756c459f5e3 9c3455ba7ebb9053c d3d3b37191d9d20e7 94baf4923b3e1f45c5 9c8ca34f9be603f891 57	Signature generation using logic provided by NDPS - encrypts merchID, password, merchTxnID, amount, txnCurrency, api, reqHashKey
amount	Mandatory	double (12,2)	10.00	Transaction Amount
txnCurrency	Mandatory	string (5)	INR	The currency code in which the payment is made

3. Response Format:

Response to the transaction status request will comprise of the below illustrated encrypted Data.

It needs to be decrypted as per AES Decryption logic provided by NDPS.

Sample Encrypted Response Data:

encData=F5140AF9DC1B3DB7AFA300D9675EE72A54618F3A30059A96C668F7148BCA948D88451AEF2A7E37F625DDC2B4E1822587E62A3287CA5788F51C2D78186D40C11ECA66EF7048D3B64306CD22251D3078D061D9B141A6E2DEA467F6BEF8C8E0A78674DF6650A4011E6462C853930F30D441CFF3F9AD4CA7443A5A3837E222E502BFF3F2D815A4DE67F29A5AB49D1DE6FA3D5E11075B8399686F554AA764E710E3D5A7FBAF44A3D490E21851F92B8F2BBEB2C1572EBFF02F

15D14D34A5C94EC909FC1B7F9F5510EE8176E968C3C52DD3C41719DE50B4223C3AA87A9D8E228245F3057CA102A37
E7229B23CBF21ABF44B19D40B3E5B32C736FF5A910EA29EB2ECCE2035A3448C20DBC3FA1FC13235F93FF289216A75
8C597E7C770E3B510B607D6E5491FE16E82C068ECA6A88C29D80922332DC1C29F362F8A9EB938C66417CA3900067EE
15F394C5179C3DFC08C1EF92A1277638D8D9DDE1F3836E4219FB4B5BB00A45E58D531C4CEBA66A9BE0C9DE2FB4F3
46F1C396EB252F98F260FB3B4977B01F56B809F4153307765B73CF56829857E4F4AD90542382AE7384D7FCCA121BFD0
1D9F2DE6E216F903BBA123A95D51D09D8FECF5174BB78986AB8D48C21AEC6AA5B2D03C16BAB1F903EAC9BC714A
E92CFA0BCAB3B71BF1499B6407251582E3A9F5E93BE2CC7E3077AEB726F4D676FC100D618144E301AAD6720303FE2
094FF26FE868F137311883D6E580C15BBD9D13EBEA&merchId=9135

Sample Encrypted Data from obtained Response to Decrypt:

F5140AF9DC1B3DB7AFA300D9675EE72A54618F3A30059A96C668F7148BCA948D88451AEF2A7E37F625DDC2B4E182
2587E62A3287CA5788F51C2D78186D40C11ECA66EF7048D3B64306CD22251D3078D061D9B141A6E2DEA467F6BEF8C
8E0A78674DF6650A4011E6462C853930F30D441CFF3F9AD4CA7443A5A3837E222E502BFF3F2D815A4DE67F29A5AB4
9D1DE6FA3D5E11075B8399686F554AA764E710E3D5A7FBAF44A3D490E21851F92B8F2BBEB2C1572EBFF02F15D14D
34A5C94EC909FC1B7F9F5510EE8176E968C3C52DD3C41719DE50B4223C3AA87A9D8E228245F3057CA102A37E7229B
23CBF21ABF44B19D40B3E5B32C736FF5A910EA29EB2ECCE2035A3448C20DBC3FA1FC13235F93FF289216A758C597E
7C770E3B510B607D6E5491FE16E82C068ECA6A88C29D80922332DC1C29F362F8A9EB938C66417CA3900067EE15F394
C5179C3DFC08C1EF92A1277638D8D9DDE1F3836E4219FB4B5BB00A45E58D531C4CEBA66A9BE0C9DE2FB4F346F1C
396EB252F98F260FB3B4977B01F56B809F4153307765B73CF56829857E4F4AD90542382AE7384D7FCCA121BFD01D9F2
DE6E216F903BBA123A95D51D09D8FECF5174BB78986AB8D48C21AEC6AA5B2D03C16BAB1F903EAC9BC714AE92CF
A0BCAB3B71BF1499B6407251582E3A9F5E93BE2CC7E3077AEB726F4D676FC100D618144E301AAD6720303FE2094FF
26FE868F137311883D6E580C15BBD9D13EBEA

Decryption of Response:

Merchant must pass the encrypted response along with Merchant Specific Response EncryptionKey [Pg. 10] and MID in the decryption method as illustrated below:

decryptor = new AtomAES().decrypt(encryptedResponse, Key, iv);

DataType	Name	Value	Description
String	decstr	BFC23F835C2840C82CCA60671	Encrypted responseto the encrypted request triggered, that needs to be decrypted
String	Key	Key provided by NDPS, to decrypt theresponse	Key provided by NDPS, to decrypt theresponse
String	IV	Same as Key	Same as Key string
String	dec	new.ATOMAES().decrypt(decstr,key,IV);	Value of this string is an object. That is used to invoke the encrypt function of ATOMAES class. Postencryption, this variable will be appended in the request along with url, and login.

Sample Decrypted Response – Open Data:

Post decrypting the response successfully, merchant will get corresponding data in the below JSON format.

Response Parameters are obtained in the format illustrated below:


```
{
 "payInstrument": [
  {
   "settlementDetails": {
    "reconStatus": "PNRNS"
   },
   "merchDetails": {
    "merchId": 9135,
    "merchTxnId": "250420221",
    "merchTxnDate": "2022-04-25 13:14:57"
   },
   "payDetails": {
    "atomTxnId": 11000000216668,
    "product": "Mangeshtest",
    "amount": 1,
    "surchargeAmount": 0,
    "totalAmount": 1
   },
   "payModeSpecificData": {
    "subChannel": "CC",
    "bankDetails": {
     "bankTxnId": "0011000000216668325",
     "otsBankName": "Hdfc Bank",
     "cardMaskNumber": "401288XXXXXX1881"
    }
   },
   "responseDetails": {
    "statusCode": "OTS0000",
    "message": "SUCCESS",
    "description": "SUCCESS"
   }
  }
 ]
}
```

Specifications of API Response:

Parameter Name	Conditional/ Optional/ Mandatory	Data Type & Max Length	Sample Value	Content/ Remarks
reconStatus	Mandatory	String(10)	RNS	Reconciliation Status
merchId	Mandatory	int(15)	9135	Unique ID assign by NDPS to merchant
merchTxnId	Mandatory	String(50)	1234567890	Unique transaction ID provided by merchant system
merchTxnDate	Mandatory	datetime	2022-05-24 20:46:00	Transaction date must be in yyyy-mm-dd hh:mm:ss
atomTxnId	Mandatory	Numeric (16)	11000000216668	Unique transaction ID (NDPS)
product	Mandatory	string (50)	ACC01	Product Id provided by NDPS. Passed during the transaction initiation.
amount	Mandatory	double (12,2)	10.00	Amount to be paid
surchargeAmount	Optional	double (12,2)	0.00	surcharge amount
totalAmount	Mandatory	double (12,2)	10.00	Total amount [amount + surcharge amount]
subChannel	Mandatory	String(10)	BQ	Product used during Transaction
otsBankId	Mandatory	String(10)	2	Bank ID as per NDPS system
bankTxnId	Mandatory	String(20)	1234567	Bank Transaction ID
cardMaskNumber	Optional	String(20)	485498XXXXXX0465	Mask Card Number
statusCode	Mandatory	String(10)	0000	0000-Success
message	Mandatory	String(80)	Message for Status Code	SUCCESS
description	Mandatory	String(100)	Status Description	TRANSACTION IS SUCCESSFUL

Response Codes:

Error Code	Message	Description
OTS0000	SUCCESS	TRANSACTION IS SUCCESSFUL / FORCE SUCCESS *
OTS0002	FORCE SUCCESS	TRANSACTION IS FORCE SUCCESS
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 / AUTO REVERSAL *
OTS0301	INITIALIZED	TRANSACTION IS INITIALIZED
OTS0351	INITIATED	TRANSACTION IS INITIATED
OTS0551	PENDING	TRANSACTION IS PENDING
OTS0951	SOMETHING WENT WRONG	UNEXPECTED ERROR

* Either of the mentioned description will be received.

Points to be Noted*-

- 'OTS401' i.e., 'NO DATA' means that the transaction data is not available due to incorrect input.
- Requery API will fetch the status only within 30 days from the day of transaction.
- The description will be 'FORCE SUCCESS' if force success is enabled & as 'AUTO REVERSAL' if auto reversal is enabled. (For scenarios, refer to Callback API document).
- In case the transaction was cancelled by the User on payment page, the transaction status shall return the description as 'ABORTED' with no status code.

4. AES Encryption Logic:

- Transaction Status (Requery) API's request and returned response are shared via AES-512 encryption.
- The following KEY are to be used for UAT:

MerchId	encResKey	encReqKey
9135	58BE879B7DD635698764745511C704AB	7813E3E5E93548B096675AC27FE2C850

A E S

Encryption Java Code:

```
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
		}
}
}
```

Signature Generation Logic:

- For any given transaction, the Transaction Status API's request and the following response signature fields to be generated using the shared Hashing code.
- Signature generation sequence [merchID + password + merchTxnID + amount + txnCurrency + api]
- The UAT request and response hash key are:

MerchId	reqHashKey	respHashKey
9135	ea59e6ee036c81d8b5	ea59e6ee036c81d8b6

Signature Generation Java Code:

```
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 AtomSignature {
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 text, String keyString)
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 scenarios pertaining to Requery API.

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