NTT DATA
Payment Services

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

Callback API

NTT Data Payment services Ltd.
www.nttdatapay.com

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	Callback API
Document Version	1.0
Owner	PMG
Author	Nikhil Dayma
Approved	Pavan Nikumbh

B. Revision History

Version	Author	Description of Version	Date	Reviewed By
1.0	Nikhil Dayma	Callback API	07-04-2022	Pavan N.

CONTENTS

1. Introduction

This document will provide overview of call-back API process flow. Callback API is an API in which NTT Data payment services system posts transaction response through server-to-server mode in name value pair format on configured merchant's URL in NTT Data payment services system.

2. General process flow:

The general process flow of the call-back API is as follows:

- The end customer after purchasing product/service on merchant websites opts for payment, he either gets redirected to NTT Data payment service page or requested through API in case of seamless request.
- The user selects his choice payment option and provides credentials for authorization
- Once the authorization is completed the transaction is executed. For transaction to get successfully executed, the IP and domain URL needs to be whitelisted and configured.
- Once the end user completes transaction successfully and if callback URL is configured for that merchant, then the response will be posted on return URL as well as callback URL in both scenarios' success/failure.

3. Pre-requisite:

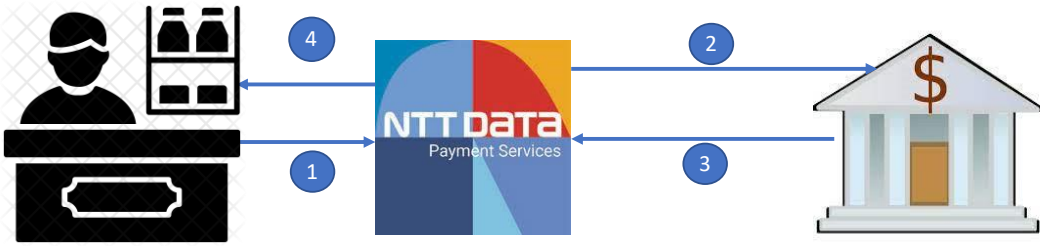
Merchant's IP and domain URL need to be whitelisted and configured at NTT data payment system and also NTT data payment server also needs to be whitelisted at their end.

4. Callback posting parameters:

Parameter Name	Conditional/ Optional/ Mandatory	Data Type & Max Length	Sample Value	Content/ Remarks
merchId	Mandatory	int(15)	8952	Unique ID assign by ATOM to merchant
merchTxnId	Mandatory	String(50)	1234567890	Unique transaction ID provided by merchant system
merchTxnDate	Mandatory	datetime	24-12-2019 20:46:00	Transaction date must be in yyyy-mm-dd hh:mm:ss
amount	Mandatory	double (12,2)	10.00	Amount to be paid
surchargeAmount	Optional	double (12,2)	0.00	surcharge amount
prodName	Mandatory	string (50)	NSE	Product Name
prodAmount	Optional	double (12,2)	10.00	Product wise amount division as provided by merchant
totalAmount	Mandatory	double (12,2)	10.00	Total amount [amount + surcharge amount]
custAccNo	Optional	String(45)	100000036600	Customer account number
clientCode	Mandatory	String(45)	12345	As provided by merchant
txnInitDate	Mandatory	datetime	03-08-2020 22:09:51	Transaction Initiation Date
txnCompleteDate	Conditional	datetime	03-08-2020 22:09:54	Transaction Completion Date for all successful txn.
subChannel	Mandatory	String (10)	BQ	Sub Channel(Payment Product will be displayed here)
otsBankId	Conditional	String (10)	2	Will share Atom bank ID
otsBankName	Mandatory	String(60)	HDFC Bank	Bank Name
bankTxnId	Mandatory	String(20)	UbiBgQp3Dwri0S347rge	Bank Txn ID
cardType	Conditional	String(20)	VISA	Card Type will be presented during a card based txn
cardMaskNumber	Conditional	String(20)	485498XXXXXX0465	Mask Card Number will be presented during a card based txn
statusCode	Mandatory	String(10)	OTS0000	OTS0000
message	Optional	String (80)	SUCCESS	Message for Status Code
description	Optional	String(100)	TRANSACTION IS SUCCESSFUL	Status Description

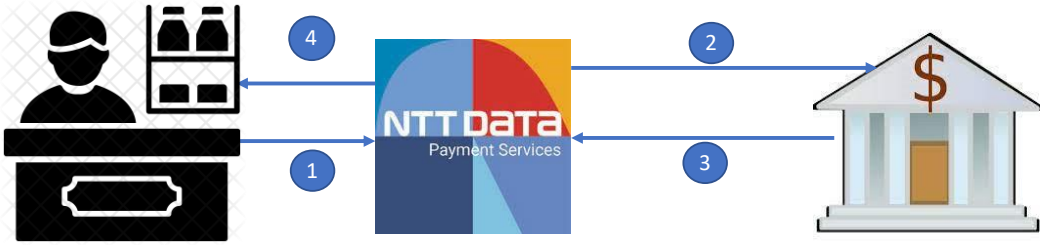
5. Flow of call-back API:

Call-back API- Non-Challan transactions



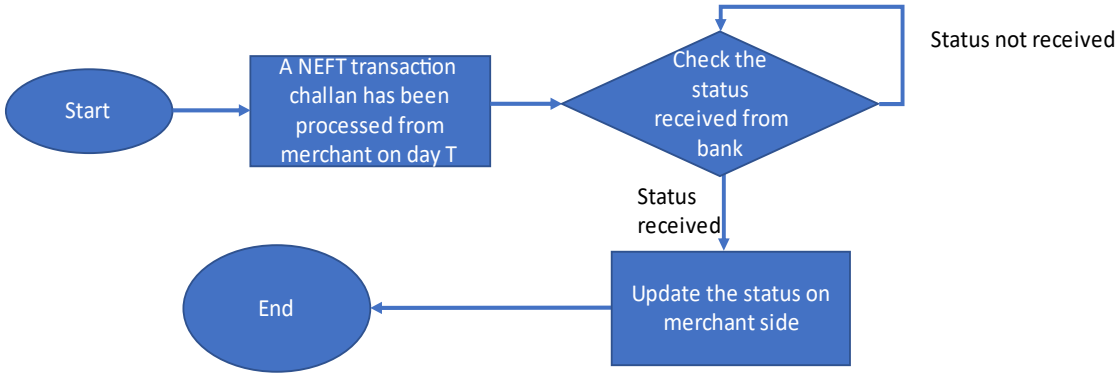
1. NTT receives payment request from merchant of the transaction carried out by end customer
2. NTT DPS forwards this request to bank
3. NTT receives response from the bank of the transaction being successful/failed till T, T+1, T+2
4. NTT posts the transaction at merchant's server through callback API

Call-back API- Non-Challan transactions after T+2 days



1. NTT receives payment request from merchant of the transaction carried out by end customer
2. NTT DPS forwards this request to bank
3. NTT receives response from the bank of the transaction being successful/failed after T+2 days
4. NTT posts the transaction at merchant's server through callback API as forced failed autoreverse the transaction to pay the end user.

Call-back API- Challan transactions



6. Sample response format:

Currently, the response is sent in JSON format to the merchant server. The response is posted in JSON format. The format is JSON.

Sample Encrypted Request Data:

The encrypted form of the response which will be received by merchant will be

encData=F5140AF9DC1B3DB7AFA300D9675EE72AFE09D27235E356AFAC21032BB1703A7EF4E8784726
3EA2FA702E373401BED367E2E6A3EDD65389EB47D1FC8630C7E708857FF81B6A671FFF0102399E14016
A9FE53552A206D95F5EC628A61822F0CC9506DB24AE82F3E1F478911D7495F4A6D105ADBAF90FBE56E
2E04CB55EA17E03FF5A09EBBC84980F72D37780A6D928BAB40A49593CD819DC0AFB6EA820B065E5E1E
3367E9C1D77C2153B190E6391B53FF0F26734B758EEAF32CFED1AD2D9FAF5FD8C4291AB992A7C84BBB
7CC2BAA556041EE0E077F0E8979F61E5BDD9B5A3D2B0D494AF4AF1E6AD32346B108983D64F916E2C02
7FFF4E4715808826EC56CD3992B014D4B930496E9CBEFC9765BD4AC3267747287096B0C264131EFB964
D279341D34C46B6463BF2A983F5F4E32338C5769108DE2F011E4CFEDB3213D55A74E85615DB9A569E7
C1EE53BE70CA6931D3F3C51F15E843426B60C3D320BF25242C6945440485210D6F76E6310D480D3ABD
B2D8D7D12ED4B892DA6B412D61A36BD125339E783F0912AF649758F617F683C4D04421D48E570D457
26FD3C82E2B0C5C2CE209247ED65830E72B27F2532F3A876ED98A783B4036971B97C55EC3CEF5BD2E4
D9B3123ACF973DC1B59AB255D6FD87312BA67433E4C15C1AD188FDC2D784EE6B2B25C9A23E34F819A
42034296997705F231C61B0C3C61AA4FD1237FE8BCAB6D238A573022E3E89F751D080E8A18142BEA96
63608A4E88172C1147680EB3B0DED793127A5E5F8935CF116A2C72F4205533B94F24836E6BBD83D892
2DF6B1DDF33668659D50C9F9410625310086609E53B2B1D1F42A137D2EB63B8547B994059A7E242776
6F0E27AE9351D9EB20D8CAC16CA2E2EB08A7B0B47CE0B8347DE157EDA051CC274AED624629649FC89
70902A8294F02C52B17ECDCB22EF839243CBC299A9227A85AF34C0363BB4E48D02854F8C6&merchId=
9135

Sample Decrypted Response (Open Data):


```
{
  "payInstrument":{
    "merchDetails":{
      "merchId":9135,
      "merchTxnId":"OTS77",
      "merchTxnDate":"2022-05-02 18:43:44"
    },
    "payDetails":{
      "amount":3.00,
      "surchargeAmount":0.43,
      "atomTxnId":11000000219503,
      "prodDetails":[
        {
          "prodName":"Mangeshtest",
          "prodAmount":3.0
        }
      ],
      "totalAmount":3.43,
      "custAccNo":"123456789",
      "clientCode":"32454",
      "txnInitDate":"2022-05-02 18:44:00",
      "txnCompleteDate":null
    },
    "payModeSpecificData":{
      "subChannel":[
        "NB"
      ],
      "bankDetails":{
        "otsBankId":1,
        "otsBankName":"State Bank of India",
        "bankTxnId":"FueLffDEmCnahpAyBw2s"
      },
      "cardDetails":{
        "cardScheme":null,
        "cardMaskNumber":null
      }
    },
    "responseDetails":{
      "statusCode":"OTS0000",
      "message":"SUCCESS",
      "description":"TRANSACTION IS SUCCESSFUL."
    }
  }
}
```

Scenario 1-

If transaction get success at atom end, on real time atom will post "**message** = **SUCCESS**" response with help of callback API.

Scenario 2-

If on T Day transaction is pending at atom end, after bank re-query response on same day if transaction is success at bank end then status will be changed from pending to success at atom end, then atom will post "**message =SUCCESS**" response on Callback API.

Scenario 3-

If on T day transaction is pending at atom end, on T+1-day post reconciliation if transaction is success then status gets changed from **pending** to **success** at atom end, then atom will post "**message =SUCCESS**" (Force success) response with help of Callback API. If it's Failed response will be "**message =FAILED**".

Scenario-4-

If on T Day transaction getting failed at atom end, on real time atom will post "**message =FAILED**" response with help of Callback API.

Scenario-5-

Until and unless we are not getting response (Success/Failed) from bank end, Atom will not trigger Callback API. Till T+2 if we're not getting response from Bank, Atom system will mark transaction as Fail, but Callback will not be triggered. If transaction status is received as success from Bank's end after T+2, irrespective to the configuration [Force Success/Auto Reversal] in Atom system, same will be marked as Auto Reversal.

Scenario-6-

For **NEFT** Transactions Challan Generated on T Day, on (T+1, T+2, T+3….) day post reconciliation transaction status gets changed from **challan generated** to **Success** at atom end, then atom will post "**message=SUCCESS**" response with help of Callback API. In This scenario, no real time response will be posted.

7.Response codes

Error Code	Message	Description
OTS0000	SUCCESS	TRANSACTION IS SUCCESSFUL
Any Other response	-	TO BE TREATED AS FAILURE

8.Decryption of data

The data received by the merchant from NTT Data payment server will be in encrypted format using AES 512. To receive the data in JSON format, the data will have to be decrypted using response encryption credentials.

The sample key for UAT is

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

Java code for 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 = 256;
 private static final byte[] ivBytes = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };



 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 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 decryptedData = AtomEncryption.decrypt("Encdata", "ASWKLSLLFS4sd4g4gsdg");
              System.out.println("decryptedData : "+decryptedData);
        } catch (Exception e) {
              // TODO: handle exception
        }

}

}
```

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.