NPCI Mandate Approval Gateway Service

Bank Specification Document

Version 4.2.3

DO NOT COPY

DOCUMENT RELEASE NOTICE

Document Details

Name	Version No.	Date	Description
Bank Specification Document	Draft	24-02-2017	Provides technical & operation specification for Banks to develop compatible application at their end for communicating with the Mandate Authorization application
NPCI Mandate Authorization Specification for Banks	1.0	01-03-2017	Updated for Debit Card
NPCI Mandate Authorization Specification for Banks	1.1	22-03-2017	Covered the specification for Signing, Check Sum & Encryption. XML Specification, XSD & XML Samples attached as zip
NPCI Mandate Authorization Specification for Banks	2.0	27-03-2017	Updated for Error Scenarios, HTTP Status codes.
NPCI Mandate Authorization Specification for Banks	3.0	26-05-2017	API to get live destination banks for e-mandate Separate URL's for Net banking & Debit Card Corporate mapping to the destination banks
NPCI Mandate Authorization Specification for Banks	3.1	08-06-2017	Updated the process flow to include bank selection in the merchant page.

NPCI Mandate Authorization Specification for Banks	3.2	24-06-2017	Addition of Dbtr tag in Request XML. Changes in Error Response XML's, Error Codes & Failure Scenarios (In Appendix)
NPCI Mandate Authorization Specification for Banks	3.3	03-Jul-2017	Error Codes & Failure Scenarios Sheet Updated. Encryption of Debtor field instead of Creditor. Changes in Server to Server communication specification.
NPCI Mandate Authorization Specification for Banks	3.4	14-Jul-2017	Changes in Request & Response XML formats and Error XML format.
NPCI Mandate Authorization Specification for Banks	3.5	07-Aug-2017	Handling of Timeout Scenario Added
NPCI Mandate Authorization Specification for Banks	3.5	20-Dec-2017	Encryption methodology updated Updates to Offline API's Error Codes Updated
NPCI Mandate Authorization Specification for Banks	3.6	18-Sep-2018	AuthMode added as additional Parameter from Merchant. Flow changes based on this parameter.
NPCI Mandate Authorization Specification for Banks	3.7	15-APR-2019	Change in API "Posting list of Open Transactions to Bank"

NPCI Mandate Authorization Specification for Banks	3.8	17-May-2019	Changes in Error XML Structure from Bank to NPCI and from NPCI to Merchant (Appendix 9.1) Changes in lengths and data types of XML elements in Merchant Request. Changes in Error Response from Bank Change in live bank list api
NPCI Mandate Authorization Specification for Banks	4.0	12-DEC-2019	Changes in Merchant request XML, Bank request XML & Merchant response XML. Encryption of additional fields Encryption of Request XML and Response XML Additional parameter in the form post for Merchant
NPCI Mandate Authorization Specification for Banks	4.1	29-JUL-2020	Introduction of New Debit Card Flow BANKID & AUTHMODE mandatory in merchant request
NPCI Mandate Authorization Specification for Banks	4.2	29-DEC-2021	Introduction of Aadhaar flow
NPCI Mandate Authorization Specification for Banks	4.2.1	29-MAR-2022	Added Signature and Checksum in bank response
NPCI Mandate Authorization Specification for Banks	4.2.2	25-MAY-2022	Signature, checksum logic added Aadhaar Mandate Validation, Aadhaar OTP validation Checksum fields added.

			Request Parameter aadhaarNumber updated as aadhaarNo Aadhaar response corrected
NPCI Mandate Authorization Specification for Banks	4.2.3	03-Jun-2022	aadhaarAuthDtls attiribute corrected in ResendOTP Aadhaar Request Resend OTP Request format added for Direct Debit card flow
NPCI Mandate Authorization Specification for Banks	4.2.4	23-Aug-2022	Debit Card failure reattempt screenshot added for Direct Debit card flow
NPCI Mandate Authorization Specification for Banks	4.3	25-10-2022	Introduction to Amend, Cancel, Suspend and Revoke,Custom cancel
NPCI Mandate Authorization Specification for Banks	4.4	30-05-2023	Introduction of PAN and Cust ID authentication mode

This document and any revised pages are subject to document control. Please keep them up-to-date using the release notices from the distributor of the document.

Table of Contents

1. Introduction

This document details the requirement for destination banks to develop the required interface for interacting with the Mandate Authorization gateway service.

The file formats for request & response are covered in this document.

1.1 Abbreviation

The below abbreviations are used in the document.

NPCI	National Payments Corporation of India
ONMAGS	Online Mandate Approval Gateway Service
UIDAI	Unique Identification Authority of India

2. Interface specification details for Mandate Approval

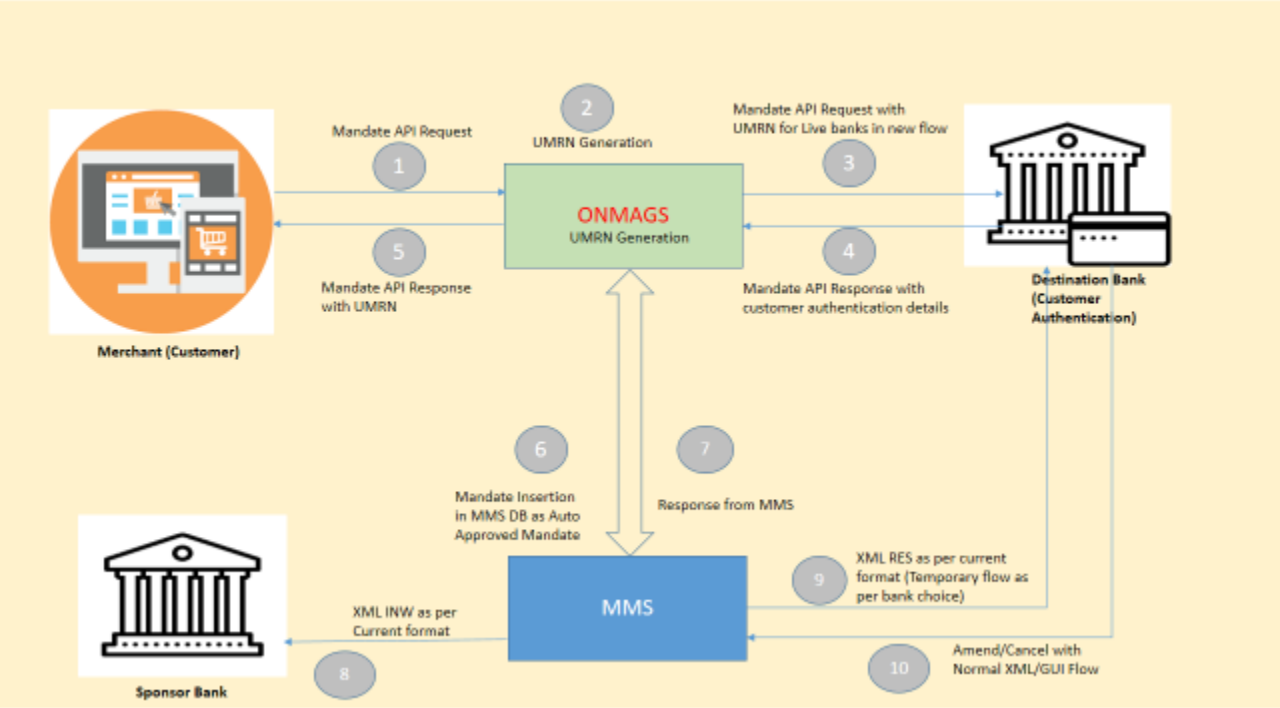
2.1 Registration with NPCI

The destination banks who want to leverage the service need to be registered with NPCI and get certified.

2.2 Mandate Approval function flow (for Net Banking & Debit Card authentication modes)

The mandate approval flow is initiated from the Merchant end, request validated at NPCI end and forwarded to the Bank for authorization. The confirmation provided back by the Destination Bank is relayed back to the merchant.

Mandates created through ONMAGS will be auto registered in MMS. The overall flow and the integration between ONMAGS and MMS systems is explained by the below diagram.

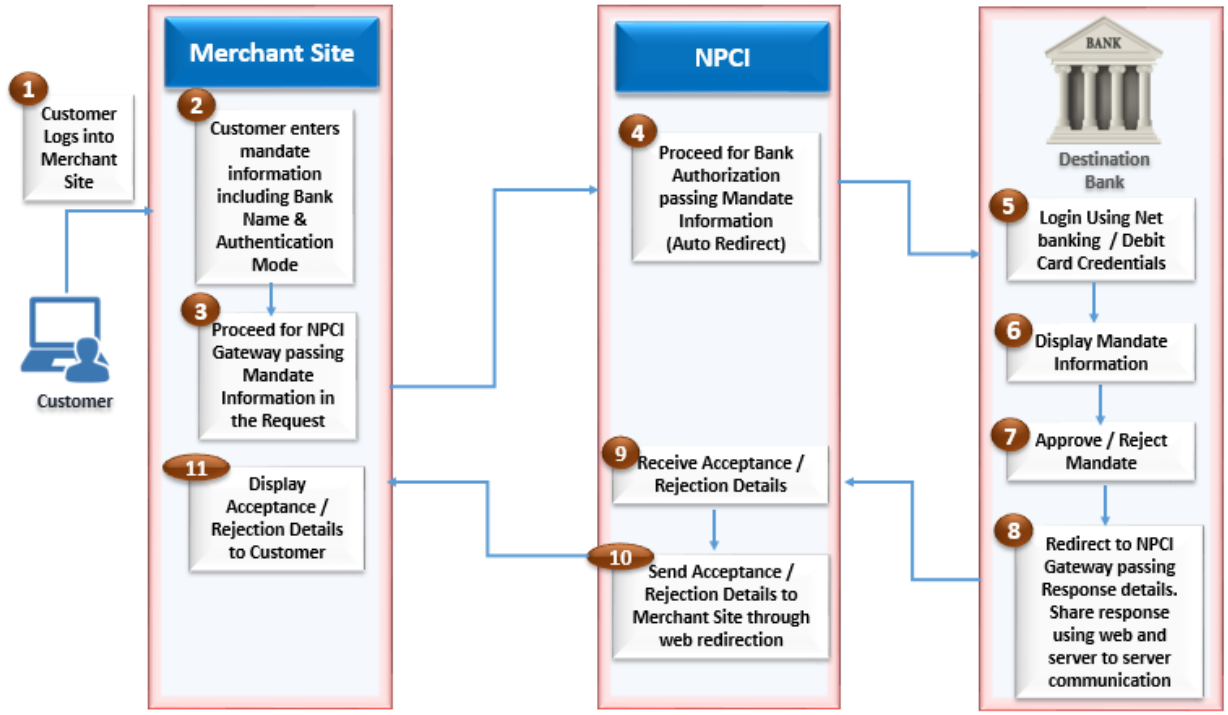


The process flow is mentioned in the next section.

Note:
From version 4.1 BankID & AuthMode are mandatory in the merchant request

2.2.1 End to End Process Flow (for Net Banking & Debit Card Authentication Modes)

The below diagram illustrates the functional flow of mandate authorization when Bank ID & Authentication Mode are passed from Merchant. This will be the default flow from version 4.1.



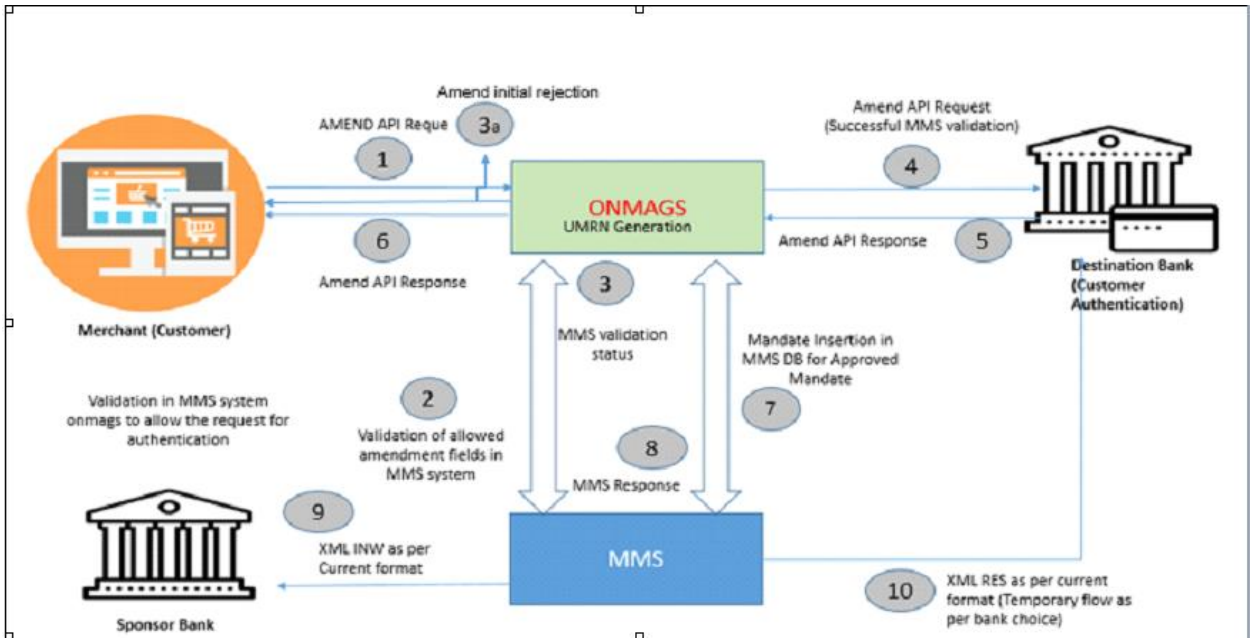
Note:-

In case of new debit card/Aadhaar flow there will not be any redirection to Bank. The debit card/Aadhaar authentication will happen in NPCI side itself. For this ONMAGS will interact with Banks through API calls for validating the mandate and debit card/Aadhaar information. (Detailed flow explained in section 4.2.3)

- Customer logins to the merchant site where he/she would be shown the mandate Information
- Specific details of the mandate along with deduction details needs to be shown.
- Customer can proceed with accepting the mandate if he/she finds the information displayed is correct (Customer needs to enter the Bank account number before proceeding)
- Merchant site needs to provide the option for selecting Bank & Authentication Mode (NetBanking, Debit Card, Aadhaar Card, PAN, Cust ID).
- Customer would be redirected to NPCI ONMAGS interface.
- NPCI Interface would show an intermittent page while processing happens in the back ground.

- If the validation is successful, then NPCI will auto redirect to Bank's authentication page based on the Bank ID & Authentication Mode selected by the end user in the merchant site.
- If the validation fails, then NPCI will redirect back to the Merchant Site posting the Error XML response.
- Bank will display the authentication Page based on the Auth mode selected by the user.
- In the Banks page customer will authenticate either using the user's net banking credential or Debit card credentials based on the authentication mode user had selected in the Merchant page.
- Bank need to validate whether the Account Number passed in the request XML matches the Account Number through which the customer has authenticated the login.
- Once verified Bank Page will display the summary of the mandate and provide option for accepting or rejecting the mandate
- Once the customer has selected either of Approve / Reject link he would be redirected back to NPCI ONMAGS interface
- The NPCI ONMAGS interface will auto redirect to the merchant site
- Merchant site will display the status of Mandate Approval

2.3 Mandate Approval function flow(AMEND, CANCEL, SUSPEND, REVOKE)



The mandate approval flow is initiated from the Merchant end, request validated at NPCI end and sent to MMS for validation, if validation is successful the request is forwarded to the Bank for authorization. The confirmation provided back by the Destination Bank is relayed back to the merchant, if the confirmation from bank is success, then persistence request is initiated to MMS.

Mandates Amend/Cancel/Suspend/Revoke through ONMAGS will be auto registered in MMS. The overall flow and the integration between ONMAGS and MMS systems is explained by the above diagram.

2.4 Interface Layer

Necessary ports need to be opened between NPCI servers & Bank servers. Also required certificates needs to be installed at NPCI & Bank Site servers.

For API flow (Direct Debit card/Aadhaar/PAN/Cust ID) authentication below details are required.

1. NPCI to Bank connectivity with specified port
2. Bank SSL certificate (FQDNS is preferred)
3. URL's (mandate validation, verify OTP and resend OTP)

3. Specification Format for Request & Response

Appendix 7.1

Lists the XML file format for the request & response.

The specification for below request / response are listed in the document.

The data format would be XML. Schema structure and sample XML's can he found in Appendix 7.2.

- NPCI Mandate Request to Bank

NPCI ONMAGS will send the request to bank in the specified format

- Response from Bank to NPCI

Destination Bank will use this format for sending response bank to NPCI ONMAGS.

4. Technical Integration Specification

The below section lists a few of the technical requirements for the implementation.

4.1 Forward Flow specification from NPCI to Bank

This flow applies to Net Banking mode of authentication or for the Old Debit Card flow authentication.

Below are the steps done for securing the content of the Request data posted to the Bank from NPCI

1. The request XML to bank with all the tags present will be in the below format: -

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="http://npci.org/ONMAGS/schema">
   <MndtAuthReq>
         <GrpHdr>
               <NPCI_RefMsgId></NPCI_RefMsgId>
               <CreDtTm></CreDtTm>
               <ReqInitPty>
                     <Info>
                           <Id></Id>
                           <CatCode></CatCode>
                           <UtilCode></UtilCode>
                           <CatDesc></CatDesc>
                           <Name></Name>
                           <Spn_Bnk_Nm></Spn_Bnk_Nm>
                     </Info>
               </ReqInitPty>
         </GrpHdr>
         <Mndt>
               <MndtReqId></MndtReqId>
               <MndtId>UMRN</MndtId>
               <Mndt_Type></Mndt_Type>
               <Schm_Nm><Schm_Nm>
               <Ocrncs>
                     <SeqTp></SeqTp>
                     <Frqcy></Frqcy>
                     <FrstColltnDt></FrstColltnDt>
                     <FnlColltnDt></FnlColltnDt>
               </Ocrncs>
               <ColltnAmt Ccy="INR"></ColltnAmt>
               <MaxAmt Ccy="INR"></MaxAmt>
               <Dbtr>
                     <Nm></Nm>
                     <AccNo></AccNo>
                     <Acct_Type></Acct_Type>
                     <Cons_Ref_No></Cons_Ref_No>
                     <Phone></Phone>
                     <Mobile></Mobile>
                     <Email></Email>
```

```
                    <Pan></Pan>
                </Dbtr>
                <CrAccDtl>
                    <Nm></Nm>
                    <AccNo></AccNo>
                    <MmbId></MmbId>
                </CrAccDtl>
            </Mndt>
    </MndtAuthReq>
</Document>
```

2. The request XML to bank with all the tags present will be in the below format for AMEND:

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="http://npci.org/ONMAGS/schema">
    <MndtAuthReq>
        <GrpHdr>
            <MsgId></MsgId>
            <CreDtTm></CreDtTm>
            <ReqInitPty>
                <Info>
                    <Id></Id>
                    <CatCode></CatCode>
                    <UtilCode></UtilCode>
                    <CatDesc></CatDesc>
                    <Name></Name>
                    <Spn_Bnk_Nm></Spn_Bnk_Nm>
                </Info>
            </ReqInitPty>
        </GrpHdr>
        <Mndt>
            <MndtReqId></MndtReqId>
            <MndtId>YESB0000000000000023<MndtId>
            <Reason>AM05<Reason>
            <Schm_Nm><Schm_Nm>
            <Ocrncs>
                <SeqTp></SeqTp>
                <Frqcy></Frqcy>
                <FrstColltnDt></FrstColltnDt>
                <FnlColltnDt></FnlColltnDt>
            </Ocrncs>
            <ColltnAmt Ccy="INR"></ColltnAmt>
            <MaxAmt Ccy="INR"></MaxAmt>
            <Dbtr>
                <Nm></Nm>
                <AccNo></AccNo>
                <Acct_Type></Acct_Type>
                <Cons_Ref_No></Cons_Ref_No>
                <Phone></Phone>
                <Mobile></Mobile>
```

```
                    <Email></Email>
                    <Pan></Pan>
                </Dbtr>
                <CrAccDtl>
                    <Nm></Nm>
                    <AccNo></AccNo>
                    <MmbId></MmbId>
                </CrAccDtl>
            </Mndt>
    </MndtAuthReq>
</Document>
```

3. The request XML to bank for CANCEL, SUSPEND , REVOKE and Custom Cancel with all the tags present will be in the below format: -

```
<?xml version="1.0" encoding="UTF-8"?>
<Document
    xmlns="http://npci.org/ONMAGS/schema">
    <MndtAuthReq>
        <GrpHdr>
            <NPCI_RefMsgId></NPCI_RefMsgId>
            <CreDtTm></CreDtTm>
            <ReqInitPty>
                <Info>
                    <Id></Id>
                    <CatCode></CatCode>
                    <UtilCode></UtilCode>
                    <CatDesc></CatDesc>

                    <Name></Name>
                   <UtilCode></UtilCode>
                    <Spn_Bnk_Nm></Spn_Bnk_Nm>
                </Info>
            </ReqInitPty>
        </GrpHdr>
        <Mndt>
          <MndtReqId></MndtReqId>
          <MndtId>YESB0000000000000023<MndtId>
          <Reason>CN01<Reason>
          <Dbtr>
                      <AccNo></AccNo>
                  </Dbtr>
            <CrAccDtl>
                <Nm></Nm>
                <AccNo></AccNo>
                <MmbId></MmbId>
            </CrAccDtl>

        </Mndt>
    </MndtAuthReq>
</Document>
```

4. Generating checksum for the secure information in the XML

   The below attributes needs to be concatenated for the purpose of generating Checksum:

   - Debtor Account Number
   - First Collection Date
   - Final Collection Date
   - Collection Amount
   - Max Amount

   The above attributes need to be concatenated with "|" symbol appended as the delimiter. The order of the attributes needs to be as mentioned above. In case any of the attribute is null then during concatenation the particular attribute will be replaced by an empty string.

   Note:
   The attributes to be concatenated might be changed at later point of time. Please refer the latest version of the document for any revision on the attributes that needs to be marked for encryption.

   Generate checksum on the concatenated values. We will use SHA-2 as the hash function.

5. Replace the secure information in the XML with the encrypted text. Below are the attributes which will be encrypted in the request XML

   - Debtor Account Number
   - First Collection Date
   - Final Collection Date
   - Collection Amount
   - Max Amount
   - Phone
   - Mobile
   - Email
   - Pan

   The attributes mentioned above needs to be encrypted individually and placed in the respective XML tags. We will use the below methodology for encryption of secure information.

   Encryption Methodology – Asymmetric

   Hashing Algorithm – SHA256

   Cryptography – RSA/ECB/OAEPWithSHA-256AndMGF1Padding 2048 bits.

   Encryption will be done using the Public Key of the certificate shared by Bank.

6. Signing of the Request XML

   The request XML got from Step-2 will be signed using the Private Key certificate of NPCI.

NPCI will send the below data as MIME content to Merchant with type as “application/x-www-form-urlencoded” in the request body.

Note :Checksum is not required for Cancel,Suspend,Revoke and Custom Cancel flows.

Key	Value
MandateReqDoc	Output of the Step-3
CheckSumVal	Encrypted Output of Step-1 (only for Create and Amend)

4.1.1 Encoding of Request XML for Banks

The request XML from NPCI to Bank will be encoded to prevent any malicious attack. Banks will need to accept the encoded xml content at their end then decode it to get the original content.

The encoded request XML will look as below: -

```
&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;
&lt;Document xmlns=&quot;http://npci.org/ONMAGS/schema&quot;&gt;
   &lt;MndtAuthReq&gt;
         &lt;GrpHdr&gt;
               &lt;NPCI_RefMsgId&gt;&lt;/NPCI_RefMsgId&gt;
               &lt;CreDtTm&gt;&lt;/CreDtTm&gt;
               &lt;ReqInitPty&gt;
                     &lt;Info&gt;
                           &lt;Id&gt;&lt;/Id&gt;
                           &lt;CatCode&gt;&lt;/CatCode&gt;
                           &lt;UtilCode&gt;&lt;/UtilCode&gt;
                           &lt;CatDesc&gt;&lt;/CatDesc&gt;
                           &lt;Name&gt;&lt;/Name&gt;
                           &lt;Spn_Bnk_Nm&gt;&lt;/Spn_Bnk_Nm&gt;
                     &lt;/Info&gt;
               &lt;/ReqInitPty&gt;
         &lt;/GrpHdr&gt;
         &lt;Mndt&gt;
               &lt;MndtReqId&gt;&lt;/MndtReqId&gt;
               &lt;MndtId&gt;UMRN&lt;/MndtId&gt;
               &lt;Mndt_Type&gt;&lt;/Mndt_Type&gt;
               &lt;Schm_Nm&gt;&lt;Schm_Nm&gt;
               &lt;Ocrncs&gt;
                     &lt;SeqTp&gt;&lt;/SeqTp&gt;
                     &lt;Frqcy&gt;&lt;/Frqcy&gt;
```

```
                    &lt;FrstColltnDt&gt;&lt;/FrstColltnDt&gt;
                    &lt;FnlColltnDt&gt;&lt;/FnlColltnDt&gt;
               &lt;/Ocrncs&gt;
               &lt;ColltnAmt
Ccy=&quot;INR&quot;&gt;&lt;/ColltnAmt&gt;
               &lt;MaxAmt Ccy=&quot;INR&quot;&gt;&lt;/MaxAmt&gt;
               &lt;Dbtr&gt;
                    &lt;Nm&gt;&lt;/Nm&gt;
                    &lt;AccNo&gt;&lt;/AccNo&gt;
                    &lt;Acct_Type&gt;&lt;/Acct_Type&gt;
                    &lt;Cons_Ref_No&gt;&lt;/Cons_Ref_No&gt;
                    &lt;Phone&gt;&lt;/Phone&gt;
                    &lt;Mobile&gt;&lt;/Mobile&gt;
                    &lt;Email&gt;&lt;/Email&gt;
                    &lt;Pan&gt;&lt;/Pan&gt;
               &lt;/Dbtr&gt;
               &lt;CrAccDtl&gt;
                    &lt;Nm&gt;&lt;/Nm&gt;
                    &lt;AccNo&gt;&lt;/AccNo&gt;
                    &lt;MmbId&gt;&lt;/MmbId&gt;
               &lt;/CrAccDtl&gt;
          &lt;/Mndt&gt;
    &lt;/MndtAuthReq&gt;
&lt;/Document&gt;
```


4.2 Bank Site Integration Requirements

4.2.1 Net Banking Flow

In case of Netbanking or if the Bank has opted for the old debit card flow, then NPCI ONMAGS would redirect to Bank Page. The URL for redirection for Net banking should be made available to NPCI by the banks. NPCI will pass the XML content mentioned in the sheet (“NPCI Mandate Request to Bank”) & CheckSumVal as part of the request.

The request body will contain the following key-value pair.

Key	Value
MandateReqDoc	Encrypted and Signed XML
CheckSumVal	Encrypted Checksum Hash value (only for Create and Amend)

Specifics on Signing, Encryption and Checksum are mentioned in the section 4.2.1

Bank site should unsign the XML using the public key of NPCI and then decrypt the key fields using the private key of the Bank. Checksum should be decrypted using the private key of the Bank. In case of any errors during unsigning, decryption or checksum validation, Bank needs to construct the Error response in the format "ErrorXML Resp from Bank to NPCI".

Below are the validations done at Bank layer for the request received from NPCI. For more details refer to sheet "NPCI Mandate Request to Bank" in the excel "NPCI Mandate Authorization Specification for Banks.xlsx" available in the Appendix Section.

Element Name	Validation	Data Type	Length	Remarks
xmlns	Namespace tag. This is mandatory tag. Value cannot be empty. Namespace value should be "http://npci.org/ONMAGS/schema"	Alpha Numeric		
NPCI_RefMsgId	NPCI_RefMsgId from NPCI should be unique	Alpha Numeric	35	Message ID for NPCI Reference
CreDtTm	Should be in ISO Date time format. E.g.2017-02-09T15:11:39	Alpha Numeric	25	
ID	Request Initiating Party ID. In this case it will be Corporate / Merchant ID. Should not be null. Will be validated if this is a valid Merchant ID with the master.	Alpha Numeric	18	ID & UtilCode value would be the same.
UtilCode	Utility Code would be validated against the masters. It should be 7 digit OLD ICS or 18 digit Utility code.	Alpha Numeric	18	ID & UtilCode value would be the same.
CatCode	Identifies under which category the mandate is created. Will be validated against the masters maintained by NPCI	Alpha Numeric	4	
Name	Should not be empty	Alpha Numeric	40	Corporate Name.
Spn_Bnk_Nm	Corporate Sponsor Bank Name	Alpha Numeric	140	Should be a valid Bank Name as per MMS
CatDesc	Category Description should correspond to Category Code in the Master	Alpha Numeric	50	
MndtReqId	Mandate Req ID length should be <= 35. Should be unique for the day	Alpha Numeric	35	
MndtId	This tag will contain the UMRN generated in MMS for the mandate.	Alpha Numeric	35	UMRN

Mndt_Type	Mandate Type	Alpha Numeric	35	Should be DEBIT
Schm_Nm	Scheme Name / Plan Reference Number	Alpha Numeric	20	
SeqTp	Allowed values are RCUR or OOFF	Alpha Numeric	4	
Frqcy	This is an optional field. If present should adhere to the list value available in MMS Masters.	Alpha Numeric	4	Allowed Values are: ADHO, INDA, DAIL, WEEK, MNTH, QURT, MIAN, YEAR, BIMN
FrstColltnDt	Date of First Collection. Mandatory Field. This field is in ISODate Format	Alpha Numeric	16	
FnlColltnDt	Date of Final Collection. Optional Field. This field is in ISODate Format	Alpha Numeric	16	If this field is left blank then deduction will happen until Cancelled.
ColltnAmt	Either of ColltnAmt or MaxAmt is mandatory. Amount Should be given as 100.00	Alpha Numeric	13	
MaxAmt	Either of ColltnAmt or MaxAmt is mandatory Amount Should be given as 100.00	Alpha Numeric	13	
Debtor Nm	Customer name should be maximum of 40 digit	Alpha Numeric	40	
Debtor AccNo	Customer Account Number should be maximum of 35 digit.	Alpha Numeric	35	
Acct_Type	Debtor Account Type	Alpha Numeric	35	Should be either of SAVINGS or CURRENT
Cons_Ref_No	Consumer Reference Number	Alpha Numeric	35	
Phone	Phone Number of the Customer	Alpha Numeric	16	Should be given in the format +91-xxx-xxxxxxxx. +91- is mandatory.
Mobile	Mobile Number of the Customer	Alpha Numeric	14	Should be given in the format +91-xxxxxxxxxx. +91- is mandatory.
Email	Email ID of the Customer	Alpha Numeric	50	Should be valid email id
Pan	Pan Number of the Customer	Alpha Numeric	10	Should be in Valid PAN format
Creditor Nm	Corporate Name. Length will be 40	Alpha Numeric	140	
Creditor AccNo	Will be the 18 digit Corporate ID	Alpha Numeric	18	

MmbId	Will be 11 digit IFSC code	Alpha Numeric	11	IFSC Code of the Sponsor Bank which is available in the ONMAG Live Bank list
Reason	Reason for Amend / Cancel / Suspend / Revoke / Custom cancel from MMS system	Alpha Numeric	4	

End user would enter his/her net banking credentials information in the authentication page of the bank. An SMS OTP validation also has to be done as second level authentication.

Upon making a successful login bank should first validate whether the bank account number passed in the request XML matches the bank account number of the authenticated end user. If the bank account number does not match the customer would not be allowed to proceed further. Appropriate error message needs to be displayed to the customer and a link provided to return back to the merchant site.

If the customer is not able to make a successful login after predetermined login attempts the Bank has to redirect back to the NPCI ONMAGS layer. The reject reason will be “Invalid Login Credentials”.

If the account number matches, then the customer needs to be shown a form which displays specific details of the mandate and a “Terms & Policy” section displaying terms and policies of the bank. A confirmation check box needs to be provided for end user for agreeing to the displayed information.

The below information needs to be mandatorily displayed to the User at the Bank end:

- Mandate request Initiate Party’s Category Description (“CatDesc”) (Only for Create and Amend)
- Name of Initiator (In all operation like create, amend, cancel, suspend and revoke)
- Collection Amount (Only for Create and Amend)
- Max Amount (Only for Create and Amend)
- Recurring Frequency (Only for Create and Amend)
- First Collection Date (Only for Create and Amend)
- Final Collection Date (Only for Create and Amend)
- UMRN (In all operation like create, amend, cancel, suspend and revoke)

User has to be provided links for either accepting the mandate or Rejecting the mandate. On selection of either of the option the user would be redirected to the NPCI ONMAGS interface. The response should contain the XML mentioned in the sheet “Response from Bank to NPCI”. The element <AccptncRslt> will contain the result of the approval status of the mandate. The URL for redirection to NPCI ONMAGS interface would be shared by NPCI.

The response body will contain the following key-value pair. Bank will send the below data as MIME content to NPCI with type as “application/x-www-form-urlencoded”.

Key	Value
BankID	Participant ID of the Bank in NACH
MandateRespDoc	Encrypted and Signed XML
CheckSumVal	Encrypted Checksum Hash value (only for Create and Amend)
RespType	Will be either of ErrorXML / RespXML
mndtType	Mandate type of Request

Note : mndtType will not be present for Create Flow

Below are the steps to be done for securing the content of the Response XML:

1. Generating checksum for the secure information in the XML

   The below attributes needs to be concatenated for the purpose of generating Checksum:

   a) Accptd
   b) AccptRefNo
   c) ReasonCode
   d) ReasonDesc
   e) RejectBy

   The above attributes need to be concatenated with "|" symbol appended as the delimiter. The order of the attributes needs to be as mentioned above.

   Note:
   The attributes to be concatenated might be changed at later point of time. Please refer the latest version of the document for any revision on the attributes that needs to be marked for

   Generate checksum on the concatenated values. We will use SHA-2 as the hash function.

2. Replace the secure information in the XML with the encrypted text.

   The attributes mentioned above needs to be encrypted individually and placed in the respective XML tags. Encryption should be done using the public key of the certificate which NPCI shares.

   We will use the below methodology for encryption of secure information.

Encryption Methodology – Asymmetric

Hashing Algorithm – SHA256

Cryptography – RSA/ECB/OAEPWithSHA-256AndMGF1Padding 2048 bits

Encryption needs to be done using the Public Key of the certificate shared by NPCI.

3. Signing of the Response XML

   The response XML got from Step-2 has to be signed using the Private Key certificate of the Bank.

The below are the validation done at NPCI ONMAGS layer for the response received from Bank. For more details refer to sheet "Response from Bank to NPCI" in the excel "NPCI Mandate Authorization Specification for Banks" available in the Appendix Section.

Element Name	Validation	Length	Remarks
Xmlns	Namespace tag. This is mandatory tag. Value cannot be empty. Namespace value should be "http://npci.org/ONMAGS/schema"		
MsgId	This is a reference generated by the bank to identify the response message. Should be unique for the day for a Bank	35	
GrpHdr - CreDtTm	Should be in ISO Date time format. E.g.2017-02-09T15:11:39	25	
ReqInitPty	Request Initiating Party ID. This will refer to the Bank Short Code	18	
MndtReqId	Mandate Request ID should be same as the MndtReqId send in the original request to Bank	35	
NPCI_RefMsgId	Message ID for NPCI Reference in the original request. Should be same as the NPCI_RefMsgId send in the original request to Bank	35	
OrgnlMsgInf - CreDtTm	Creation Date Time send in the original request to Bank	18	
MsgNmId	Both the tag & value are optional		
Accptd	Mandatory. Allowed values are true / false	5	Indicates whether the mandate request was accepted or rejected.
AccptRefNo	Will be non-empty if accptd is true. Should be unique for the Bank. If accptd is false empty value can be provided.	34	Accepted Reference Number.

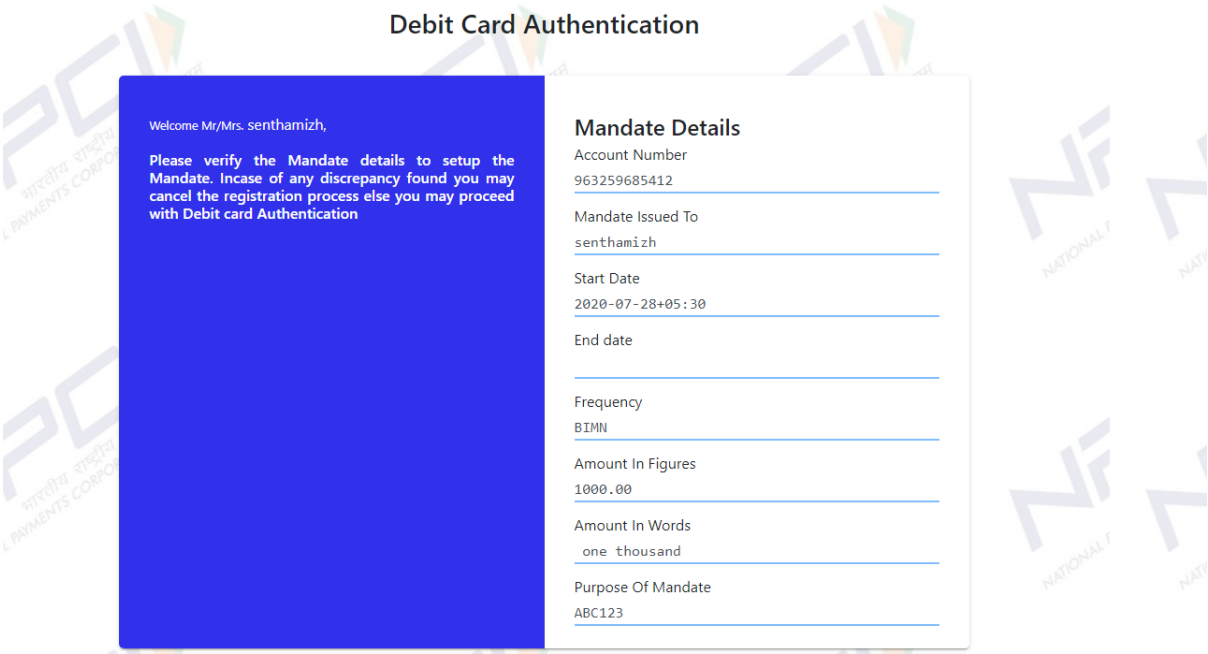
ReasonCode	Mandatory if <Accptd> value is false. Reason code should be as per master provided by NPCI.	5	If acceptance is false, reason code of rejection is entered here. If acceptance is true then this value would be “N/A”.
ReasonDesc	Mandatory. Reason Description should match the Reason Code specified by NPCI.	50	If acceptance is false, reason description of rejection is entered here. If acceptance is true then this value would be “N/A”.
RejectBy	Mandatory. Should be either of “BANK” or “USER” or “N/A”	10	If acceptance is true then this value would be “N/A”.
IFSC	Mandatory if <Accptd> tag value is true, IFSC of the destination bank	11	

4.2.2 New Debit Card Flow

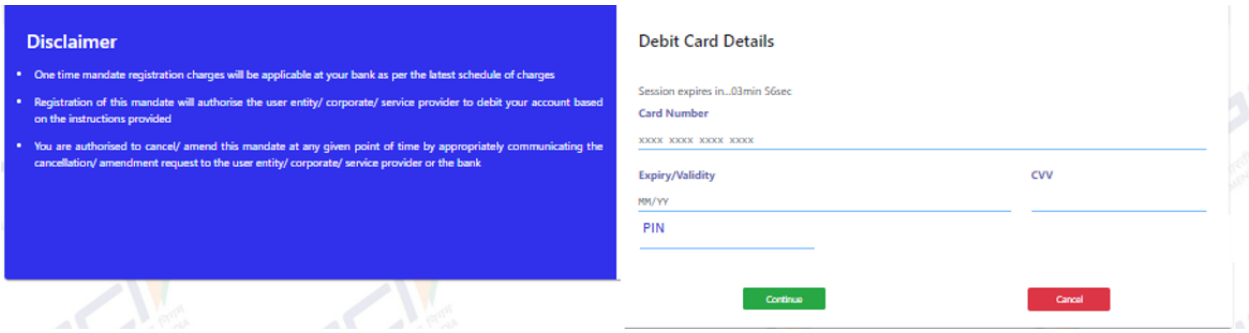
In case Bank has opted for the new Debit Card Flow, then from the merchant site, the user will be landing on the NPCI’s ONMAGS Debit Card authentication page.

Debit Card Information will be accepted in ONMAGS page itself and validated with Bank through server to server call. The steps in this flow is described below:

The mandate information passed by the merchant will be displayed in the top portion of the page.



User needs to verify the mandate information displayed in the Mandate Details section. Once mandate information are verified by the user he/she can proceed with entering the Debit Card information in the lower section of the page.



Below are the validation done related to the entered Debit Card Details

- Card Number should be 16 digit Numerical.
- Expiry Year and Month should be current month or future year month.
- Expiry period cannot be greater than 10 years.
- CVV should be 3 digit Numeric.
- PIN number 4 to 6 digit numeric field
- CVV/PIN/BOTH is mandatory based on the banks preference (Banks can opt for CVV+PIN (or) either CVV/PIN)

On entering the Debit Card Information user can click on Continue, to proceed with Debit Card Verification.

In case the user does not want to proceed further with authentication then he/she can click on Cancel. On clicking on Cancel, the transaction will be cancelled, merchant response gets generated redirects to the Merchant response page.

4.2.2.1 Request Information to Bank

Once the User clicks on Continue, ONMAGS will construct the below JSON request and post to the Bank. Both the Mandate Details and the Card Information will be passed in the request. The request will be made as an API call to the bank and will happen as a server to server call. The response to the API call has to be provided in a synchronous manner by the bank.

JSON Request with Mandate and Card information

```
{
   "mandateAuthDtls": {
      "transactionID": "<Transaction ID>",
      "mndtType":"<AMEND /CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL attributes>",
      "mandateRequestDtl": {
         "MandateReqDoc": "<Encrypted and Signed response XML>",
```

```
            "CheckSumVal": "<Check sum value of secure attributes>"
        },
        "cardInfo": {
            "cardNo": "<Encrypted Card Number>",
            "expiry": "<Encrypted expiry Date>",
            "cvv": "<Encrypted CVV>",
             "pin": "<Encrypted pin>"
        }
    }
    }
```

Note:

Based on the banks the pin / cvv / pin and cvv will be present. Pin length can be configured as 4 or 6 based on the bank.

- mndtType will not be present for Create Flow

- MandateReqDoc XML will be encoded and sent.
- For encryption the existing logic and keys will be used (i.e, NPCI will do the encryption using the Public Key provided by the bank and Bank will do the decryption using their private key).
- Pin attribute will be encrypted using separate public key and bank will do decryption using the respective private key.

Bank needs to first verify the mandate request details and then the card details and provide the response in any of the below formats.

A. If the destination bank is unable to parse the mandate request it will send the response in the below format. Bank need not validate the card details if sending failure response (because of request XML validation failure at bank end).

```
{
    "mandateVerifyDtls": {
        "transactionID": "<Transaction ID>",
        "mndtType":"<AMEND /CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL attributes>",
        "mandateValidation": "failure",
        "cardValidation": "none",
        "mandateRejectDtl": {
            "ErrorCode": "<Error Code>",
            "ErrorDesc": "<Error Description>"
        }
    }

}
```

Note:-

Attribute values mandateValidation, cardValidation, ErrorCode & ErrorDesc needs to be encrypted. Bank needs to encrypt using NPCI public key.

B. If destination bank is able to successfully parse the mandate request XML but business validation of XML fails, then bank needs to send the response in the below format. Card details need not be validated in such a scenario.

```
{
    "mandateVerifyDtls": {
        "transactionID": "<Transaction ID>",
        "mndtType":"<AMEND /CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL attributes>",
```

```
            "mandateValidation": "failure",
            "cardValidation": "none",
            "mandateRejectDtl": {
               "ReasonCode": "<Reason Code>",
               "ReasonDesc": "<Reason Description>"
            }
         }
      }
```

Note:-

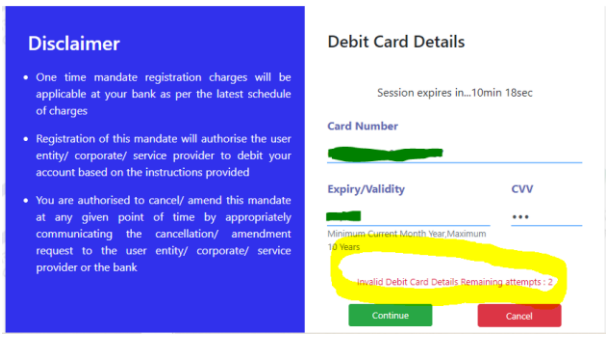
Attribute values mandateValidation, cardValidation, ReasonCode & ReasonDesc needs to be encrypted

I. If the destination bank is able to successfully parse the mandate request XML and business validation passes, then bank needs to validate the card details. Bank needs to verify the below details:Verify Debit Card Number
II. Verify Expiry / validity
III. Verify CVV number
IV. Verify PIN number (if its applicable)
V. Account number of debit card matches with the “Debtor AccNo” provided in the mandate Request XML

If any of the above validation fails, then the bank needs to provide the response as below: -

```
{
   "mandateVerifyDtls": {
      "transactionID": "<Transaction ID>",
      “mndtType”:”<AMEND /CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL attributes>”,
      "mandateValidation": "success",
      "cardValidation": "failure",
      "mandateResponseDtl": {
         "accptRefNo": "<Accept Reference Number>",
          "dbtrIfsc": "<Debtor IFSC>",
          "dbtrAcctType": "<Debtor Account Type>"
      },
      "cardVerifyDtl": {
         "ErrorCode": "<Error Code>"
      }
   }
}
```

☒ If Card validation is failure User would be provided with option of reattempting Card validation further 2 times. An alert message as Invalid Debit Card Details Remaining attemts:2 will appear on the screen. User can proceed by entering the correct Card details again and continue.



Note:-

Attribute values mandateValidation, cardValidation, AccptRefNo & ErrorCode needs to be encrypted.

The below table provides the error codes details that are newly introduced for Direct Debit Card Flow

error_code_id	error_desc	applicable_leg
601	Invalid Debit Card Number	BTN
602	Invalid Expiry / Validity	BTN
603	Invalid CVV	BTN
604	Debit card information is not matched with the associated account number	BTN
605	Otp Verification Failure	BTN
606	Duplicate Request	MTN
607	Previous Request in Progress	MTN
608	Bank Restricts Duplicate request	MTN
609	Invalid PIN	BTN

C. If all the above validation passes, then the bank needs to provide the response as below: -

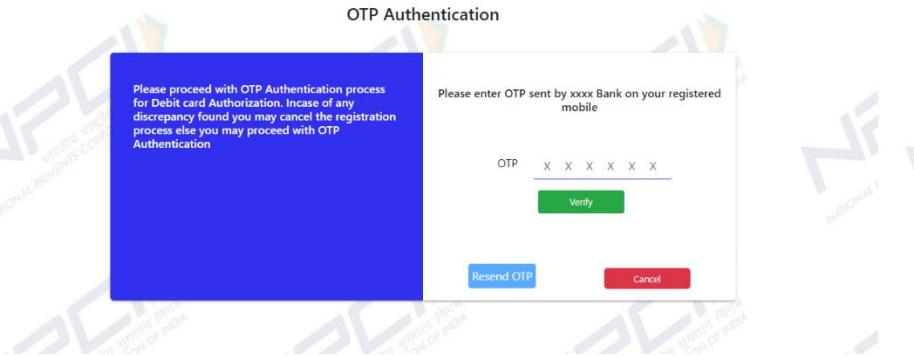
```
{
   "mandateVerifyDtls": {
      "transactionID": "<Transaction ID>",
      "mndtType":"<AMEND /CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL attributes>",
      "mandateValidation": "success",
      "cardValidation": "success",
      "mandateResponseDtl": {
         "accptRefNo": "<Accept Reference Number>",
          "dbtrIfsc": "<Debtor IFSC>",
          "dbtrAcctType": "<Debtor Account Type>"
      },
      "cardVerifyDtl": {
         "successCode": "<Success Code>"
      }
   }
}
```

Note:-

- Attribute values mandateValidation, cardValidation, AccptRefNo & successCode needs to be encrypted
- IFSC code is optional field, if banks give invalid IFSC code in the Response ONMAGS system will update the IFSC code as per the Bank Masters
- Bank needs to store the mandate details received along with the transaction ID for the subsequent OTP validation.

For scenarios (a), (b) & (c) ONMAGS will construct the merchant rejection response and redirect to the merchant. Bank needs to mark the mandate as rejected at their end for these scenarios. For scenario (d) mandate status will be "In Process" for the bank until the OTP verification is completed.

For scenario (d) ONMAGS will redirect to the OTP verification page.



- OTP will be a 6 digit numeric number.
- In case User did not receive OTP, there is an option to Resend OTP which the user can retry maximum of 3 times.
- Once user clicks on the verify button the entered OTP is encrypted and sent to the server. From the server end VerifyOTP API call will be made to the bank server.

```
{
   "otpInfo":
      {
         "transactionID": "<Transaction ID>",
         "otp": "<Encrypted OTP Value>"
      }
}
```

- In case of retry as well the request will be posted to bank in the above mentioned format only.

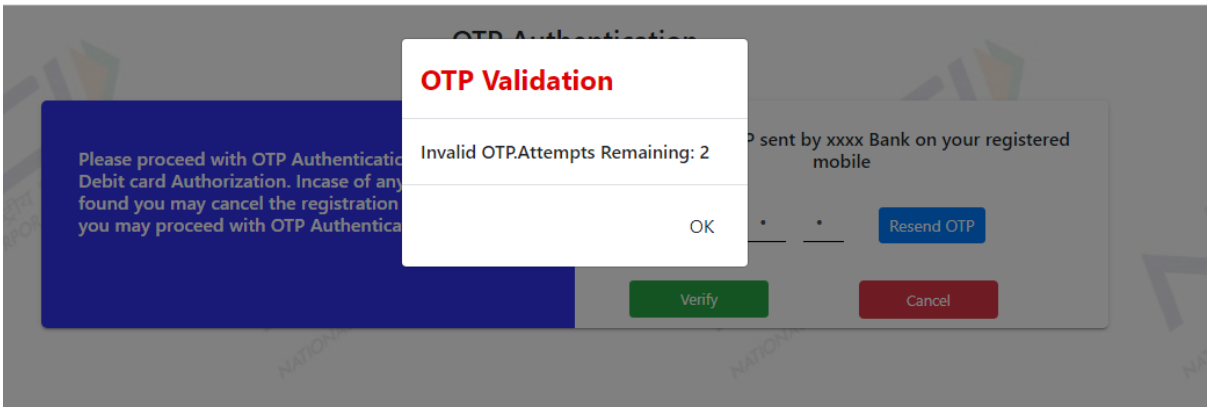
The encryption on the OTP will follow the existing encryption methodology. Bank needs to decrypt the OTP and verify it based on the transaction ID. The OTP verification status needs to be sent in the below json format by the bank.

```
{
   "otpVerifyInfo":
      {
         "transactionID": "<Transaction ID>",
         "optVerifyStatus": "<Encrypted OTP verification status. It will be either
success / failure>"
```

```
        }
}
```

If OTP verification is successful only Bank needs to mark the mandate as accepted at their end. Until OTP validation is passed the mandate would be in non-accepted state at the Bank end.

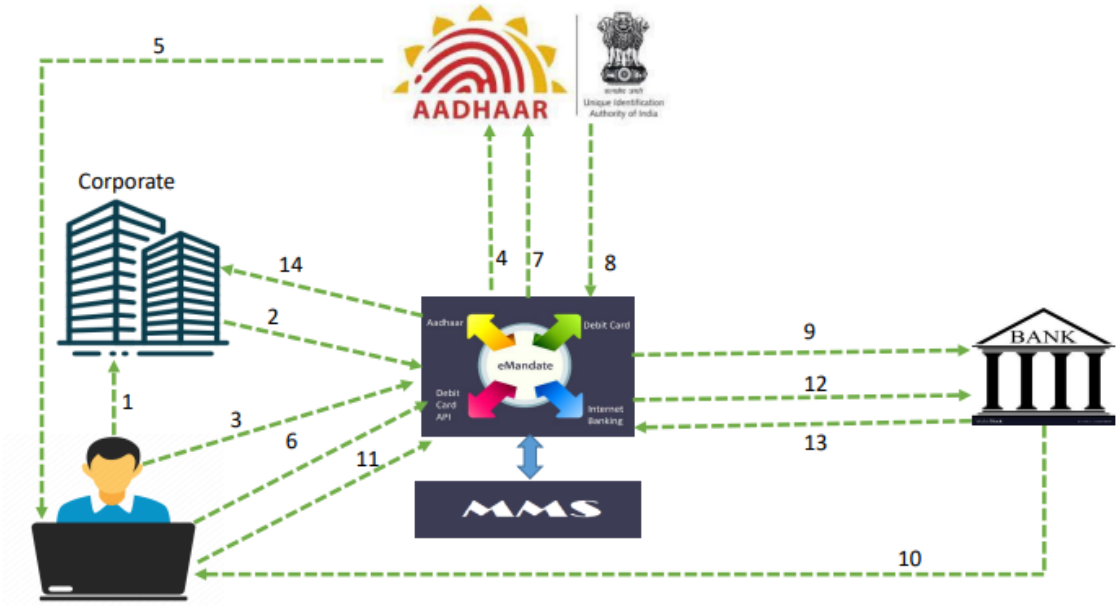
If OTP validation is failure User would be provided with option of reattempting OTP validation further 2 times. An alert message as below will be shown to the user. User can then proceed with entering the correct OTP again and re-verify.



Resend OTP:

```
{
  "debitAuthDtls": {
    "transactionID": "",
    "cardInfo": {
      "cardNo": "",
      "expiry": "",
      "cvv": ""
      "pin": "<Encrypted pin>"
    }
  }
}
```

4.2.3 Aadhaar Based Authentication Flow



Step1: Customer has initiated the request via Merchant Portal i.e., Web Browser

Step2: Customer will be redirected to ONMAGS Platform to enter the details required for Aadhaar authentication.

Step3: Customer enters Aadhaar Number along with required details.

Step4: ONMAGS Platform will forward that request to UIDAI for Customer Authentication via OTP generation

Step5: UIDAI will generate the OTP and send it to Customer's registered mobile number for Authentication

Step6: Customer will enter the OTP in the ONMAGS OTP page

Step7: ONMAGS Platform will forward that OTP to UIDAI for Verification

Step8: UIDAI sends response for OTP Verification. If the request is not authenticated by UIDAI then the flow ends here by showing the error message in Merchant Portal.

Step9: Once the customer is successfully authenticated, then the ONMAGS platform will send the mandate request to destination bank. If customer bank doesn't opt for additional OTP authentication then skip Step 10, Step 11 and Step 12.

Step10: After customer successfully authenticated by UIDAI, he/she will be landed on ONMAGS OTP page. ONMAGS will send an API request to customer's Bank to verify the customer details will generate the Bank OTP and send it to customer for Authentication.

Step11: Customer will enter the Bank OTP in ONMAGS platform for Authentication.

Step12: ONMAGS platform will forward that OTP to destination bank for Verification.

Step13: If OTP verification is successful only Bank needs to mark the mandate as accepted at their end. Until OTP validation is passed the mandate would be in non-accepted state at the Bank end.

Step 14: ONMAGS Platform in turn redirects the response to Merchant Web Page where customer can view the response.

Auth mode: Aadhaar

Privilege: Initiated by ONMAGS (NPCI)

API type: Sync

Request Type: JSON

HTTP Method: POST

Parameter Specification

Parameters	Data Type	Description
mandateAuthDtls	JSON Object	This will contains mandate Request details and aadhaar Info
transactionID	String	This is used for the complete transaction for mandate registration. ALPNUM String with Length is 20.
mandateRequestDtl	JSON Object	This will contains Encrypted mandate Request Doc XML and Encrypted checksum value
MandateReqDoc	String	See below table for Mandate Request Doc.
CheckSumVal	String	How to generate Checksum value is mentioned above.

authMode	String	If Authmode is null, then user will get cardInfo JSON Object and consider as authomode as Debit Card else authMode value will be Aadhaar and user will get aadhaarInfo JSON Object in request.
aadhaarInfo	JSON Object	This will contains the aadhaar details and flag indicating that the customer authentication has been successful though UIDAI.
aadhaarNo	String	Last four digit of aadhaar number
uidaiAuthenticated	Char	Always Y to be sent

Mandate Request to Bank:

```
{
   "mandateAuthDtls": {
      "transactionID": "<Transaction ID>",
      "mndtType":"<AMEND /CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL attributes>",
      "mandateRequestDtl": {
         "MandateReqDoc": "<Encrypted and Signed request XML>",
         "CheckSumVal": "<Check sum value of secure attributes>"
      },
      "authMode":"Aadhaar",
      "aadhaarInfo": {
         "aadhaarNo": "<Encrypted Aadhaar Number>",
         "uidaiAuthenticated" : "Y"
      }
   }
}
```

Unencrypted and Unsigned request XML for MandateReqDoc Key:

Element Name	Validation	Data Type	Length	Remarks

Xmlns	Namespace tag. This is mandatory tag. Value cannot be empty. Namespace value should be "http://npci.org/ONMAGS/schema"	Alpha Numeric		
NPCI_RefMsgId	NPCI_RefMsgId from NPCI should be unique	Alpha Numeric	35	Message ID for NPCI Reference
CreDtTm	Should be in ISO Date time format. E.g.2017-02-09T15:11:39	Alpha Numeric	25	
ID	Request Initiating Party ID. In this case it will be Corporate / Merchant ID. Should not be null. Will be validated if this is a valid Merchant ID with the master.	Alpha Numeric	18	ID & UtilCode value would be the same.
UtilCode	Utility Code would be validated against the masters. It should be 18 digit Utility code.	Alpha Numeric	18	ID & UtilCode value would be the same.
CatCode	Identifies under which category the mandate is created. Will be validated against the masters maintained by NPCI	Alpha Numeric	4	
Name	Should not be empty	Alpha Numeric	40	Corporate Name.
Spn_Bnk_Nm	Corporate Sponsor Bank Name	Alpha Numeric	140	Should be a valid Bank Name as per MMS
CatDesc	Category Description should correspond to Category Code in the Master	Alpha Numeric	50	
MndtReqId	Mandate Req ID length should be <= 35. Should be unique for the day	Alpha Numeric	35	

MndtId	This tag will contain the UMRN generated in MMS for the mandate.	Alpha Numeric	20	UMRN
Mndt_Type	Mandate Type	Alpha	35	Should be DEBIT
Schm_Nm	Scheme Name / Plan Reference Number	Alpha Numeric	20	
SeqTp	Allowed values are RCUR or OOFF	Alpha Numeric	4	
Frqcy	This is an optional field. If present should adhere to the list value available in MMS Masters.	Alpha Numeric	4	Allowed Values are: ADHO, INDA, DAIL, WEEK, MNTH, QURT, MIAN, YEAR, BIMN
FrstColltnDt	Date of First Collection. Mandatory Field. This field is in ISODate Format	Alpha Numeric	16	
FnlColltnDt	Date of Final Collection. Optional Field. This field is in ISODate Format	Alpha Numeric	16	If this field is left blank then deduction will happen until Cancelled.
ColltnAmt	Either of ColltnAmt or MaxAmt is mandatory. Amount Should be given as 100.00	Alpha Numeric	13	
MaxAmt	Either of ColltnAmt or MaxAmt is mandatory Amount Should be given as 100.00	Alpha Numeric	13	
Debtor Nm	Customer name should be maximum of 35 digit	Alpha Numeric	40	

Debtor AccNo	Customer Account Number should be maximum of 35 digit.	Alpha Numeric	35	
Acct_Type	Debtor Account Type	Alpha	35	Should be either of SAVINGS or CURRENT
Cons_Ref_No	Consumer Reference Number	Alpha Numeric	20	
Phone	Phone Number of the Customer	Alpha Numeric	34	Should be given in the format +91-xxx-xxxxxxxx. +91- is mandatory.
Mobile	Mobile Number of the Customer	Alpha Numeric	34	Should be given in the format +91-xxxxxxxxxx. +91- is mandatory.
Email	Email ID of the Customer	Alpha Numeric	50	Should be valid email id
Pan	Pan Number of the Customer	Alpha Numeric	27	Should be in Valid PAN format
Creditor Nm	Corporate Name. Length will be 40	Alpha Numeric	140	
Creditor AccNo	Will be the 18 digit Corporate ID	Alpha Numeric	18	
MmbId	Will be 11 digit IFSC code	Alpha Numeric	11	IFSC Code of the Sponsor Bank which is available in the ONMAG Live Bank list
MndtId	Will be 20 digit UMRN	Alpha Numeric	20	Except Create Flow

ReasonCode	will be 4 digit Reason code	Alpha Numeric	4	Except Create Flow

Bank needs to first verify the mandate request details

a) If the destination bank is unable to parse the mandate request it will send the response in the below format. Bank need not validate the aadhaar details if sending failure response (because of request XML validation failure at bank end).

Parameters	Datatypes	Description
mandateVerifyDtls	JSON Object	Mandate verify details contains transaction ID, mandate Validation and mandate reject details
transactionID	String	This is the same transaction ID which Is passed in request for mandate registration. ALPNUM String with Length is 20.
mndtType	String	This will contain the operation AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL. mndtType will not be present for Create Flow.
mandateValidation	String	This will return either success or failure.
aadhaarValidation	String	This will return either success or failure.
mandateRejectDtl	JSON Object	This will contain error code and error desc
ErrorCode	Integer	This will be between 000 to 999

ErrorDesc	String	This will be the corresponding error description for the error code.
signature	String	The Response payload will be signed with bank's private key and algorithm used as ***RSA_USING_SHA256***
checkSumVal	String	Generate checksum on the entire payload. We will use SHA-2 as the hash function

Error Response from Bank for Mandate request:

```
{
   "mandateVerifyDtls": {
      "transactionID": "<Transaction ID>",
      "mndtType":"<AMEND /CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL attributes>",
      "mandateValidation": "failure",
      "aadhaarValidation": "none",
      "mandateRejectDtl": {
         "ErrorCode": "<Error Code>",
         "ErrorDesc": "<Error Description>"
      }
},
      "signature": "<Encrypted and Signed response JSON>",
      "checkSumVal": "<Check sum value of complete payload>"


}
```

Note:-

Attribute values Mandate Validation, Aadhaar Validation, Error Code & ErrorDesc needs to be encrypted. Bank needs to encrypt using NPCI public key.

b) If destination bank is able to successfully parse the mandate request XML but business validation of XML fails, then bank needs to send the **response** in the below format. Aadhaar details need not be validated in such a scenario.

```
{
   "mandateVerifyDtls": {
      "transactionID": "<Transaction ID>",
      "mndtType":"<AMEND /CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL attributes>",
      "mandateValidation": "failure",
      "aadhaarValidation": "none",
```

```
        "mandateRejectDtl": {
           "ReasonCode": "<Reason Code>",
           "ReasonDesc": "<Reason Description>"
        }
},
      "signature": "<Encrypted and Signed response JSON>",
      "checkSumVal": "<Check sum value of complete payload>"


}
```

Note:-

Attribute values Mandate Validation, Aadhaar Validation, Reason Code,Reason Desc & checkSumVal needs to be encrypted

c) Aadhaar Validation

1. Aadhaar number of debtor should matches with the "Aadhaar linked with the Debtor AccNo" provided in the mandate Request XML
2. Aadhaar number should linked with the debtor Account Number.

If the above validation fails then the bank needs to provide the response as above format 2nd type.

```
{
   "mandateVerifyDtls": {
      "transactionID": "<Transaction ID>",
      "mndtType":"<AMEND /CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL attributes>",
      "mandateValidation": "success",
      "aadhaarValidation": "failure",
      "mandateResponseDtl": {
         "accptRefNo": "<Accept Reference Number>",
          "dbtrIfsc": "<Debtor IFSC>",
          "dbtrAcctType": "<Debtor Account Type>"
      },
      " aadhaarRejectDtl ": {
         "ReasonCode": "<Reason Code>"
      }
},
       "signature": "<Encrypted and Signed response JSON>",
       "checkSumVal": "<Check sum value of complete payload>"

}
```

The below table provides the error codes for different failure reasons.

Failure Reason	Reason Code
Aadhaar number Does not Match with debtor Account number	AP48
Aadhaar number not linked with the debtor Account number	AP51

d. If all the above validation passes then the bank needs to provide the success **response** as below:-

Success Response for Mandate request to Bank:

```
{
   "mandateVerifyDtls": {
      "transactionID": "<Transaction ID>",
      "mndtType":"<AMEND /CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL attributes>",
      "mandateValidation": "success",
       "aadhaarValidation": "success",
      "mandateResponseDtl": {
         "accptRefNo": "<Accept Reference Number>",
          "dbtrIfsc": "<Debtor IFSC>",
          "dbtrAcctType": "<Debtor Account Type>"
      },
      "aadhaarVerifyDtl": {
         "successCode": "<Success Code>"
      }
},
      "signature": "<Encrypted and Signed response JSON>",
      "checkSumVal": "<Check sum value of complete payload>"


}
```

The below table provides the code for the success

Success Reason	Success Code
Aadhaar number matches with Aadhaar linked with debtor account number Validation Passed	000

Note:-

i. Attribute values mandateValidation, aadhaarValidation, AccptRefNo , successCode &b checkSumVal needs to be encrypted.

ii. Bank needs to store the mandate details received along with the transaction ID for the subsequent OTP validation.

For scenarios (a), (b) and (c) ONMAGS will construct the merchant rejection response and redirect to the merchant. Bank needs to mark the mandate as rejected at their end for these scenarios. For scenario (d) if bank has opted for OTP validation then mandate status will be “In Process” for the bank until the OTP verification is completed, else mandate status will be “Accept” and send the response back to ONMAGS.

For scenario (d) ONMAGS will redirect to the OTP verification page.

Below are the steps to be done for securing the content of the Response JSON:

1. Generating checksum for the secure information in the Response JSON (Mandate and Aadhaar validation)

The below attributes need to be concatenated for the purpose of generating Checksum:

A. Transaction ID
B. Mandate Validation
C. Accepted Ref No.
D. Dbtr Account type
E. Dbtr IFSC
F. Reason Code
G. Reason Desc
H. Error Code
I. Error Desc
J. Aadhaar Validation
K. Success Code
L. Aadhaar Reason Code
M. Aadhaar Error Code
N. JSON Web Signature

2. Generating checksum for the secure information in the Response JSON (OTP Validation)

The below attributes need to be concatenated for the purpose of generating Checksum:

A. Transaction ID
B. Verify Status
C. Error Code
D. Reason Code
E. JSON Web Signature

The above attributes need to be concatenated with "|" symbol appended as the delimiter. The order of the attributes needs to be as mentioned above.

Note:

The attributes to be concatenated might be changed at a later point of time. Please refer the latest version of the document for any revision on the attributes that needs to be marked for

Generate checksum on the concatenated values. We will use SHA-2 as the hash function.

3. Signing of the Response JSON

- The complete response we are going to use as a payload.

- The response JSON has to be signed using the Private Key certificate of the Bank.
- Json Web Signature is used for generating digital signatures and the same will be validated at the NPCI end.

Note :

- Except transaction ID, Dbtr Account Type and Dbtr IFSC field, all the fields are encrypted. For generating checksum, we are going to use encrypted values. If value is not present in response, then we will use empty string for that key.
- Since we are using Signature value while generating Checksum, so that first we need to sign the response then generate checksum

Bank OTP Verification Request for Same Mandate request:

Parameters	DataTypes	Description
otpInfo	JSON Object	This will contains the transaction Id same used in OTP generation and Encrypted OTP which is received on registered mobile in bank
transactionID	String	Same transaction ID used in mandate request to bank. ALPNUM String with Length is 20.
otp	String	Encrypted OTP received on registered mobile in the bank. Length is 4.

```
{
   "otpInfo": [
      {
         "transactionID": "<Transaction ID>",
         "otp": "<Encrypted OTP Value>"
      }
   ]
}
```

- In case of retry as well the request will be posted to bank in the above mentioned format only.

The encryption on the OTP will follow the existing encryption methodology. Bank needs to decrypt the OTP and verify it based on the transaction ID. The OTP verification status needs to be sent in the below json format by the bank.

Response From Bank for Bank OTP Verification for the same Mandate request:

Parameters	DataTypes	Description
otpVerifyInfo	JSON Object	This will contain the same transaction Id which is sent in mandate request to bank and encrypted status as success or failure.
transactionID	String	Transaction Id is the same Which is sent in verify request bank OTP. ALPNUM String with Length is 20.
optVerifyStatus	String	Encrypted OTP verification status. It will be either success / failure

a) If OTP verification at bank end is success then the response will be as below:

```
{
   "otpVerifyInfo":
      {
         "transactionID": "<Transaction ID>",
         "optVerifyStatus": "success",
        "errorCode" : "",
         "reasonCode": ""
       },
       "signature": "<Encrypted and Signed response JSON>",
       "checkSumVal": "<Check sum value of complete payload>"

}
```

b) If OTP verification failed at bank end then response will be as below:

```
{
        "otpVerifyStatus": {
                "transactionID" : "<Transaction ID>",
                "optVerifyStatus":"failure",
                "errorCode":"",
                "reasonCode" : <Reason Code>
        },
        "signature":"<Encrypted and Signed response JSON>",
```

```
        "checkSumVal" : "<Check sum value of complete payload"
}
```

Failure Reason	Reason Code
Bank OTP invalid	AP39
Maximum tries exceeded for OTP	AP40
Time expired for OTP	AP41
Bank Aadhaar OTP Verification response failed	AP50

If OTP verification is successful only Bank needs to mark the mandate as accepted at their end. Until OTP validation is passed the mandate would be in non-accepted state at the Bank end.

If OTP validation is failure User would be provided with option of reattempting OTP validation further 2 times. An alert message as below will be shown to the user. User can then proceed with entering the correct OTP again and re-verify.

Request for Resend Bank OTP:

Parameters	Datatypes	Description
mandateAuthDtls	JSON Object	This will contains transaction Id same which is sent in the first generate bank OTP request and encrypted aadhaar number
transactionID	String	transaction Id same which is sent in the mandate request to bank. ALPNUM String with Length is 20.
aadhaarInfo	JSON Object	This will contains Encrypted aadhaar number
aadhaarNo	String	Encrypted aadhaar number only last four digit.

JSON Request:

```
{
   "aadhaarAuthDtls": {
      "transactionID": "<Transaction ID>",
      "aadhaarInfo": {
         "aadhaarNo": "< Encrypted Aadhaar Number>"
      }
   }
}
```

- Response for Resend Request will be '200' status code.

- If OTP verification is successful only Bank needs to register the mandate as accepted at their end.
- In case OTP verification fails in all the attempts bank can mark the mandate as rejected at their end.

Bank will not generate any OTP, skip the OTP verification step and needs to mark the mandate as accepted at their end.

Technical Integration requirement for Aadhaar Authentication

1.Connectivity:

Communication between NPCI to Bank Server with specific port

2. Certificates

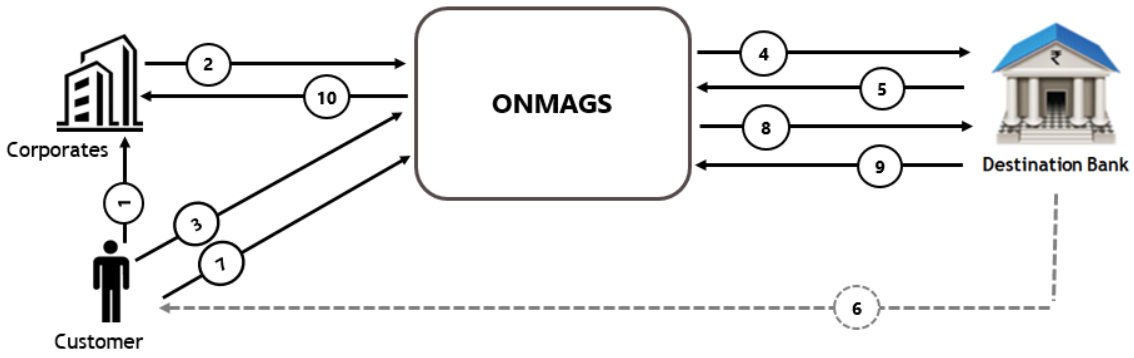
- ☒ Bank SSL certificate(FQDNS)
- ☒ Bank Signing certificate

3.Keys exchange for UIDAI Authentication

 Bank should share their AUA Keys

Bank has to share the keys as part of onboarding process, else we will use NPCI AUA Key

4.2.4 PAN/CUST ID authentication mode



Step1: Customer has initiated the request via Merchant Portal i.e., Web Browser

Step2: Customer will be redirected to ONMAGS Platform to enter the details required for PAN / Cust ID authentication.

Step3: Customer enters PAN Number / Cust ID along with required details.

Step4: If the amount value of the mandate is greater than the defined value, then the flow ends here by showing the error message in Merchant Portal

If the amount value of the mandate is less than or equal to the defined value, then the ONMAGS platform will send the mandate request to destination bank.

Step5: Once mandate details and PAN/Cust ID details verified, bank will provide response bank to ONMAGS. If the mandate details and PAN / Cust ID details verification is failed, bank will provide faliure response to ONMAGS and same will be routed to merchant. The flow ends here.

Step6: Bank will send OTP to the registered mobile number of the customer.

Step7: Once mandate details and PAN / Cust ID details verified at bank successfully, he/she will be landed on ONMAGS OTP page. ONMAGS will send an API request to customer's Bank to verify the customer details will generate the Bank OTP and send it to customer for Authentication.

Step8 : Customer will enter the Bank OTP in ONMAGS platform for Authentication. ONMAGS platform will forward that OTP to destination bank for Verification.

Step9: If OTP verification is successful only Bank needs to mark the mandate as accepted at their end. Until OTP validation is passed the mandate would be in non-accepted state at the Bank end.

Step 10: ONMAGS Platform in turn redirects the response to Merchant Web Page where customer can view the response.

Auth mode: PAN/Cust ID

Privilege: Initiated by ONMAGS (NPCI)

API type: Sync

Request Type: JSON

HTTP Method: POST

Parameter Specification

Parameters	Data Type	Description
mandateAuthDtls	JSON Object	This will contains mandate Request details and aadhaar Info

transactionID	String	This is used for the complete transaction for mandate registration. ALPNUM String with Length is 20.
mndtType	String	This will contain the operation AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL. mndtType will not be present for Create Flow.
mandateRequestDtl	JSON Object	This will contains Encrypted mandate Request Doc XML and Encrypted checksum value
MandateReqDoc	String	See below table for Mandate Request Doc.
CheckSumVal	String	How to generate Checksum value is mentioned above.
authMode	String	If Authmode is PAN, then user will get user will get panInfo JSON. If Authmode is CustID , then user will get user will get custidInfo JSON
panInfo / custidInfo	JSON Object	This will contains the PAN or Cust ID based on the authentication mode selection.
pan/custId	String	PAN/Cust ID of the user

Mandate Request to Bank:

For PAN based authentication

```
{
  "mandateAuthDtls": {
    "transactionID": "<Transaction ID>",
    "mndtType": "< AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL>",
    "mandateRequestDtl": {
      "MandateReqDoc": "<Encrypted and Signed request XML>",
                },
    "authMode": "PAN",
    "panInfo": {
      "pan": "<Encrypted PAN>”
    }
  }
}
```

For Cust ID based authentication

```
{
  "mandateAuthDtls": {
    "transactionID": "<Transaction ID>",
    "mndtType": "<AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL>",
    "mandateRequestDtl": {
      "MandateReqDoc": "<Encrypted and Signed request XML>",
      },
    "authMode": "custid",
    "custidInfo": {
      "custid": "<Encrypted custid>”
    }
  }
}
```

Note :

- mndtType will not be present for Create Flow
- Unencrypted and Unsigned request XML for MandateReqDoc Key is similar to New debit and aadhaar authentication mode:

Bank needs to first verify the mandate request details

a) If the destination bank is unable to parse the mandate request it will send the response in the below format. Bank need not validate the PAN/Cust ID details if sending failure response (because of request XML validation failure at bank end).

Parameters	Datatypes	Description
mandateVerifyDtls	JSON Object	Mandate verify details contains transaction ID, mandate Validation and mandate reject details
transactionID	String	This is the same transaction ID which Is passed in request for

		mandate registration. ALPNUM String with Length is 20.
mndtType	String	This will contain the operation AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL. mndtType will not be present for Create Flow.
mandateValidation	String	This will return either success or failure.
panValidation / custidValidation	String	This will return either success or failure.
mandateRejectDtl	JSON Object	This will contain error code and error desc
ErrorCode	Integer	This will be between 000 to 999
ErrorDesc	String	This will be the corresponding error description for the error code.
signature	String	The Response payload will be signed with bank's private key and algorithm used as RSA_USING_SHA256
checkSumVal	String	Generate checksum on the entire payload. We will use SHA-2 as the hash function

Error Response from Bank for Mandate request:

For PAN authentication mode

```
{
  "mandateVerifyDtls": {
    "transactionID": "<Transaction ID>",
```

```
    "mndtType": "<AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL>",
    "mandateValidation": "failure",
    "panValidation": "none",
     "mandateRejectDtl": {
      "ErrorCode": "<Error Code>",
      "ErrorDesc": "<Error Description>"
    }
  },
  "signature": "<Encrypted and Signed response JSON>",
  "checkSumVal": "<Check sum value of complete payload>"
}
```

For Cust ID based authentication

```
{
  "mandateVerifyDtls": {
    "transactionID": "<Transaction ID>",
    "mndtType": "<AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL>",
    "mandateValidation": "failure",
    " custidValidation ": "none",
     "mandateRejectDtl": {
      "ErrorCode": "<Error Code>",
      "ErrorDesc": "<Error Description>"
    }
  },
  "signature": "<Encrypted and Signed response JSON>",
  "checkSumVal": "<Check sum value of complete payload>"
}
```

Note:-

Attribute values Mandate Validation, PAN/CustID Validation, Error Code & ErrorDesc needs to be encrypted. Bank needs to encrypt using NPCI public key.

b) If destination bank is able to successfully parse the mandate request XML but business validation of XML fails, then bank needs to send the **response** in the below format. PAN/Cust ID details need not be validated in such a scenario.

For PAN authentication mode

```
{
  "mandateVerifyDtls": {
    "transactionID": "<Transaction ID>",
    "mndtType": "<AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL>",
    "mandateValidation": "failure",
    "panValidation": "none",
    "mandateRejectDtl": {
      "ReasonCode": "<Reason Code>",
      "ReasonDesc": "<Reason Description>"
    }
  },
  "signature": "<Encrypted and Signed response JSON>",
  "checkSumVal": "<Check sum value of complete payload>"
}
```

For CustID authentication mode

```
{
  "mandateVerifyDtls": {
    "transactionID": "<Transaction ID>",
    "mndtType": "<AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL>",
    "mandateValidation": "failure",
    "custidValidation": "none",
    "mandateRejectDtl": {
      "ReasonCode": "<Reason Code>",
      "ReasonDesc": "<Reason Description>"
    }
  },
  "signature": "<Encrypted and Signed response JSON>",
  "checkSumVal": "<Check sum value of complete payload>"
}
```

Note:-

Attribute values Mandate Validation, PAN Validation/Custid validation, Reason Code,Reason Desc & checkSumVal needs to be encrypted

c) Validation at banks

- For PAN authentication mode PAN of debtor should match with the "PAN linked with the Debtor AccNo" provided in the mandate Request XML

  If the above validation fails, then the bank needs to provide the response as above format 2nd type.

```
{
  "mandateVerifyDtls": {
    "transactionID": "<Transaction ID>",
     "mndtType": "<AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL>",
    "mandateValidation": "success",
    "panValidation": "failure",
    "mandateResponseDtl": {
      "accptRefNo": "<Accept Reference Number>",
      "dbtrIfsc": "<Debtor IFSC>",
      "dbtrAcctType": "<Debtor Account Type>"
    },
    " panRejectDtl ": {
      "ReasonCode": "<Reason Code>"
    }
  },
  "signature": "<Encrypted and Signed response JSON>",
  "checkSumVal": "<Check sum value of complete payload>"
  }
```

- For Cust ID authentication mode[DG1]

If the above validation fails, then the bank needs to provide the response as above format 2nd type.

```
{
  "mandateVerifyDtls": {
```

```
    "transactionID": "<Transaction ID>",
    "mndtType": "<AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL>",
    "mandateValidation": "success",
    "custidValidation": "failure",
    "mandateResponseDtl": {
      "accptRefNo": "<Accept Reference Number>",
      "dbtrIfsc": "<Debtor IFSC>",
      "dbtrAcctType": "<Debtor Account Type>"
    },
    " custidRejectDtl ": {
      "ReasonCode": "<Reason Code>"
    }
  },
  "signature": "<Encrypted and Signed response JSON>",
  "checkSumVal": "<Check sum value of complete payload>"
  }
```

If all the above validation passes then the bank needs to provide the success **response** as below:-

d) Success Response for Mandate request to Bank:

For PAN authentication

```
{
  "mandateVerifyDtls": {
    "transactionID": "<Transaction ID>",
    "mndtType": "<AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL>",
    "mandateValidation": "success",
    "panValidation": "success",
    "mandateResponseDtl": {
      "accptRefNo": "<Accept Reference Number>",
      "dbtrIfsc": "<Debtor IFSC>",
      "dbtrAcctType": "<Debtor Account Type>"
    },
    "panVerifyDtl": {
      "successCode": "<Success Code>"
    }
  },
  "signature": "<Encrypted and Signed response JSON>",
  "checkSumVal": "<Check sum value of complete payload>"
}
```

For Cust ID authentication

```
{
  "mandateVerifyDtls": {
    "transactionID": "<Transaction ID>",
    "mndtType": "<AMEND / CANCEL / SUSPEND/ REVOKE /CUSTOM_CANCEL>",
    "mandateValidation": "success",
    "custidValidation": "success",
    "mandateResponseDtl": {
      "accptRefNo": "<Accept Reference Number>",
      "dbtrIfsc": "<Debtor IFSC>",
      "dbtrAcctType": "<Debtor Account Type>"
    },
    "custidVerifyDtl": {
      "successCode": "<Success Code>"
    }
  },
  "signature": "<Encrypted and Signed response JSON>",
```

```
  "checkSumVal": "<Check sum value of complete payload>"
}
```

Note:-

iii. Attribute values mandateValidation, panValidation/custidValidation, AccptRefNo , successCode &b checkSumVal needs to be encrypted.

iv. Bank needs to store the mandate details received along with the transaction ID for the subsequent OTP validation.

For scenarios (a), (b) and (c) ONMAGS will construct the merchant rejection response and redirect to the merchant. Bank needs to mark the mandate as rejected at their end for these scenarios. For scenario (d) if bank has opted for OTP validation then mandate status will be “In Process” for the bank until the OTP verification is completed, else mandate status will be “Accept” and send the response back to ONMAGS.

For scenario (d) ONMAGS will redirect to the OTP verification page.

Below are the steps to be done for securing the content of the Response JSON:

4. Generating checksum for the secure information in the Response JSON (Mandate and Pan validation/Cust Id validation)

The below attributes need to be concatenated for the purpose of generating Checksum:

O. Transaction ID
P. Mandate Validation
Q. Accepted Ref No.
R. Dbtr Account type
S. Dbtr IFSC
T. Reason Code
U. Reason Desc
V. Error Code
W. Error Desc
X. Pan Validation / Cust Id validation
Y. Success Code
Z. Aadhaar Reason Code
AA. Aadhaar Error Code
BB. JSON Web Signature

5. Generating checksum for the secure information in the Response JSON (OTP Validation)

The below attributes need to be concatenated for the purpose of generating Checksum:

F. Transaction ID
G. Verify Status
H. Error Code
I. Reason Code
J. JSON Web Signature

The above attributes need to be concatenated with "|" symbol appended as the delimiter. The order of the attributes needs to be as mentioned above.

Note:

The attributes to be concatenated might be changed at a later point of time. Please refer the latest version of the document for any revision on the attributes that needs to be marked for

Generate checksum on the concatenated values. We will use SHA-2 as the hash function.

6. Signing of the Response JSON

- The complete response we are going to use as a payload.
- The response JSON has to be signed using the Private Key certificate of the Bank.
- Json Web Signature is used for generating digital signatures and the same will be validated at the NPCI end.

Note :

- Except transaction ID, Dbtr Account Type and Dbtr IFSC field, all the fields are encrypted. For generating checksum, we are going to use encrypted values. If value is not present in response, then we will use empty string for that key.
- Since we are using Signature value while generating Checksum, so that first we need to sign the response then generate checksum

Bank OTP Verification Request for Same Mandate request:

Parameters	DataTypes	Description
otpInfo	JSON Object	This will contains the transaction Id same used in OTP generation and Encrypted OTP which is received on registered mobile in bank

transactionID	String	Same transaction ID used in mandate request to bank. ALPNUM String with Length is 20.
otp	String	Encrypted OTP received on registered mobile in the bank. Length is 4.

```
{
   "otpInfo": [
      {
         "transactionID": "<Transaction ID>",
         "otp": "<Encrypted OTP Value>"
      }
   ]
}
```

- In case of retry as well the request will be posted to bank in the above mentioned format only.

The encryption on the OTP will follow the existing encryption methodology. Bank needs to decrypt the OTP and verify it based on the transaction ID. The OTP verification status needs to be sent in the below json format by the bank.

Response From Bank for Bank OTP Verification for the same Mandate request:

Parameters	DataTypes	Description
otpVerifyInfo	JSON Object	This will contain the same transaction Id which is sent in mandate request to bank and encrypted status as success or failure.
transactionID	String	Transaction Id is the same Which is sent in verify request bank OTP. ALPNUM String with Length is 20.
optVerifyStatus	String	Encrypted OTP verification status. It will be either success / failure

e) If OTP verification at bank end is success then the response will be as below:

```
{
   "otpVerifyInfo":
      {
         "transactionID": "<Transaction ID>",
         "optVerifyStatus": "success",
        "errorCode" : "",
         "reasonCode": ""
```

```
        },
        "signature": "<Encrypted and Signed response JSON>",
        "checkSumVal": "<Check sum value of complete payload>"

}
```

f) If OTP verification failed at bank end then response will be as below:

```
{
    "otpVerifyStatus": {
            "transactionID" : "<Transaction ID>",
            "optVerifyStatus":"failure",
            "errorCode":"",
            "reasonCode" : <Reason Code>
    },
    "signature":"<Encrypted and Signed response JSON>",
    "checkSumVal" : "<Check sum value of complete payload"
}
```

Failure Reason	Reason Code
Bank OTP invalid	AP39
Maximum tries exceeded for OTP	AP40
Time expired for OTP	AP41

If OTP verification is successful only Bank needs to mark the mandate as accepted at their end. Until OTP validation is passed the mandate would be in non-accepted state at the Bank end.

If OTP validation is failure User would be provided with option of reattempting OTP validation further 2 times. An alert message as below will be shown to the user. User can then proceed with entering the correct OTP again and re-verify.

Request for Resend Bank OTP for PAN authentication mode:

Parameters	Datatypes	Description
mandateAuthDtls	JSON Object	This will contains transaction Id same which is sent in the first generate bank OTP request and encrypted aadhaar number
transactionID	String	transaction Id same which is sent in the mandate request to bank. ALPNUM String with Length is 20.
panInfo	JSON Object	This will contains Encrypted PAN
pan	String	Encrypted PAN

JSON Request:

```
{
   "mandateAuthDtls": {
      "transactionID": "<Transaction ID>",
      "panInfo": {
         "pan": "< Encrypted PAN Number>"
      }
   }
}
```

Request for Resend Bank OTP for CustID authentication mode:

Parameters	Datatypes	Description
mandateAuthDtls	JSON Object	This will contains transaction Id same which is sent in the first generate bank OTP request and encrypted aadhaar number
transactionID	String	transaction Id same which is sent in the mandate request to bank. ALPNUM String with Length is 20.
custidInfo	JSON Object	This will contains Encrypted custid
custid	String	Encrypted custid

JSON Request:

```
{
   "mandateAuthDtls": {
      "transactionID": "<Transaction ID>",
      "custidInfo": {
         "custid": "< Encrypted PAN Number>"
      }
   }
}
```

- Response for Resend Request will be '200' status code.

- If OTP verification is successful only Bank needs to register the mandate as accepted at their end.
- In case OTP verification fails in all the attempts bank can mark the mandate as rejected at their end.

4.3 Signing and Encryption process

Below is the process for encryption & signing during the various flows.

- **NPCI to Bank**
  - Encryption will be done using the Public Key of the certificate shared by Bank.
  - Signing Using Private key certificate of NPCI
- **Bank to NPCI**
  - Encryption will be done using the Public Key of the certificate shared by NPCI.
  - Signing Using Private key certificate of Bank

4.4 Encoding Guidelines

The request XML & response XML posted to NPCI and received from NPCI should in encoded format. As part of encoding specific characters would be replaced by escaped character of those.

Symbol	Spelled	Escaped Character
‘	Single Quotes	&apos;
“	Double Quotes	&quot;
&	Ampersand	&amp;
<	Less Than	&lt;
>	Greater Than	&gt;

5. Response through Offline Server to Server Communication

To account for online failures, the response from Bank to NPCI needs to be sent using server to server communication as well.

NPCI will expose an API for accepting server to server communication from Bank. Bank needs to invoke this URL for posting response through server to server communication.

Note:

- Since communication is received both by browser redirection & server to server call, NPCI would mark the status of the transaction based on the first response received. The second communication received would be ignored.
- Error Code & Error Description list will be shared by NPCI.

5.1 Handling of Time out / not reachable Scenarios

During the entire flow time out can happen at various stages. The following timeouts needs to be maintained at individual levels across the participating entities.

Flow	Auth Mode/Request	Timeout	Remarks
NPCI to Bank	**Old Net Banking/Debit Card**	30Min	No response from Bank for original request & for 3 subsequent sync requests. Request will be marked as timed out at NPCI. (merchant can use Status API to know the status of the request)
	New Debit Card – Submit Card Details	90 Sec	No response from Bank for original request & for 3 subsequent sync requests. Request will be marked as timed out at NPCI and user will be redirected to merchant site with appropriate error code
	New Debit Card – OTP Verification	90 Sec	No response from Bank for original request & for 3 subsequent sync requests. Request will be marked as timed out at NPCI and user will be redirected to merchant site with appropriate error code
	Aadhaar Authentication – Aadhaar verification by UIDAI	90 Sec	No response from UIDAI for original request. Request will be marked as timed out at NPCI and user will be redirected to merchant site with appropriate error code
	Aadhaar Authentication – Aadhaar OTP Authentication	90 Sec	No response from Bank for original request & for 3 subsequent sync requests. Request will be marked as timed out at NPCI and user will be redirected to merchant site with appropriate error code
	Aadhaar Authentication – Account verification by Bank	90 Sec	No response from Bank for original request & for 3 subsequent sync requests. Request will be marked as timed out at NPCI and user will be redirected to merchant site with appropriate error code

	Aadhaar Authentication – Bank OTP Authentication	90 Sec	No response from Bank for original request & for 3 subsequent sync requests. Request will be marked as timed out at NPCI and user will be redirected to merchant site with appropriate error code

Explained below are the actions taken at NPCI ONMAGS layer for timeouts happening at various levels.

- **NPCI to Bank**

  Scenario-1: Destination Bank not reachable

  Action: The request will be auto closed as Failed at NPCI end after the specified duration. Merchant will be shown respective error code.

- **Bank to NPCI**

  Scenario-1: Bank has not responded to NPCI within the timeout period.

  Action: NPCI will send list of transactions for which communication is not received from Bank at periodic interval. Once the pre-defined cut off time for the transaction is reached the transaction would be marked as "No Response from Bank" auto closed.

  Scenario-2: Bank sends response to NPCI after the timeout period

  Action: Any response after the time out period would be ignored by NPCI ONMAGS. The transaction would be treated as no response from Bank and the action for Scenario-1 would be followed.

  Scenario-3: Bank sends invalid response to NPCI within the timeout period

  Action: NPCI ONMAGS will mark the transaction as "Invalid Response from Bank" and corresponding Response XML with applicable error code will be send to Merchant.

  Scenario-4: Bank is unable to reach NPCI.

  Action: Bank needs to communicate the response to NPCI using server to server call. NPCI will update the transaction status at our end. (applicable only for net banking & Old debit card Flow)

5.1.1 JSON Response Formats

Given below are the JSON Response formats for Server to Server Communication.

Note:

Error Response XML would be shared in case the original request is not readable.

5.1.1.1 Bank to NPCI (Success & Business Rejections)

```
{
   "bankResponseDtl":[
      {
         "BANKID":"<Participant ID of the Bank in NACH>",
         "MandateRespDoc":"<Encrypted and Signed response XML>",
         "CheckSumVal":"<Check sum value of secure attributes>",
         "RespType":"RespXML"
      }
   ]
}
```

5.1.1.2 Bank to NPCI Error Response (Technical Rejections)

```
{
   "bankResponseDtl":[
      {
         "BANKID":"<Participant ID of the Bank in NACH>",
         "MandateRespDoc":"<ErrorResponse XML>",
         "RespType":"ErrorXML"
      }
   ]
}
```

6. API services

6.1 API to get Transaction Status for Banks

For the purpose of getting the transaction status of a particular transaction or group of transactions for Banks, NPCI ONMAGS would expose a rest service which will accept list of NPCI Transaction Reference Numbers in JSON format. The response of this API will also be in JSON Format. There will be a limitation on the number of items posted per request. Currently the limit is set as 50.

Sample Input JSON:

```
{

  "npcirefmsgID":[
```

```
    "000f0f29dc27f00000101b09c5227457f17",
    "000f0f29dc27f00000101b09c5227457E23",
    "000f0f29dc27f00000101b09c5227453S42"
  ]
}
```


Sample Output JSON:


```
{
  " tranStatus ":[
    {
      "npcirefmsgID":"000f0f29dc27f00000101b09c5227457f17",
      "Accptd":"false",
      "AccptRefNo":"tranid3432kkkeke",
      "MndtId":"xxxxxxxxxxxxxxxxxxx",
      "ReasonCode":"343",
      "ReasonDesc":"Invalid Account",
      "RejectBy":"Bank",
      "ErrorCode":"000",
      "ErrorDesc":"NA"
    },
    {
      "npcirefmsgID":"000f0f29dc27f00000101b09c5227457E23",
      "Accptd":"true",
      "AccptRefNo":"tranid352254221",
      "MndtId":"xxxxxxxxxxxxxxxxxxx",
```

```
      "ReasonCode":"000",
      "ReasonDesc":"NA",
      "RejectBy":"NA",
      "ErrorCode":"000",
      "ErrorDesc":"NA"
    },
    {
      "npcirefmsgID":"000f0f29dc27f00000101b09c5227453S42",
      "Accptd":"NULL",
      "AccptRefNo":"NULL",
      "MndtId":"NULL",
      "ReasonCode":"NULL",
      "ReasonDesc":"NULL",
      "RejectBy":"NULL",
      "ErrorCode":"452",
      "ErrorDesc":"No Details available for the requested parameters. Please check the values provided"
    }
  ]
}
```


In case the details provided in the request are invalid then ErrorCde & ErrorDesc will have the corresponding error code & description. For the valid request ErrorCode would be “000” and “ErrorDesc” would be “NA”.**API URL would be of the below format:**

https://enach.npci.org.in/apiservices/getTransStatusForBanks

UAT:

https://103.14.161.144/8086/apiservices/getTransStatusForBanks

6.2 API for posting list of Open Transactions to Bank

-
- NPCI will post the open transaction (transaction for which response has not been received from Bank end) to Bank at predefined interval. Bank should expose a listener for accepting the request from NPCI and send response in the same request (**Synchronous**). In the request NPCI provide either MndtId or NpciRefMsgID or Both as a input, bank ready to accept the input and provide details as mentioned in the below format.

☒ **Request:**

```
{
   "openMandateTrans":[

      {
         "MndtId":"xxxxxxxxxxxxxxxxxxx",
         "NpciRefMsgID":"000f0f29dc27f00000101b09c522743SK65"
      }
   ]
}
```

For the open transaction bank needs to provide the response in the same request mentioned in the below

Note:

Bank needs to provide the API URL for accepting this request which should accept the above JSON format.

Given below are the JSON Response formats.

Success Response

```
{
   "bankResponseDtl":[
      {
         "BANKID":"<Participant ID of the Bank in NACH>",
         "MandateRespDoc":"<Encrypted and Signed response XML>",
         "CheckSumVal":"<Check sum value of secure attributes>",
         "RespType":"RespXML"
      }
   ]
}
```

Error Response

Error Response XML would be shared in case the original request is not readable or in case they didn't receive any request for the given npcirefmsgid.

```
{
   "bankResponseDtl":[
      {
         "BANKID":"<Participant ID of the Bank in NACH>",
         "MandateRespDoc":"<ErrorResponse XML>",
         "RespType":"ErrorXML"
      }
   ]
}
```

6.3 HEART BEAT API

ONMAGS system will check the LIVE status of the Banks for particular interval. ONMAGS will send the HTTPS request to banks and in the same request banks give the response **(Synchronous).**

The request and response structure below.

For Banks

Banks can provide "Live" as response to NPCI only if banks can process the API E-Mandate successfully at their end. Assume if banks requires more than one service to successfully register a mandate, banks should check the availability of all services and provide "Live". Even if one service is not working, the response should be provided as "Not live".

6.3.1 Request:

The request will be in the Json format to their respective shared URL's

{

"action":"HEART BEAT REQUEST",

"data":

{

"server_status":"ALIVE",

"current_time":"2019-11-04T09:09:09"

}

}

6..3.2 Response:

{

"action": " HEART BEAT RESPONSE",

"data": {

"status": "ALIVE",

"current_time":"2019-11-04T09:09:09"

}

}

7. Appendix

7.1 Request & Response XML Specification for Banks

Validation_Sheet_Bank

Validation sheet.xlsx

7.2 Sample XML Formats and Schemas

Request Response Files_V9.zip

7.3 Error Codes

ErrorCode - Bank

Error codes.xlsx

7.4 Bank Reject Reason codes

Bank_Reject_reason
_codes.xls

7.5 Guidelines and design for Netbanking page, Debit Card and corporate page

Bank page - API
E-Mandate - Interne

Bank page - API
E-Mandate - Debit c

Implementation
guidelines for Corp

7.6 Logic for generating JSON Web Signature (JWS)

JWS_logc_Aadhaar.d
ocx

7.7 Checksum login for bank response to NPCI

7.7.1 CheckSum Logic for Mandate Validation

Generating Checksum with concatenating below fields

```
txnId + "|" + mndtValidation + "|" + acceptRefNo + "|" + dbtrAccType + "|" +
dbtrIfsc + "|" + reasonCode + "|" + reasonDesc + "|" + errorCode + "|" +
errorDesc + "|" + aadhaarValidation + "|" + successCode + "|" +
aadhaarReasonCode + "|" + aadhaarErrorCode + "|" + signature value
```

Before check sum Hashing example

```
ONMG7032712190068010|U/jQgxdNd4WsN
AvUYZGmvtxO7u3pweD7Wstz7GGdCMmiokzavwiJSStlmagwXE6QFwAyktVFomgAM0QHCFUu/76tiZ
5BXb0uaBxID2PXbYXzhxf28a1hQPJrztfBYoh7cMKoamLMvZOaroFoPImM1IdhIfBvObzLObSfWBy
BmufvqozdXWXU7
```

```
Bm0Zlntujz ZZ6XrdWTVFrG15XaJDDw84CfkEjgklI1kJuC63hhWAnxRN7kTVgjdcdtmczH7GgoJNs
RF3zirTLoJjZJhO23O4J5O5pcsDEeKIUMxtUPFM4M1m0uAz7zYjRPVclDQwCKSIqdmU433GJMxyhj
G1hcmHQ==|st1X
m8hMcQoHfKuqGtWQKCGbeqyzsH1uKp/ocKOjsQ63p569uiWZdGnILKbv5f5vxdZtXFGDFrnC3r3g4
/oRp03AGJTRPCzMX5zETn4d4jwTl+/ujy3zG3idDJUfKl91LN0mN0mjnT9OabiWa3/k4UZztesWv8
vASfVjyO7G0Z5E
FlIWFjj18Xpv7oWea9J0x20EK9UbyVaMDl4JF0zNxIllo45LDH9/IIDn/UB/mZ/EUm6yHHsXzmKub
m9qyWo4NejXnkp9yy9cJ3dVlsApod410LUmbZiwyKgNfU+W9V4PwBeWvDzYu6ZCib3gbxniZsDekw
K+vn6FnhNzrXt1
VA==|DEBIT|508548|||||MNnjhDntjQahfKVLDnaech9LEGf+v2+tIGmyOs/jHce9AgtxFlgyTQE
0tDl31LeAOcmYghJbfc+e8v1ryhauw0B8l5uvva4u3GUEvDxoIbId4XnjZ30c/lUdMcdrI453mtn3
J0sEBpRhkMMWyH
V53T4yRAWcoj6+4oGKtZWw2bpS1YjZrgLjoXAvdgDpvN+rsSdIrUqWg8eFeuDkWTJZo9vOnYHt6pR
qKwSgZUqzcjrYej5fm/5NkI14GJwYB0hjQW6OUn0rchopWL9M1ImxgNAH1G64hyg9UtI8KG6eWAhE
Lm3yEKuZLqGbSm
20QK4wBMHBKlzSbPXcycCDJUOLiA==|GhgMgyEVKMHyElVD/OFScgkyhBp1AOQqLIPJN8jgHCkksb
FOJDO4TFDqHebn77H0ejPMB38SBCJqAO5JAajb3bJSiux0C5dpRrUgpPEJX5OETJlnDCHjiMaIJjb
JmzCeEMUlVTV27
YCIjGKLVfZ+2JTrKHhQWQx03rHLr5OA2aV+AYIGqDWdwQHXJuo0FeJvZZscexMutacNgANLDgJLpS
rTyqFjTrOq5ruYRhLa8ZrSfb3MW+DeyHiveHbj2lQwMsC1/1v7GqrrzZ5PyFPU0HFxoN2RAWVuEpK
UECFmWe+lbLp3u
L/DmlStUQoLEw/rF/KTX2D/A0tKJ3NKU86r3Q==|||eyJhbGciOiJSUzUxMiJ9.eyJtYW5kYXRlVm
VyaWZ5RHRscyI6eyJ0cmFuc2FjdGlvbklEIjoiU0FNUExFMDA5OVVNUk4iLCJtYW5kYXRlVmFsaWR
hdGlvbiI6IlUva
lFneGROZDRXc05BdlVZWkdtdnR4Tzd1M3B3ZUQ3V3N0ejdHR2RDTW1pb2t6YXZ3aUpTU3RsbWFnd1
hFNlFGd0F5a3RWRm9tZ0FNMFFIQ0ZVdS83NnRpWjVCWGIwdWFCeElEMlBYYllYemh4ZjI4YTFoUVB
Kcnp0ZkJZb2g3Y
01Lb2FtTE12Wk9hcm9Gb1BJbU0xSWRoSWZCdk9iekxPYlNmV0J5Qm11ZnZxb3pkWFdYVTdCbTBabG
50dWp6Wlo2WHJkV1RWRnJHMTVYYUpERHc4NENma0VqZ2tsSTFrSnVDNjNoaFdBbnhSTjdrVFZnamR
jZHRtY3pIN0dnb
0pOc1JGM3ppclRMb0pqWkpoTzIzTzRKNU81cGNzREVlS0lVTXh0VVBGTTRNMW0wdUF6N3pZalJQVm
NsRFF3Q0tTSXFkbVU0MzNHSk14eWhqRzFoY21IUT09IiwiYWFkaGFhclZhbGlkYXRpb24iOiJNTm5
qaERudGpRYWhmS1ZMRG5hZWNoOUxFR2YrdjIrdElHbXlPcy9qSGNlOUFndHhGbGd5VFFFMHREbDMx
TGVBT2NtWWdoSmJmYytlOHYxcnloYXV3MEI4bDV1dnZhNHUzR1VFdkR4b0liSWQ0WG5qWjMwYy9sV
WRNY2RySTQ1M210bjNKMHNFQnBSaGtNTVd5SFY1M1Q0eVJBV2NvajYrNG9HS3RaV3cyYnBTMVlqWn
JnTGpvWEF2ZGdEcHZOK3JzU2RJclVxV2c4ZUZldURrV1RKWm85dk9uWUh0NnBScUt3U2daVXF6Y2p
yWWVqNWZtLzVOa0kxNEdKd1lCMGhqUVc2T1VuMHJjaG9wV0w5TTFJbXhnTkFIMUc2NGh5ZzlVdEk4
S0c2ZVdBaEVMbTN5RUt1WkxxR2JTbTIwUUs0d0JNSEJLbHpTYlBYY3ljQ0RKVU9MaUE9PSIsIm1hb
mRhdGVSZXNwb25zZUR0bCI6eyJhY2NwdFJlZk5vIjoic3QxWG04aE1jUW9IZkt1cUd0V1FLQ0diZX
F5enNIMXVLcC9vY0tPanNRNjNwNTY5dWlXWmRHbklMS2J2NWY1dnhkWnRYRkdERnJuQzNyM2c0L29
ScDAzQUdKVFJQQ3pNWDV6RVRuNGQ0andUbCsvdWp5M3pHM2lkREpVZktsOTFMTjBtTjBtam5UOU9h
YmlXYTMvazRVWnp0ZXNXdjh2QVNmVmp5TzdHMFo1RUZsSVdGamoxOFhwdjdvV2VhOUoweDIwRUs5V
WJ5VmFNRGw0SkYwek54SWxsbzQ1TERIOS9JSURuL1VCL21aL0VVbTZ5SEhzWHptS3VibTlxeVdvNE
5lalhua3A5eXk5Y0ozZFZsc0Fwb2Q0MTBMVW1iWml3eUtnTmZVK1c5VjRQd0JlV3ZEell1NlpDaWI
zZ2J4bmlac0Rla3dLK3ZuNkZuaE56clh0MVZBPT0iLCJkYnRySWZzYyI6IjUwODU0OCIsImRidHJB
Y2N0VHlwZSI6IkRFQklUIn0sImFhZGhhYXJWZXJpZnlEdGwiOnsic3VjY2Vzc0NvZGUiOiJHaGdNZ
3lFVktNSHlFbFZEL09GU2Nna3loQnAxQU9RcUxJUEpOOGpnSENra3NiRk9KRE80VEZEcUhlYm43N0
gwZWpQTUIzOFNCQ0pxQU81SkFhamIzYkpTaXV4MEM1ZHBSclVncFBFSlg1T0VUSmxuRENIamlNYUl
KamJKbXpDZUVNVWxWVFYyN1lDSWpHS0xWZlorMkpUcktIaFFXUXgwM3JITHI1T0EyYVYrQVlJR3FE
V2R3UUhYSnVvMEZlSnZaWnNjZXhNdXRhY05nQU5MRGdKTHBTclR5cUZqVHJPcTVydVlSaExhOFpyU
2ZiM01XK0RleUhpdmVIYmoybFF3TXNDMS8xdjdHcXJyelo1UHlGUFUwSEZ4b04yUkFXVnVFcEtVRU
NGbVdlK2xiTHAzdUwvRG1sU3RVUW9MRXcvckYvS1RYMkQvQTB0S0ozTktVODZyM1E9PSJ9fX0.hgB
qdsH9qzksjuzJJpf9MBkV--
```

xuPEMniBmtjk5P6qlWGuck5TKWh6LnUVnn8O1oTfDFIwNQXzFaVRJov24DVaWdpxgL90RHYQH9Ww2
5ByydKv5xBj37cN0mjPBDxqF0VGdYAi717n0wJNQC-
8v14LZ2txPzfKhG9jXASToPYcdUS0wL2c4gYjkIxKn_aD1lYfoFMWWnYwgU4U7QAlDfr9AHhhPQcD
XK-CSMXy2GJOUwDmbUtpVzYyi3-
t3xtt4WFfub6HLSt_5cacNbrCrcDCyHDnIJ60G32NKngXV7MhYC-
5m2BQOQQbPLHCoHbqgmMwN4Dpb2bZTuF4J0ISVpBQ

After checksum hashing SHA 256

654940560c978580c88c0f96f805fff653741be7cee04feba3cd92377d1533a0

Encrypting the checksum with NPCI public key

hBbotAWO67zN/RdYY++PFybVFoBBo8o3phhfrWx6AnFv3/CuAVp73o/X0awterQVp42N+JblY0ypr
NNLk1sqGgQRZZ+C7KBeyG8BBgZOA0X5wveBhDQHUs/Kv8zQw1MLjOkmrdqzWlKrgfCw4AC7tsE6+T
jyp5cnZvfMp9aM43t9rOSGvzr7ivuSPzqYB8agJkAR0WkDQ3Sd67BG40vvzOVyxRVli1ky4bkRmMZ
Z+YlcEVGoynR4MlisxDrg/rcC3FPI3LpQZJ/bz+zaA1O4dFduChLn9d82vZiBoJ3tgupSw0tsSkHD
1yisAa4zyki3OTD4P22JAt2SIifgQc1Jjg==

Final Response with signature and checksum:


```
{
  "mandateVerifyDtls": {
    "transactionID": "ONMG7032712190068010",
    "mandateValidation":
"U/jQgxdNd4WsNAvUYZGmvtxO7u3pweD7Wstz7GGdCMmiokzavwiJSStlmagwXE6QFwAyktVFomgAM0QHCFUu/76tiZ5BXb0u
aBxID2PXbYXzhxf28a1hQPJrztfBYoh7cMKoamLMvZOaroFoPImM1IdhIfBvObzLObSfWByBmufvqozdXWXU7Bm0Zlntujz2Z
6XrdWTVFrG15XaJDDw84CfkEjgklI1kJuC63hhWAnxRN7kTVgjdcdtmczH7GgoJNsRF3zirTLoJjZJhO23O4J5O5pcsDEeKIU
MxtUPFM4M1m0uAz7zYjRPVclDQwCKSIqdmU433GJMxyhjG1hcmHQ==",
    "aadhaarValidation":
"MNnjhDntjQahfKVLDnaech9LEGf+v2+tIGmyOs/jHce9AgtxFlgyTQE0tDl31LeAOcmYghJbfc+e8v1ryhauw0B8l5uvva4u
3GUEvDxoIbId4XnjZ30c/lUdMcdrI453mtn3J0sEBpRhkMMWyHV53T4yRAWcoj6+4oGKtZWw2bpS1YjZrgLjoXAvdgDpvN+rs
SdIrUqWg8eFeuDkWTJZo9vOnYHt6pRqKwSgZUqzcjrYej5fm/5NkI14GJwYB0hjQW6OUn0rchopWL9M1ImxgNAH1G64hyg9Ut
I8KG6eWAhELm3yEKuZLqGbSm20QK4wBMHBKlzSbPXcycCDJUOLiA==",
    "mandateResponseDtl": {
      "accptRefNo":
"st1Xm8hMcQoHfKuqGtWQKCGbeqyzsH1uKp/ocKOjsQ63p569uiWZdGnILKbv5f5vxdZtXFGDFrnC3r3g4/oRp03AGJTRPCzM
X5zETn4d4jwTl+/ujy3zG3idDJUfKl91LN0mN0mjnT9OabiWa3/k4UZztesWv8vASfVjyO7G0Z5EFlIWFjj18Xpv7oWea9J0x
20EK9UbyVaMDl4JF0zNxIllo45LDH9/IIDn/UB/mZ/EUm6yHHsXzmKubm9qyWo4NejXnkp9yy9cJ3dVlsApod410LUmbZiwyK
gNfU+W9V4PwBeWvDzYu6ZCib3gbxniZsDekwK+vn6FnhNzrXt1VA==",
      "dbtrIfsc": "508548",
      "dbtrAcctType": "DEBIT"
    },
    "aadhaarVerifyDtl": {
      "successCode":
"GhgMgyEVKMHyElVD/OFScgkyhBp1AOQqLIPJN8jgHCkksbFOJDO4TFDqHebn77H0ejPMB38SBCJqAO5JAajb3bJSiux0C5dp
RrUgpPEJX5OETJlnDCHjiMaIJjbJmzCeEMUlVTV27YCIjGKLVfZ+2JTrKHhQWQx03rHLr5OA2aV+AYIGqDWdwQHXJuo0FeJvZ
ZscexMutacNgANLDgJLpSrTyqFjTrOq5ruYRhLa8ZrSfb3MW+DeyHiveHbj2lQwMsC1/1v7GqrrzZ5PyFPU0HFxoN2RAWVuEp
KUECFmWe+lbLp3uL/DmlStUQoLEw/rF/KTX2D/A0tKJ3NKU86r3Q=="
    }
  },
  "checkSumVal":
"hBbotAWO67zN/RdYY++PFybVFoBBo8o3phhfrWx6AnFv3/CuAVp73o/X0awterQVp42N+JblY0yprNNLk1sqGgQRZZ+C7KBe
yG8BBgZOA0X5wveBhDQHUs/Kv8zQw1MLjOkmrdqzWlKrgfCw4AC7tsE6+Tjyp5cnZvfMp9aM43t9rOSGvzr7ivuSPzqYB8agJ
kAR0WkDQ3Sd67BG40vvzOVyxRVli1ky4bkRmMZZ+YlcEVGoynR4MlisxDrg/rcC3FPI3LpQZJ/bz+zaA1O4dFduChLn9d82vZ
iBoJ3tgupSw0tsSkHD1yisAa4zyki3OTD4P22JAt2SIifgQc1Jjg==",
  "signature":
"eyJhbGciOiJSUzUxMiJ9.eyJtYW5kYXRlVmVyaWZ5RHRscyI6eyJ0cmFuc2FjdGlvbklEIjoiU0FNUExFMDA5OVVNUk4iLCJ
```

tYW5kYXRlVmFsaWRhdGlvbiI6IlUvalFneGROZDRXc05BdlVZWkdtdnR4Tzd1M3B3ZUQ3V3N0ejdHR2RDTW1pb2t6YXZ3aUpT
U3RsbWFnd1hFNlFGd0F5a3RWRm9tZ0FNMFFIQ0ZVdS83NnRpWjVCWGIwdWFCeElEMlBYYllYemh4ZjI4YTFoUVBKcnp0ZkJZb
2g3Y01Lb2FtTE12Wk9hcm9Gb1BJbU0xSWRoSWZCdk9iekxPYlNmV0J5Qm11ZnZxb3pkWFdYVTdCbTBabG50dWp6Wlo2WHJkV1
RWRnJHMTVYYUpERHc4NENma0VqZ2tsSTFrSnVDNjNoaFdBbnhSTjdrVFZnamRjZHRtY3pIN0dnb0pOc1JGM3ppclRMb0pqWkp
oTzIzTzRKNU81cGNzREVlS0lVTXh0VVBGTTRNMW0wdUF6N3pZalJQVmNsRFF3Q0tTSXFkbVU0MzNHSk14eWhqRzFoY21IUT09
IiwiYWFkaGFhclZhbGlkYXRpb24iOiJNTm5qaERudGpRYWhmS1ZMRG5hZWNoOUxFR2YrdjIrdElHbXlPcy9qSGNlOUFndHhGb
Gd5VFFFMHREbDMxTGVBT2NtWWdoSmJmYytlOHYxcnloYXV3MEI4bDV1dnZhNHUzR1VFdkR4b0liSWQ0WG5qWjMwYy9sVWRNY2
RySTQ1M210bjNKMHNFQnBSaGtNTVd5SFY1M1Q0eVJBV2NvajYrNG9HS3RaV3cyYnBTMVlqWnJnTGpvWEF2ZGdEcHZOK3JzU2R
JclVxV2c4ZUZldURrV1RKWm85dk9uWUh0NnBScUt3U2daVXF6Y2pyWWVqNWZtLzVOa0kxNEdKd1lCMGhqUVc2T1VuMHJjaG9w
V0w5TTFJbXhnTkFIMUc2NGh5ZzlVdEk4S0c2ZVdBaEVMbTN5RUt1WkxxR2JTbTIwUUs0d0JNSEJLbHpTYlBYY3ljQ0RKVU9Ma
UE9PSIsIm1hbmRhdGVSZXNwb25zZUR0bCI6eyJhY2NwdFJlZk5vIjoic3QxWG04aE1jUW9IZkt1cUd0V1FLQ0diZXF5enNIMX
VLcC9vY0tPanNRNjNwNTY5dWlXWmRHbklMS2J2NWY1dnhkWnRYRkdERnJuQzNyM2c0L29ScDAzQUdKVFJQQ3pNWDV6RVRuNGQ
0andUbCsvdWp5M3pHM2lkREpVZktsOTFMTjBtTjBtam5UOU9hYmlXYTMvazRVWnp0ZXNXdjh2QVNmVmp5TzdHMFo1RUZsSVdG
amoxOFhwdjdvV2VhOUoweDIwRUs5VWJ5VmFNRGw0SkYwek54SWxsbzQ1TERIOS9JSURuL1VCL21aL0VVbTZ5SEhzWHptS3Vib
TlxeVdvNE5lalhua3A5eXk5Y0ozZFZsc0Fwb2Q0MTBMVW1iWml3eUtnTmZVK1c5VjRQd0JlV3ZEell1NlpDaWIzZ2J4bmlac0
Rla3dLK3ZuNkZuaE56clh0MVZBPT0iLCJkYnRySWZzYyI6IjUwODU0OCIsImRidHJBY2N0VHlwZSI6IkRFQklUIn0sImFhZGh
hYXJWZXJpZnlEdGwiOnsic3VjY2Vzc0NvZGUiOiJHaGdNZ3lFVktNSHlFbFZEL09GU2Nna3loQnAxQU9RcUxJUEpOOGpnSENr
a3NiRk9KRE80VEZEcUhlYm43N0gwZWpQTUIzOFNCQ0pxQU81SkFhamIzYkpTaXV4MEM1ZHBSclg1T0VUSmxuRENIa
mlNYUlKamJKbXpDZUVNVWxWVFYyN1lDSWpHS0xWZlorMkpUcktIaFFXUXgwM3JITHI1T0EyYVYrQVlJR3FEV2R3UUhYSnVvME
ZlSnZaWnNjZXhNdXRhY05nQU5MRGdKTHBTclR5cUZqVHJPcTVydVlSaExhOFpyU2ZiM01XK0RleUhpdmVIYmoybFF3TXNDMS8
xdjdHcXJyelo1UHlGUFUwSEZ4b04yUkFXVnVFcEtVRUNGbVdlK2xiTHAzdUwvRG1sU3RVUW9MRXcvckYvS1RYMkQvQTB0S0oz
TktVODZyM1E9PSJ9fX0.hgBqdsH9qzksjuzJJpf9MBkV--
xuPEMniBmtjk5P6qlWGuck5TKWh6LnUVnn8O1oTfDFIwNQXzFaVRJov24DVaWdpxgL90RHYQH9Ww25ByydKv5xBj37cN0mjPB
DxqF0VGdYAi717n0wJNQC-
8v14LZ2txPzfKhG9jXASToPYcdUS0wL2c4gYjkIxKn_aD1lYfoFMWWnYwgU4U7QAlDfr9AHhhPQcDXK-
CSMXy2GJOUwDmbUtpVzYyi3-t3xtt4WFfub6HLSt_5cacNbrCrcDCyHDnIJ60G32NKngXV7MhYC-
5m2BQOQQbPLHCoHbqgmMwN4Dpb2bZTuF4J0ISVpBQ"
}

7.7.2 CheckSum Logic for OTP Validation

Generating Checksum with concatenating below fields

```
txnId + "|" + verifyStatus + "|" + errorCode + "|" + reasonCode + "|" + signature value
```

Before check sum Hashing example

**SAMPLE0099UMRN|ea7HBMRsu32GWkZv8sLTCD0JvfGXz7bc975yjuQfy2jUmM8cOfNiSCN2x31uyS
tYTdfAC4kpEFDAIW0v0ensqA1TYQE7r8MOJ+UG0M+eexz3/OBaUFQhHnBBbln+YsilrGKIkZVeaf1
PC/5X7yTjOlI5WQe9kyPy2/np4Cgp+lMW7ggABQ3PN8xb18Y6s0eIFz+0rTgBpqZyKhCgHUGpiTcl
DNWb1a6OIEn0q2E1jrAnqwGomFHYV1KIiGWA2pa6uLVVEnVQts5c6M/b5vPQHJBOdnkWCPtNUIl3f
0fMPt/K1UTeuG6WEg3s/tPsTk1bTahnex6hYZ38ltqLpHBpcg==|bgduA1UdDuSJgHLC3E08jwxP2
BvFsCcE0PaiXooPn2y1jwj/ANHTfq0/aA1gNJ/+uhZW9/+nnBNCBna/NdoD5h4ctulSPXPUG/DrQY
y6jEmTYhHr+Lac2Sg/ZIZv3b2JN1Yg8Epnk8DhkuW1s4r9R0toQuktWCGb7646Mt6chKDJHL4V/iw
IHisJlOIQmZ9UF2ao5Hw1ftBbr8dx8HsbGJFaOZrmRmNX83P1hWV68Unf3tDyHpkQKRFoXfKbjhP5
emZxW6DSdm1BfMsVi22b1BCrpyxfHR/jMEa02LTJSsJLtKj37XxXn5x9fm5yJPX54Q+ETH0QasPuo
ypj45Gv6Q==||eyJhbGciOiJSUzUxMiJ9.eyJvdHBWZXJpZnlJbmZvIjp7InRyYW5zYWN0aW9uSUQ
iOiJTQU1QTEUwMDk5VU1STiIsIm9wdFZlcmlmeVN0YXR1cyI6ImVhN0hCTVJzdTMyR1drWnY4c0xU
Q0QwSnZmR1h6N2JjOTc1eWp1UWZ5MmpVbU04Y09mTmlTQ04yeDMxdXlTdFlUZGZBQzRrcEVGREFJV
zB2MGVuc3FBMVRZUUU3cjhNT0orVUcwTStlZXh6My9PQmFVRlFoSG5CQmJsbitZc2lsckdLSWtaVm
VhZjFQQy81WDd5VGpPbEk1V1FlOWt5UHkyL25wNENncCtsTVc3Z2dBQlEzUE44eGIxOFk2czBlSUZ
6KzByVGdCcHFaeUtoQ2dIVUdwaVRjbEROV2IxYTZPSUVuMHEyRTFqckFucXdHb21GSFlWMUtJaUdX
QTJwYTZ1TFZWRW5WUXRzNWM2TS9iNXZQUUhKQk9kbmtXQ1B0TlVJbDNmMGZNUHQvSzFVVGV1RzZXR
Wczcy90UHNUazFiVGFobmV4NmhZWjM4bHRxTHBIQnBjZz09IiwiZXJyb3JDb2RlIjoiYmdkdUExVW
REdVNKZ0hMQzNFMDhqd3hQMkJ2RnNDY0UwUGFpWG9vUG4yeTFqd2ovQU5IVGZxMC9hQTFnTkovK3V**

oWlc5LytubkJOQ0JuYS9OZG9ENWg0Y3R1bFNQWFBVRy9EclFZeTZqRW1UWWhIcitMYWMyU2cvWkla djNiMkpOMVlnOEVwbms4RGhrdVcxczRyOVIwdG9RdWt0V0NHYjc2NDZNdDZjaEtESkhMNFYvaXdJS GlzSmxPSVFtWjlVRjJhbzVIdzFmdEJicjhkeDhIc2JHSkZhT1pybVJtTlg4M1AxaFdWNjhVbmYzdE R5SHBrUUtSRm9YZktiamhQNWVtWnhXNkRTZG0xQmZNc1ZpMjJiMUJDcnB5eGZIUi9qTUVhMDJMVEp Tc0pMdEtqMzdYeFhuNXg5Zm01eUpQWDU0UStFVEgwUWFzUHVveXBqNDVHdjZRPT0ifX0.ltmq6Eeo yBJ9C05cKl73-t7qgQLKkjHYqrOA6EaK82Y3tqqt1TuOw_hkB6K-c78eVWzef6oohveQfMr4cPGwqT0LJAs7tnhmDjH2z1iCdLZI2SgmUffZ5s2POe0KgJvwrIPOkvnEU0eYexdatYhFGskola4uycRQJ19UjLtj3Bxa_Rlh_EmDD4Nbol_XslcTJ_NbKS1wvFeVrqG3gFX6Zp RHn2MxSsVA8lTOadP3Zk-jKOi15n500ZoXVwvCSv2Qp4CPLr6gphRWg8T_JX-7ZBirGb8ZrZ4Ntn8FSdkBvQrVbKGfy7YVvg_jmG34XoO3ozhhToLZLQYo-5eHeXIydw

After checksum hashing SHA 256

```
654940560c978580c88c0f96f805fff653741be7cee04feba3cd92377d1533a0
```

Encrypting the checksum with NPCI public key

```
TP1stsqIpWnRdr+TTE3+b1dQo5pUVUV/mQMPEFRljlcpFPGYEkFpxFAXEpmvslFR5+9segYY4771w
nhDAAneepYmpQ/+yrABfy3fvnLc5LTAVrZcnIFZa6qjIwOIh1eiQm2mJ75rxizn+uWxEj8D9B16Gr
/3q0YjI8bkNQXN5Aeh2ldZfKECxZ6liBXRJEizZsXXd01lKxkyyx/3nFozZjpmoIq44N3Xk7xwJGG
Gvbg/z5jT773MRw57NIU7vXGkmmW6gFtMX/nnZuxCEnsjee0hzIvKuxMNwMslt9XSh5GvcU6FtFci
Du8x4mNL5o+7EIfI3fSNEWXYv7xlPsoxDQ==
```

Final Response with signature and checksum:

```
{
  "otpVerifyInfo": {
    "transactionID": "ONMG70327121900680l0",
    "optVerifyStatus":
"ea7HBMRsu32GWkZv8sLTCD0JvfGXz7bc975yjuQfy2jUmM8cOfNiSCN2x31uyStYTdfAC4kpEFDAIW0v0ensq
A1TYQE7r8MOJ+UG0M+eexz3/OBaUFQhHnBBbln+YsilrGKIkZVeaf1PC/5X7yTjOlI5WQe9kyPy2/np4Cgp+lM
W7ggABQ3PN8xb18Y6s0eIFz+0rTgBpqZyKhCgHUGpiTclDNWb1a6OIEn0q2E1jrAnqwGomFHYV1KIiGWA2pa6u
LVVEnVQts5c6M/b5vPQHJBOdnkWCPtNUIl3f0fMPt/K1UTeuG6WEg3s/tPsTk1bTahnex6hYZ38ltqLpHBpcg=
=",
    "errorCode":
"bgduA1UdDuSJgHLC3E08jwxP2BvFsCcE0PaiXooPn2y1jwj/ANHTfq0/aA1gNJ/+uhZW9/+nnBNCBna/NdoD5
h4ctulSPXPUG/DrQYy6jEmTYhHr+Lac2Sg/ZIZv3b2JN1Yg8Epnk8DhkuW1s4r9R0toQuktWCGb7646Mt6chKD
JHL4V/iwIHisJlOIQmZ9UF2ao5Hw1ftBbr8dx8HsbGJFaOZrmRmNX83P1hWV68Unf3tDyHpkQKRFoXfKbjhP5e
mZxW6DSdm1BfMsVi22b1BCrpyxfHR/jMEa02LTJSsJLtKj37XxXn5x9fm5yJPX54Q+ETH0QasPuoypj45Gv6Q=
="
  },
  "checkSumVal":
"TP1stsqIpWnRdr+TTE3+b1dQo5pUVUV/mQMPEFRljlcpFPGYEkFpxFAXEpmvslFR5+9segYY4771wnhDAAnee
pYmpQ/+yrABfy3fvnLc5LTAVrZcnIFZa6qjIwOIh1eiQm2mJ75rxizn+uWxEj8D9B16Gr/3q0YjI8bkNQXN5Ae
h2ldZfKECxZ6liBXRJEizZsXXd01lKxkyyx/3nFozZjpmoIq44N3Xk7xwJGGGvbg/z5jT773MRw57NIU7vXGkm
mW6gFtMX/nnZuxCEnsjee0hzIvKuxMNwMslt9XSh5GvcU6FtFciDu8x4mNL5o+7EIfI3fSNEWXYv7xlPsoxDQ=
=",
  "signature":
"eyJhbGciOiJSUzUxMiJ9.eyJvdHBWZXJpZnlJbmZvIjp7InRyYW5zYWN0aW9uSUQiOiJTQU1QTEUwMDk5VU1S
TiIsIm9wdFZlcmlmeVN0YXR1cyI6ImVhN0hCTVJzdTMyR1drWnY4c0xUQ0QwSnZmR1h6N2JjOTc1eWp1UWZ5Mm
pVbU04Y09mTmlTQ04yeDMxdXlTdFlUZGZBQzRrcEVGREFJVzB2MGVuc3FBMVRZUUU3cjhNT0orVUcwTStlZXh6
My9PQmFVRlFoSG5CQmJsbitZc2lsckdLSWtaVmVhZjFQQy81WDd5VGpPbEk1V1FlOWt5UHkyL25wNENncCtsTV
c3Z2dBQlEzUE44eGIxOFk2czBlSUZ6KzByVGdCcHFaeUtoQ2dIVUdwaVRjbEROV2IxYTZPSUVuMHEyRTFqckFu
```

cXdHb21GSFlWMUtJaUdXQTJwYTZ1TFZWRW5WUXRzNWM2TS9iNXZQUUhKQk9kbmtXQ1B0TlVJbDNmMGZNUHQvSz
FVVGV1RzZXRWczcy90UHNUazFiVGFobmV4NmhZWjM4bHRxTHBIQnBjZz09IiwiZXJyb3JDb2RlIjoiYmdkdUEx
VWREdVNKZ0hMQzNFMDhqd3hQMkJ2RnNDY0UwUGFpWG9vUG4yeTFqd2ovQU5IVGZxMC9hQTFnTkovK3VoWlc5Ly
tubkJOQ0JuYS9OZG9ENWg0Y3R1bFNQWFBVRy9EclFZeTZqRW1UWWhIcitMYWMyU2cvWkladjNiMkpOMVlnOEVw
bms4RGhrdVcxczRyOVIwdG9RdWt0V0NHYjc2NDZNdDZjaEtESkhMNFYvaXdJSGlzSmxPSVFtWjlVRjJhbzVIdz
FmdEJicjhkeDhIc2JHSkZhT1pybVJtTlg4M1AxaFdWNjhVbmYzdER5SHBrUUtSRm9YZktiamhQNWVtWnhXNkRT
ZG0xQmZNc1ZpMjJiMUJDcnB5eGZIUi9qTUVhMDJMVEpTc0pMdEtqMzdYeFhuNXg5Zm01eUpQWDU0UStFVEgwUW
FzUHVveXBqNDVHdjZRPT0ifX0.ltmq6EeoyBJ9C05cKl73-t7qgQLKkjHYqrOA6EaK82Y3tqqt1TuOw_hkB6K-
c78eVWzef6oohveQfMr4cPGwqT0LJAs7tnhmDjH2z1iCdLZI2SgmUffZ5s2POe0KgJvwrIPOkvnEU0eYexdatY
hFGskola4uycRQJ19UjLtj3Bxa_Rlh_EmDD4Nbol_XslcTJ_NbKS1wvFeVrqG3gFX6ZpRHn2MxSsVA8lTOadP3
Zk-jKOi15n500ZoXVwvCSv2Qp4CPLr6gphRWg8T_JX-
7ZBirGb8ZrZ4Ntn8FSdkBvQrVbKGfy7YVvg_jmG34XoO3ozhhToLZLQYo-5eHeXIydw"
}