RS SOFTWARE

The ePayments Experts

Table of Contents

1. Introduction

Purpose

The objective of this document is to provide a description of basic functional SDK APIs that are to be consumed by mobile Apps. This document describes the inputs and outputs for each SDK APIs.

Scope

This document covers the following description

A. Description of Each API
B. Inputs for the APIs
C. Outputs/ Consumption Process for each API.

2. UPI SDK API Description

This UPI SDK release exposes different UPI functionalities in form of SDK APIs which can be further consumed by Mobile Apps for executing UPI specific transactions. Following are the exposed APIs along with their inputs and outputs:

initConfigurationParametres

Purpose: This API should be called first for getting all the values that will be used by the PSP app further.

Inputs:

Input Argument	Description
InputConfigParam	Class object needed.
APICallback< ConfigElement > here T = com.upi.sdk.core.UpiSDKContext.UPIPay Output: ConfigElement object contains- bankPSP-PSP Bank bankCode-Bank Code publickKey-Public Key for password encryption maxLimit-PSP max limit bankName-Bank Name payValidInSec-Pay request default validity in milliseconds collectValidInSec- Collect request default validity in milliseconds k0Hex-Key for encrypt message.	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the Mobile App. ***function onSuccess<T>:*** *On successful execution of the API, this callback function returns the expected result as an object of* ConfigElement. ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Example Usage:

```
InputConfigParam inputConfigParam = new InputConfigParam();
                UpiSDKContext.initConfigurationParametres(this,inputConfigParam,
new APICallback<ConfigElement>() {
                    @Override
                    public void onSuccess(ConfigElement configElement) {


                    }

                    @Override
                    public void onFailure(String sdkErrorCode,String upiErrorCode) {
                                        }
                });
            } catch (Exception e) {

            }
```

Initialize SDK

Purpose: This API should be called for Initializing the SDK Context with PSP specific data. This is the SDK API which should be invoked by the Mobile App before calling any other APIs. If the API returns the error INVALID_USER, the user is not registered with the PSP backend. In this case the client needs to call *createDefaultVPA* API with the required values.

Note:

If user do not have any account he has to follow the below steps for registration.

1) First call **sentSMS** API which will return a message on sms delivered. Then call **validatemobileNumber** API with that same message which will return the same mobile number on success.

2) With that mobile number as a non editable one call API **createDefaultVPA** where the mobile number is the same one that user get by calling **validatemobileNumber** API.

3) After that call **bindDevice** API on which user will get true response after device binding. Then your user is registered successfully and you can move forward with that user.

Inputs:

Input Argument	Description
applicationContext	Application context for the App
customerRef	Existing unique identifier of the Customer. DataType: String. Length: 20 characters.

	E.g. existing Customer ID or User ID.
passphrase	User password. DataType: String. Length: 6 characters.
APICallback<T> here T = com.upi.sdk.core.UpiSDKContext.UPIPay	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the Mobile App. ***function onSuccess<T>:*** *On successful execution of the API, this callback function returns the expected result as an object of UPIPay.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in*

Example Usage:

```
UpiSDKContext.UPIPay.initializeSDK(Npci_Login.this, <userId,
                <USER_PASSWORD>, new APICallback<UpiSDKContext.UPIPay>() {
                   @Override
                   public void onSuccess(UpiSDKContext.UPIPay upiPay)  {
                        }@Override
                   public void onFailure(String sdkErrorCode,String upiErrorCode) {
}
catch (Exception e) {
     CommonUtils.cancelDialog(mProgressDialog);
     e.printStackTrace();
}
```

Initialize the SDK instance for further use:

The UPI SDK instance needs to be initialized first with application context and then with that instance one have to revoke all it's method.

Example Usage:

```
public static ApplicationClassUpiTest getApplication() {
        return application;
    }

Return the application instance.

private UpiSDKContext.UPIPay upiPay = null;

public UpiSDKContext.UPIPay getUpiPay() {
    return this.upiPay;
}

public void setUpiPay(UpiSDKContext.UPIPay upiPay) {
    this.upiPay = upiPay;
}


Setter and getter function for UpiSDKContext.UPIPay instance.
```

addBankAccount

Purpose: This API should be called by the Mobile App for adding a Bank Account for the User.

Inputs:

Input Argument	Description
Object of Class ***InputAddBankAccount*** having below Members :	Object containing Bank Account addition inputs.
• customerRefId(not mandatory)	Existing unique identifier of the Customer. E.g. existing Customer ID or User ID. DataType: String. Length: 20 characters.

• nickName(*)	Nick Name to be assigned to the added Bank. DataType: String. Length: 30 characters.
• bankAcNumber(*)	Bank Account Number. DataType: String. Length: 11 characters.
• ifsc(*)	IFSC number for the corresponding Bank. DataType: String. Length: 20 characters.
• meba	Set meba tag value as true or false that User will get from listBankAccounts API
• atmCredLength	Cred Length of ATM that User will get from listBankAccountByMobile Ifsc API
• atmCredType	Cred Type of ATM that User will get from listBankAccountByMobile Ifsc API
APICallback<T> here T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T>:*** *On scessful execution of the API, this callback function returns the success message in form of String.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and*

Example Usage:

```
InputAddBankAccount inBankAccount = new InputAddBankAccount();
inBankAccount.setBankAcNumber(bankAcnumber);
inBankAccount.setNickName(nickName);
inBankAccount.setIfsc(ifsc);
inBankAccount .setMeba(<true
or false>);
try {

ApplicationClassUpiTest.getApplication().getUpiPay().addBankAccount(inBankAccount,
new APICallback<String>() {
        @Override
        public void onSuccess(String result) {

        }
        @Override
        public void onFailure(String sdkErrorCode,String upiErrorCode) {

    });
} catch (UPISDKException e) {
    e.printStackTrace();
```

1.1. listBankAccount

Purpose: This API should be called by the Mobile App for getting a list of Bank Accounts added by the User.

Inputs:

Input Argument	Description
Object of Class ***InListLinkAccountQuery*** having below Members :	Object containing query inputs for Bank Account listing.
• customerRefId(not mandatory)	Existing unique identifier of the Customer. E.g. existing Customer ID or User ID. DataType: String. Length: 20 characters.
• resultPerPage(not mandatory)	Number of records to be shown in a page. DataType: integer.
APICallback<T> here T = LinkAccountDetails[]	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T>:*** *On scessful execution of the API, this callback function returns the success message in form of array of LinkAccount object.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A***.

Members of Class : LinkAccountDetails	Description
linkAccId	Linked Account Number
userId	User ID
bankAcnumber	Bank Account Number
ifsc	IFSC Code for the Bank
mmid	MMID for the Account
mobile	Mobile Number associated with the Bank Account
aadhaar	Aadhaar Number associated with the Bank Account
accountType	Account Type e.g (S/B, Current etc.)
branchName	Name of the Branch
nickName	Assigned nick name for the bank account
ownBank	Whether the bank account belongs to the same PSP or not.
acceptedCread	Accepted Cred like PIN or MPIN
creadDataType	Cred data type
credLength	Cred length for that account
accountBalance	Balance in the Bank Account
UserActivity (enum)	*Current User Activity.* *e.g. register/fetch/update/active etc.*
meba	*Mobile banking registration tag*
List<Cred>	

Example Usage:

```
ApplicationClassUpiTest.getApplication().getUpiPay().listBankAccount(inQuery, new
    APICallback<LinkAccountDetails[]>() {
                @Override
                public void onSuccess(LinkAccountDetails[] result) {


                }

                @Override
                public void onFailure(String sdkErrorCode,String upiErrorCode) {

                }
            });
        }

    } catch (UPISDKException e) {
    }
```

fetchBanks

Purpose: This API should be called by the Mobile App for getting list of psp Banks.

Input: None.

Members of Class : *PspBankList*	Description
accPvdId	Bank Id

accPvdName	Bank Name
accPvdIfsc	Bank IFSC
status	Status
mobRegFormat	Regmobformat for atm pin type

Example Usage:

```
try {

    ApplicationClassUpiTest.getApplication().getUpiPay().fetchBanks(new
    APICallback<PspBankList[]>() {

        @Override

        public void onSuccess(PspBankList[] result) {



        }



        @Override

        public void onFailure(String sdkErrorCode,String upiErrorCode) {



        }

    });

} catch (UPISDKException e) {



}
```

listBankAccountByMobileIfsc

Purpose: This API should be called by the Mobile App for getting list of Banks by mobile number and IFSC.

Inputs:

Input Argument	Description
APICallback<T> here T = PspBankList[] Object of Class **InListLinkAccountQueryByMobile** having below Members : ifsc(*) mobile_number(*)	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form Array of PspBankList object.*

	function onError (String sdkErrorCode,String upiErrorCode): *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.* *IFSC: Bank ifsc for which accounts will be fetched.* DataType: String. Length: 20 characters.

Example Usage:

```
ApplicationClassUpiTest.getApplication().getUpiPay().listBankAccountByMobileIfs
c(InListLinkAccountQueryByMobile inpQuery, new APICallback<LinkAccountDetails[]>() {
            @Override
            public void onSuccess(LinkAccountDetails[] linkAccountDetailses) {
                        }

            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {
                        });

    }
} catch (UPISDKException e) {

}
```

deleteBankAccount

Purpose: This API should be called by the Mobile App for marking an existing Bank Account as deleted.

Inputs:

Input Argument	Description
Object of Class ***InputDeleteBankAccount*** having below Members :	Object containing query inputs for Bank Account listing.
• accountId(*)	Existing unique id of that account. DataType: Integer.
• nickName(*)	nickName for the Linked Account to be deleted. DataType: String. Length: 20 characters.

APICallback<T> here T = String	A generic Callback Interface (having callback functions like *onSucces* and *onFailure* that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of String.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A***.

Example Usage:

```
InputDeleteBankAccount inputDeleteBankAccount=new InputDeleteBankAccount();
inputDeleteBankAccount.setNickName("");
inputDeleteBankAccount.setLinkAccId("");
try {

ApplicationClassUpiTest.getApplication().getUpiPay().deleteBankAccount(inputDeleteB
ankAccount, new APICallback<String>() {
        @Override
        public void onSuccess(String s) {

        }
        @Override
        public void onFailure(String sdkErrorCode,String upiErrorCode) {

}catch (UPISDKException e){
    e.printStackTrace();

}
```

checkVPAAvailability

Purpose: This API should be called by the Mobile App for checking the availability/uniqueness of the Virtual Payment Address before adding the same.

Inputs:

Input Argument	Description
Object of Class ***InputcheckVPAAvailability*** having below Members :	Object containing query inputs for Bank Account listing.
• customerRefId(not mandatory)	Existing unique identifier of the Customer. E.g. existing Customer ID or User ID. DataType: String. Length: 20 characters.
• userVPA(*)	User VPA to be checked for uniqueness. DataType: String. Length: 50 characters.
APICallback<T> here T = Boolean	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of Boolean value.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Anexture-A***.

Example Usage:

```
try {
     UpiSDKContext.UPIPay.checkVPAAvailability(InputcheckVPAAvailability
     inVpaAvailable, new
APICallback<Boolean>() {
          @Override
          public void onSuccess(Boolean result) {


          }
          @Override
          public void onFailure(String sdkErrorCode,String upiErrorCode) {

     });
} catch (UPISDKException e) {
     e.printStackTrace();

}
```

addVPA

Purpose: This API should be called by the Mobile App for adding a Virtual Payment Address.

Inputs:

Input Argument	Description
Object of Class ***InputAddVPA*** having below Members :	Object containing query inputs for Bank Account listing.
• customerRefId(not mandatory)	Existing unique identifier of the Customer. e.g. existing Customer ID or User ID. DataType: String. Length: 20 characters.
• userVPA(*)	User VPA to be Added. DataType: String. Length: 50 characters.
• validUpto(not mandatory)	VPA valid Upto date.(YYYY-MM-DD). DataType: Timestamp.
• maxLimit(*)	Maximum limit per transaction. DataType: Integer
• isOneTimeUse(*)	Boolean : Whether for one time / multiple use.
• forSelectivePayee(not mandatory)	Boolean : Whether for All / Selective Payees
• PaymentAccountID(*)	Payment Bank Account ID(These are the linkAccId that user get from listbankAccount API). DataType: Integer
• CollectAccountID(*)	Collection Bank Account ID(These are the linkAccId that user get from listbankAccount API). DataType: Integer
• List<VpaPayee>(not mandatory)	List of Selected Payees applicable for the VPA

APICallback<T> here T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of String.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Members of Class : *VpaPayee*	Description
beneficiaryId(*)	Beneficiary ID
beneficiaryName(*)	Name of the Beneficiary
paymentAddress(*)	VPA Address of the Beneficiary
status	Status of the beneficiary

Example Usage:

```
try {
        ApplicationClassUpiTest.getApplication().getUpiPay().addVPA(InputAddVPA
        inputAddVPA,
    new APICallback<String>() {
            @Override
            public void onSuccess(String s) {
                        }

            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {
                    });
    } catch (UPISDKException e) {

    }
```

updateVPA

Purpose: This API should be called by the Mobile App for updating details of a Virtual Payment Address.

Inputs:

Input Argument	Description
Object of Class ***InputUpdateVPA*** having below Members :	Object containing query inputs for Bank Account listing.
• customerRefId	Existing unique identifier of the Customer. E.g. existing Customer ID or User ID. DataType: String. Length: 20 characters.
• VPAID(*)	ID for the selected VPA to be modified. DataType: Integer.
• userVPA(*)	User VPA to be Added. DataType: String. Length: 50 characters.
• validUpto(not mandatory)	VPA valid Upto date. DataType: Timestamp.
• maxLimit(*)	Maximum limit per transaction. DataType: Integer.
• isOneTimeUse(*)	Boolean : Whether for one time / multiple use.
• forSelectivePayee	Boolean : Whether for All / Selective Payees
• PaymentAccountID(*)	Payment Bank Account ID. DataType: Integer
• CollectAccountID(*)	Collection Bank Account ID DataType: Integer.
• List<VpaPayee>(not mandatory)	List of Selected Payees applicable for the VPA. DataType: List
APICallback<T> here T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App.

	function onSuccess<T> : *On scessful execution of the API, this callback function returns the success message in form of String.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Members of Class : *VpaPayee*	Description
beneficiaryId(*)	Beneficiary ID
beneficiaryName(*)	Name of the Beneficiary
paymentAddress(*)	VPA Address of the Beneficiary
status	Status of the beneficiary

Example Usage:

```
try {

    ApplicationClassUpiTest.getApplication().getUpiPay().updateVPA(InputUpdateVPA
    inputUpdateVPA,
    new APICallback<String>() {
            @Override
            public void onSuccess(String s) {
            }

            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {
                    });
    }catch (UPISDKException e){

    }
```

CreateDefaultVPA

Purpose: This API is for registering a user with the PSP back end. If the current user is not registered and initialize SDK is called, then the error INVALID_USER is returned. In that case the client needs to call this API with the necessary information for registering the user and proceed with his task afterwards. This API will create the first and default VPA for the user. However, he can create additional VPA later using the addVpa API.

Inputs:

Input Argument	Description
Context	Context of the App
Object of Class ***InputAddDefaultVPA*** having below Members :	Object containing query inputs for Bank Account listing.
• userVpa(*)	User Virtual Payment Address. DataType: String. Length: 50 characters.
• username(*)	User Name. DataType: String. Length: 50 characters.
• userMobile(*)	User Mobile. DataType: String. Length: 10 characters.

• userEmail(*)	User Email. DataType: String. Length: 30 characters.

• userAddress(*)	User Address. DataType: String. Length: 100 characters.
• userMaxPayLimit(*)	User Max Pay Limit. DataType: Integer
• passphrase(*)	User Password . DataType: String. Length: 6 characters.
• idNumber(not mandatory)	ID Number(Like Voter or Aadhar Number)
• idType(not mandatory)	Id Type
• cutomerRefId(not mandatory)	Customer Reference Id. DataType: String. Length: 20
APICallback<T> here T = com.upi.sdk.core.UpiSDKContext.UPIPay	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the expected result as an object of specified type.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Anexture-A****.*

Example Usage:

```
try {
    UpiSDKContext.UPIPay.createDefaultVPA(this, InputAddDefaultVPA  inDefVPA, new
APICallback<UpiSDKContext.UPIPay>() {
        @Override
        public void onSuccess(UpiSDKContext.UPIPay result) {

        }
        @Override
        public void onFailure(String sdkErrorCode,String upiErrorCode) {
        }
    });
} catch (UPISDKException e) {
    e.printStackTrace();
}
```

deleteVPA

Purpose: This API should be called by the Mobile App for updating details of a Virtual Payment Address.

Inputs:

Input Argument	Description
Object of Class ***InputDeleteVPA*** having below Members :	Object containing query inputs for Bank Account listing.

• `setVpa` (*)	Existing vpa for that user. DataType: String. Length: 50 characters.
• vpaId(*)	ID for the selected VPA to be deleted. DataType: Integer.

APICallback<T> here T = String	A generic Callback Interface (having callback functions like Dz*onSuccess*dz and Dz*onFailure*dz) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of String.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Exaple Usage:

```
try {

ApplicationClassUpiTest.getApplication().getUpiPay().deleteVPA(InputDeleteVPA
inputDeleteVPA, new
APICallback<String>() {
            @Override
            public void onSuccess(String s) {
            }

            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {

            }
        });
    } catch (UPISDKException e) {
    }
}
```

listVPA

Purpose: This API should be called by the Mobile App for getting a list of Virtual Payment Addresses added by the User.

Inputs:

Input Argument	Description
Object of Class ***InListVPAQuery*** having below Members :	Object containing query inputs for Bank Account listing.
• customerRefId(not mandatory)	Existing unique identifier of the Customer. E.g. existing Customer ID or User ID. DataType: String. Length: 20 characters.
• fetchActiveVPAOnly(*)	Boolean : Whether to fetch all VPA or Active VPAs only. Active VPA's are those VPA's which already exists in CBS with their respective details.
• resultPerPage(not mandatory)	Number of records to be shown in a page.
APICallback<T>	A generic Callback Interface (having callback

here T = Vpa[]	functions like D*zonSuccess*dz and D*zonFailure*dz) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of Array of Vpa objects.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and*

Members of Class : *Vpa*	**Description**
ownerId	User ID / Customer ID
Vpa	VPA Address
vpaID	VPA ID for the VPA
validUpto	VPA valid Upto date(YYYY-MM-DD)
maxLimit	Maximum Limit per Transaction for the VPA
selectivePayee	Boolean : Whether for All / Selective Payees
isOneTime	Boolean : Whether for one time / multiple use.
defaultVPA	Boolean : Whether this is default VPA or not.
Status	Status for the VPA. e.g Active/Inactive etc.
acctLinkRuleId	Rule ID applicable for the VPA
payAccId	Payment Bank Account ID
payNickName	Nick Name for Payment Bank Account
collectAccId	Collection Bank Account ID
collectNickName	Nick Name for Collection Bank Account
List<VpaPayee>	List of Selected Payees applicable for the VPA
Members of Class : *VpaPayee*	**Description**
beneficiaryId(*)	Beneficiary ID
beneficiaryName(*)	Name of the Beneficiary
paymentAddress(*)	VPA Address of the Beneficiary
Status	Status of the beneficiary

Example Usage:

```
InListVPAQuery inQuery = new InListVPAQuery();
try {
    ApplicationClassUpiTest.getApplication().getUpiPay().listVPA(inQuery, new
APICallback<Vpa[]>() {
        @Override
        public void onSuccess(Vpa[] result) {

        }
        @Override
        public void onFailure(String sdkErrorCode,String upiErrorCode) {

} catch (UPISDKException e) {
    e.printStackTrace();
```

```
}
```

initiatePayment

Purpose: This API should be called by the Mobile App for initiating a Payment request.

Inputs:

Input Argument	Description
Object of Class ***InPayRequest*** having below Members :	Object containing query inputs for Bank Account listing.
• UpiPayer(*)	Object containing Payer Here the payer may be with aadhar number or account ifsc rather than VPA. If payer use aadhar number then his VPA will be- His aadhar number+aadhaar.npci If payer use account/IFSC number then his VPA will be- His Account no+@+His IFSC code+ifsc.npci
• List<UpiPayee>(*)	List of Objects containing Payees Detail
• note(*)	Notes/narration for the Collection Request. DataType: String. Length: 50 characters.
• validUpto(not mandatory)	In case of Collection Request, how long the request will remain active is indicated by this date. DataType: Timestamp.
APICallback<T> here T = OutPayRequest	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On scessful execution of the API, this callback function returns the success message in form of OutPayRequest object.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Example Usage:

```
InPayRequest payRequest = new InPayRequest();


        payRequest.setTxn_type("PAY");
        payRequest.setNote("");
```

```
        UpiPayer payer = new UpiPayer();
        payer.setName(username);
        payer.setVirtualPaymentAddress(vpa);
        payRequest.setBank_name(linkAcc.getAccPvdName());
         payRequest.setBankAcNumber(acc_number);
//
            if (total_result != null && total_result.length > 0) {

                for (LinkAccountDetails account : total_result) {
                    CredAllowed credAllowed = new CredAllowed();
                    if (sub_string_nickname.equalsIgnoreCase(account.getNickName())
&& account.getCredDataType() != null && account.getCredLength() != null) {

credAllowed.setSubtype(CredentialSubtype.fromValue(account.getCredDataType()));
                        credAllowed.setDlemgth(account.getCredLength());
                        credAllowed.setDtype("ALPH | NUM");
                        payRequest.setCredAllowed(credAllowed);
                    }

                }


            }
            payer.setAcNickName(sub_string_nickname);
            payer.setCurrency("INR");

payer.setAmount(BigDecimal.valueOf(Double.valueOf(payItemDetails.getTxnAmount())));
            payRequest.setPayer(payer);
        } catch (Exception e) {
            e.printStackTrace();
        }

        if (AppConstant.PAYEE_LIST != null) {
            for (Map.Entry<String, PayeeItem> entry :
AppConstant.PAYEE_LIST.entrySet()) {
                UpiPayee payee = new UpiPayee();
                PayeeItem item = entry.getValue();
                payee.setVirtualPaymentAddress(item.getPayee_VPA());
                payee.setName(item.getPayee_nick_name());

payee.setAmount(BigDecimal.valueOf(Double.valueOf(item.getAmount())));
                payRequest.getPayees().add(payee);
            }


payee.setAmount(BigDecimal.valueOf(Double.valueOf(item.getAmount())));
                    payRequest.getPayees().add(payee);
                }
        }




ApplicationClassUpiTest.getApplication().getUpiPay().initiatePayment(payRequest,
new APICallback<OutPayRequest>() {
                @Override
                public void onSuccess(OutPayRequest result) {



                }
```

```
            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {};
            }
        });
    } catch (UPISDKException e) {
                }
```

initiateCollect

Purpose : This API should be called by the Mobile App for initiating a Collection request.

Inputs:

Input Argument	Description
Object of Class ***InPayRequest*** having below Members :	Object containing query inputs for Bank Account listing.
• UpiPayer(*)	Object containing Payer Details
• List<UpiPayee>(*)	List of Objects containing Payee Details
• note(*)	Notes/narration for the Collection Request. DataType: String. Length: 50 characters.
• validUpto(not mandatory)	In case of Collection Request, how long the request will remain active is indicated by this date. DataType: Timestamp.
APICallback<T> here T = OutPayRequest	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of OutPayRequest object.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Members of Class : *OutPayRequest*	Description
txnId	Transaction ID for the Collection Request
Status	Transaction status for the Collection Request

Example Usage:

```
try {

InPayRequest mInPayRequest = new InPayRequest();
UpiPayer mUpiPayer = new UpiPayer();
mUpiPayer.setVirtualPaymentAddress(payerVpa.getText().toString().trim());
mUpiPayer.setAmount(BigDecimal.valueOf(Double.valueOf(paymentAmount.getText().toStr
ing())));

    mInPayRequest.setValidUpto(edtxt_validUpto.getText().toString() + " " +
edtxt_validUptotime.getText().toString());


mInPayRequest.setNote(notes.getText().toString().trim());
UpiPayee upiPayee = new UpiPayee();
upiPayee.setAcNickName(payeeAcNickName.getText().toString());
upiPayee.setAmount(BigDecimal.valueOf(Double.valueOf(paymentAmount.getText().toStri
ng())));
upiPayee.setVirtualPaymentAddress(payeeVpa.getText().toString().trim());
mInPayRequest.setTxn_type("COLLECT");
mInPayRequest.getPayees().add(upiPayee);
mInPayRequest.setPayer(mUpiPayer);


ApplicationClassUpiTest.getApplication().getUpiPay().initiateCollect(mInPayRequest,
new APICallback<OutPayRequest>() {
         @Override
         public void onSuccess(OutPayRequest outPayRequest) {

         }
         @Override
         public void onFailure(String sdkErrorCode,String upiErrorCode) {


} catch (UPISDKException e) {
    e.printStackTrace();
}
```

1.2. getTransactionHistory

Purpose : This API should be called by the Mobile App for obtaining a list of Transactions (Pay/Collect) made by the User.

Inputs:

Input Argument	Description
Object of Class ***InTxnHistoryQuery*** having below Members :	Object containing query inputs for Transaction History listing.
• customerRefId(not mandatory)	User ID / Customer ID for the User. DataType: String. Length: 20 characters. DataType: String. Length: 20
• fromDate(not mandatory)	Start date of Query. DataType: Timestamp.
• endDate(not mandatory)	End date of Query. DataType: Timestamp
• resultPerPage(not mandatory)	Number of records to be shown in a page DataType: Integer.
APICallback<T> here T = TransactionHistory[]	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of Array of TransactionHistory object .* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI*

Members of Class : TransactionHistory	Description
txnId	Transaction ID for the Pay/Collection Request
upiTxnId	Transaction ID specific to UPI
txnType	Transaction Type : Pay/Collect
txnTotalAmt	Total Transaction Amount
txnStatus	Transaction status for the Pay/Collection Request
note	Note specified for the transaction
linkAccId	Bank Account ID affected by the pay/collect request
initiatedBy	Reeqest Initiating user ID
creationTs	Creation Time Stamp
disputeDetails	Return the dispute array if dispute raised against that transaction
payeeName	Payee Name
errorCode	ErrorCode of transaction if present

Example Usage:

```
try {
     if(ApplicationClassUpiTest.getApplication().getUpiPay()==null)
          return;
     else{
          InTxnHistoryQuery inTxnHistoryQuery=new InTxnHistoryQuery();

ApplicationClassUpiTest.getApplication().getUpiPay().getTransactionHistory(inTxnHis
toryQuery, new APICallback<List<TransactionHistory>>() {
               @Override
               public void onSuccess(List<TransactionHistory> transactionHistories) {

     }
}catch (UPISDKException e){
     e.printStackTrace();

}
```

getPendingCollectRequests

Purpose : This API should be called by the Mobile App for obtaining a list of Pending Collection Request.

Inputs:

Input Argument	Description
Object of Class ***InPendingTxnQuery*** having below Members :	Object containing query inputs for Pending Collection Request listing.
• customerRefId(*)	User ID / Customer ID for the User. DataType: String. Length: 20 characters.
• fromDate(not mandatory)	Start date of Query. DataType: Timestamp.
• endDate(not mandatory)	End date of Query. DataType: Timestamp
• resultPerPage(not mandatory)	Number of records to be shown in a page. DataType: Integer.
APICallback<T> here T =OutPendingTxn[]	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of Array of TransactionHistory object .* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI*

Members of Class : OutPendingTxn	Description

txnId	Transaction ID for the Collection Request

upiTxnId	Transaction ID specific to UPI
txnType	Transaction Type : Collect
txnTotalAmt	Total Transaction Amount
txnStatus	Transaction status for the Collection Request
note	Note specified for the transaction
linkAccId	Bank Account ID affected by the collect request
initiatedBy	Reqest Initiating user ID
creationTs	Creation Time Stamp
minAmount	Minimum Amount

Example Usage:

```
ApplicationClassUpiTest.getApplication().getUpiPay().getPendingCollectRequests(InPe
    ndingTxnQuery  inPe ndingTxnQuery, new APICallback<OutPendingTxn[]>() {
                        @Override
                        public void onSuccess(OutPendingTxn[] pendingTxns) {

                        }

                        @Override
                        public void onFailure(String sdkErrorCode,String
                        upiErrorCode) {

                        }

                } catch (UPISDKException e) {
                }
```

SendOtpRequest

Purpose: This API should be called by the Mobile App for OTP Request for forget password validation.

Inputs:

Input Argument	Description
APICallback<T> where T = Boolean	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of Boolean.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI*

Example Usage:

```
try {
     ApplicationClassUpiTest.getApplication().getUpiPay().sendOtpRequest(new
APICallback<Boolean>() {
          @Override
          public void onSuccess(Boolean result) {


          }
```

```
        @Override
        public void onFailure(String sdkErrorCode,String upiErrorCode) {

} catch (UPISDKException e) {
    e.printStackTrace();
}
```

BindDevice

Purpose: This API should be called by the Mobile App for bind device with that particular user where his device id and transaction details are bind with backend for every time check whether the requests are coming from a valid device or not.

Inputs:

Input Argument	Description
APICallback<T> here T = Boolean	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of Boolean .* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI*

Example Usage:

```
try {
        ApplicationClassUpiTest.getApplication().getUpiPay().bindDevice("", new
 APICallback<Boolean>() {
            @Override
            public void onSuccess(Boolean result) {
                        }

            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {
                        }
        });
    } catch (UPISDKException e) {
    }
}
```

AddBeneficiary

Purpose: This API should be called by the Mobile App for adding Beneficiary.

Inputs:

Input Argument	Description
Object of Class ***InputBeneficiary*** having	Object containing query inputs for Pending

below Members :	Collection Request listing.
• beneficiaryName (*)	Beneficiary Name
• paymentAddress (*) • aadharNumber(*) • accNo(*) • ifsc(*) • beneficiaryType(*)(ENUM of Beneficiary Type) • All this parametres are mandatory based on their type that mentioned below- 1)If user select beneficiary type vpa then he has to provide beneficiary name and payment address and beneficiary type as vpa 2)If user select beneficiary type aadhar then he has to provide beneficiary name and aadhar number and beneficiary type as aadhar 3) If user select beneficiary type acc_ifsc then he has to provide beneficiary name and account no and ifsc and beneficiary type as account_ifsc	Virtual Payment Address
APICallback<T> where t = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of String.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Anexture-A****.*

Example Usage:

```
 try {
      if (ApplicationClassUpiTest.getApplication().getUpiPay() == null) return;
      else {

InputBeneficiary inputBeneficiary = new InputBeneficiary();

if(beneficiary_type_selected.equalsIgnoreCase("aadhar")){

    inputBeneficiary.setBeneficiaryName(beneficiaryName);

    inputBeneficiary.setAadhar_num(aadhar_number);

    inputBeneficiary.setBeneficiaryType(BeneficiaryType.AADHAR);

}else if(beneficiary_type_selected.equalsIgnoreCase("acc_ifsc")){

    inputBeneficiary.setBeneficiaryName(beneficiaryName);

    inputBeneficiary.setAccount_no(acc_no);

    inputBeneficiary.setIfsc(ifsc);

    inputBeneficiary.setBeneficiaryType(BeneficiaryType.ACCOUNT_IFSC);

}else{

    inputBeneficiary.setBeneficiaryName(beneficiaryName);

    inputBeneficiary.setPaymentAddress(vpa);

    inputBeneficiary.setBeneficiaryType(BeneficiaryType.VPA);

}ApplicationClassUpiTest.getApplication().getUpiPay().addBeneficiary(inputBeneficiar y, new
APICallback<String>() {@Override public void onSuccess(String s) {

                }
                @Override
                public void onFailure(String sdkErrorCode,String upiErrorCode) {
      }
 } catch (UPISDKException e) {
 }
```

UpdateBeneficiary

Purpose: This API should be called by the Mobile App for adding Beneficiary.

Inputs:

Input Argument	Description
Object of Class ***InputUpdateBeneficiary*** having below Members :	Object containing query inputs for Pending Collection Request listing.
• beneficiaryId (*)	Beneficiary ID. DataType: Integer.

• beneficiaryName (*)	Beneficiary Name
APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of String.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI*

Example Usage:

```
try {
     if (ApplicationClassUpiTest.getApplication().getUpiPay() == null)
          return;
     else {
          InputUpdateBeneficiary inputUpdateBeneficiary = new
InputUpdateBeneficiary();
          inputUpdateBeneficiary.setBeneficiaryName(beneficiaryName);
          inputUpdateBeneficiary.setBeneficiaryId(benfId);

ApplicationClassUpiTest.getApplication().getUpiPay().updateBeneficiary(inputUpdateB
eneficiary, new APICallback<String>() {
               @Override
               public void onSuccess(String s) {

               }
               @Override
               public void onFailure(String sdkErrorCode,String upiErrorCode) {

     }
} catch (UPISDKException e) {
     e.printStackTrace();

}
```

DeleteBeneficiary

Purpose : This API should be called by the Mobile App for deleting Beneficiary.

Inputs:

Input Argument	Description
Object of Class ***InputUpdateBeneficiary*** having below Members :	Object containing query inputs for Pending Collection Request listing.
• beneficiaryId (*)	Beneficiary ID
APICallback<T>	A generic Callback Interface (having callback

where T = String	functions like Dz*onSuccess*sdz and Dz*onFailure*dz) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of the API, this callback function returns the success message in form of String.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-4***

Example Usage:

```
try {
     if (ApplicationClassUpiTest.getApplication().getUpiPay() == null)
          return;
     else {
          InputDeleteBeneficiary inputDeleteBeneficiary = new
InputDeleteBeneficiary();
          inputDeleteBeneficiary.setBeneficiaryId(benfId);

ApplicationClassUpiTest.getApplication().getUpiPay().deleteBeneficiary(inputDeleteB
eneficiary, new APICallback<String>() {
               @Override
               public void onSuccess(String s) {

               }
               @Override
               public void onFailure(String sdkErrorCode,String upiErrorCode) {

     }
} catch (UPISDKException e) {
     e.printStackTrace();
     }
```

ListBeneficiary

Purpose: This API should be called by the Mobile App to generate list of Beneficiaries.

Inputs:

Input Argument	Description
Object of Class ***InListAddedBeneficiaryQuery*** having below Members :	Object containing query inputs for List of Beneficiaries.
• customerRefId(not mandatory)	Customer Ref Id. DataType: String. Length: 20
• resultPerPage(not mandatory)	List of Beneficiaries per page. DataType: Integer

APICallback<T> where T = BeneficiaryDetails[]	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T> :*** *On successful execution of*

	the API, this callback function returns the success message in form of String. ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Members of Class : BeneficiaryDetails	Description
userId	User Id of the Beneficiary
beneficiaryId	Beneficiary Id
beneficiaryName	Beneficiary Name
paymentAddress	Virtual Payment Address
status	Status

Example Usage:

```
InListAddedBeneficiaryQuery inputBeneficiary = new InListAddedBeneficiaryQuery();
try {
     if (ApplicationClassUpiTest.getApplication().getUpiPay() == null)
          return;
     else {

ApplicationClassUpiTest.getApplication().getUpiPay().listBeneficiary(inputBeneficia
ry, new APICallback<BeneficiaryDetails[]>() {
               @Override
               public void onSuccess(BeneficiaryDetails[] beneficiaryDetailses) {

               }
               @Override
               public void onFailure(String sdkErrorCode,String upiErrorCode) {


               }
          });
     }
} catch (UPISDKException e) {
     e.printStackTrace();

}
```

Authorize or Decline CollectionRequest

Purpose: This API should be called by the Mobile App for authorizing a particular Pending Collection Request.

Inputs:

Input Argument	Description
Object of Class ***InPaymentAuthorization***	Object containing query inputs for Pending

having below Members :	Collection Request listing.
• txnId(*)	Transaction ID for the Collection Request. DataType: Integer
• authorized(*)	Boolean : true for authorize and false for decline
• accountNickName(*)	Payer Account Nick Name affected by the transaction. DataType: String. Length: 50
• paymentAddress(*)	Payer VPA Address affected by the transaction. DataType: String. Length: 50 characters.
• amount(*)	Authorized Transaction amount DataType: Double.
• currency(not mandatory)	Currency of Transaction. DataType: String. Length: 10 characters.
APICallback<T> here T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<T>:*** *On successful execution of the API, this callback function returns the success message in form of String.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI*

Example Usage:

```
InPaymentAuthorization payAuth = new InPaymentAuthorization();
    payAuth.setTxnId("");
payAuth.setAccountNickName("");
 CredAllowed credAllowed = new CredAllowed();

    credAllowed.setSubtype(CredentialSubtype.fromValue(account.getCredDataType()));
                            credAllowed.setDlemgth(account.getCredLength());
                            credAllowed.setDtype("NUM");
                            payAuth.setCredAllowed(credAllowed);
                        }
                    }
                }

    payAuth.setAmount(BigDecimal.valueOf(Double.valueOf(npci_pendingTxnsItems.getTxnI
    ndividualAmt())));
                payAuth.setAuthorized(isAuthorized);
                payAuth.setCurrency(npci_pendingTxnsItems.getCurruency());

    payAuth.setPayeePaymentAddress(npci_pendingTxnsItems.getPayeeAddress());
                payAuth.setPayeeName(npci_pendingTxnsItems.getPayeeName());
                payAuth.setPaymentAddress(npci_pendingTxnsItems.getPaymentAddress());

            try {

    ApplicationClassUpiTest.getApplication().getUpiPay().authorizeCollectionRequest(p
    ayAuth, new APICallback<Boolean>() {
                    @Override
                    public void onSuccess(Boolean result) {
                    }
```

```
                    @Override
                    public void onFailure(String sdkErrorCode,String upiErrorCode) {
                });
            } catch (UPISDKException e) {

            }
```

Sent SMS

Purpose: This API should be called by the Mobile App for sending sms to VMN.

Input: Activity context and Sim Subscription Id from which user wants to send the message. If it's a single sim device put simSubscriptionId as null otherwise send the simSubscriptionId of that sim that the user has selected.

Return: Return encrypted message as string.

Example Usage:

```
try {

        UpiSDKContext.sentSMS(Context context, new IMessageListener() {
            @Override
            public void getMessage(String s) {
                    }

                    @Override
                    public void onFailure(String sdkErrorCode,String upiErrorCode) {

                    }
                    },<simsubscriptionId>);
```

Validate Mobile Number

Purpose: This API should be called by the Mobile to validate the mobile from which the SMS has been delivered.

Input: Same message that has been received during SentSMS API.

Return: Return the same mobile number from which the SMS has been delivered.

Example Usage:

```
  UpiSDKContext.validateMobileNUmber(<message>, new APICallback<String>() {
                @Override
                public void onSuccess(String mobile_number) {

                }

                @Override
                public void onFailure(String sdkErrorCode,String upiErrorCode) {
```

```
                                    }
            });
        }
    });
```

FORGET PIN

Purpose: This API should be called by the Mobile to reset pin if user forget that.

Process Flow:

1) requestOTPforForgetPIN: This api is called for otp request.
Input: UesrId or Mobile Number(*)
Context- Application Context.

Return: Return registered mobile number.

Example Usage:

```
UpiSDKContext.requestOTPforForgetPIN(uID, context,new APICallback<String>() {
                    @Override
                    public void onSuccess(String mobileNo) {

                    }

                    @Override
                    public void onFailure(String sdkErrorCode,String upiErrorCode) {
                                        }
                });
```

2) validateOTPforForgetPIN: This api is called for validate otp request.
Input: OtpServiceDetails class that contains two fields setMobileNo , setOtpVal where in mobile user needs to pass registered mobile number that has been obtained from previous flow and the otp sent to his mobile number.

Return: Return Success message.

Example Usage:

```
UpiSDKContext.validateOTPforForgetPIN(otpServiceDetails, new APICallback<String>() {
        @Override
        public void onSuccess(String s) {
        }

        @Override
        public void onFailure(String sdkErrorCode,String upiErrorCode) {

        }
    });
```

3) updatePIN: This api is called next to update the pin.

Input: UpdatePassword class that contains two fields userId and userPassword where user have to pass his user id and new PIN.

Note: If user have more than one app with same mobile number then to update password via mobile number user have to send SDK the application context by **setContext** field.

Return: Return Success.

Example Usage:

```
UpiSDKContext.updatePIN(updatePassword,new APICallback<String>() {
        @Override
        public void onSuccess(String s) {
        }

        @Override
        public void onFailure(String sdkErrorCode,String upiErrorCode) {
        }
    });
```

Get Account Balance

Purpose: This API should be called to retrieve the account balance of a bank account.

Inputs:

Input Argument	Description
Object of Class **BalanceEnquiryRequest** having below Members : • nickname(*)-Nick name of bank • bankAcNumber(*)-Account number • mobileNumber(*)-Mobile Number • bank_name(*)-Bank Name	Object containing query inputs for Balance Enquiry
APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<*** `BalanceEnquiryResponse` ***>:*** *On successful execution of the API, this callback function returns the balance in* `accountBalance field`. ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI*

Example Usage:

```
try {
                                    final BalanceEnquiryRequest balanceEnquiryRequest =
new BalanceEnquiryRequest();

balanceEnquiryRequest.setBankAcNumber("")
;
balanceEnquiryRequest.setNickName"");
balanceEnquiryRequest.setMobileNumber("");
balanceEnquiryRequest.setBank_name("");
ApplicationClassUpiTest.getApplication().getUpiPay().getAccountBalance(balanceEnqui
ryRequest,
                                            new APICallback<BalanceEnquiryResponse>() {
                                                @Override
                                                public void
onSuccess(BalanceEnquiryResponse balEnquiryResp) {
balEnquiryResp.getAccountBalance();

                                                    }

                                                    @Override
                                                    public void onFailure(String
                                                    sdkErrorCode,String upiErrorCode) {

    }
                                    } catch (UPISDKException e) {

                        }
                });
```

Register Mobile Banking

Purpose: This API should be called to register mobile banking for a user.

Inputs:

Input Argument	Description
Object of Class **ReqRegMobile** having below Members :	Object containing query inputs for Registetr mobile banking
• cardDigit(*)-Last 6 digits of your card • expDate(*)-MM/YY • mobileNu mber(*)-Mobile number • bank_name(*)-Bank Name • ifsc(*)-Bank IFSC Code • vpa(not mandatory)-User default VPA • mobRegFormat • LinkAccountDetails(*) • AccNo(*)	DataType: String. Length: 6 characters. DataType: MM/yy DataType: String. Length: 10 characters. DataType: String. Length: 11 characters. DataType: String. Length: 20 characters. DataType: String. Length: 20 characters. Bank account object of the bank that user selected Bank account number

APICallback<T> where T = RespRegMobile	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<*** `RespRegMobile` ***> :*** *On successful execution of the API, this callback function returns success as string.*

	function onError (String sdkErrorCode,String upiErrorCode): *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Example Usage:

```
try {
        ReqRegMobile reqRegMobile = new ReqRegMobile();
        reqRegMobile.setCardDigit("");
        reqRegMobile.setExpDate("");
        reqRegMobile.setIfsc("");
        reqRegMobile.setBank_name("");
        reqRegMobile.setLinkAccountDetails(<LinkAccountDetailsObject of user
      selected Account>);
        reqRegMobile.setMobRegFormat(<If not null get it from listBankPSP API
       for that bank user have selected>);


    reqRegMobile.setMobile(ApplicationClassUpiTest.getSharedPreference().getMobileN
    umber());

    reqRegMobile.setVpa(ApplicationClassUpiTest.getSharedPreference().getUser_ID()
    + "@" + ApplicationClassUpiTest.getSharedPreference().getPSPBank());


    ApplicationClassUpiTest.getApplication().getUpiPay().registerMobileBanking(reqR
    egMobile, new APICallback<RespRegMobile>() {
            @Override
            public void onSuccess(RespRegMobile respCbsRegMobile) {
                        }

            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {

    } catch (UPISDKException e) {
    }
```

Change MPIN

Purpose: This API should be called to change MPIN.

Inputs:

Input Argument	Description
Object of Class **InputChangeMPIN** having below Members :	Object containing query inputs for Change MPIN

• CredAllowed-Cred block along with credType,CreddLength,DLength • bank_name(*)-Bank Name • ifsc(*)-Bank IFSC Code	DataType: Credbolck DataType: String. Length: 11 characters. DataType: String. Length: 20 characters.
APICallback<T> where T = ResponseSetCre dential	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<*** `ResponseSetCre`

	> *: On successful execution of the API, this callback function returns success as string in status object.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Example Usage:

```
try {
            InputChangeMPIN inputChangeMPIN = new InputChangeMPIN();
            inputChangeMPIN.setIfsc(ifsc);

                        inputChangeMPIN.setBank_name(linkAcc.getAccPvdName());
                        CredAllowed credAllowed = new CredAllowed();

    credAllowed.setSubtype(CredentialSubtype.fromValue(account.getCredDataType()));
                        credAllowed.setDlemgth(account.getCredLength());
                        credAllowed.setDtype("ALPH | NUM");
                        inputChangeMPIN.setCredAllowed(credAllowed);


    ApplicationClassUpiTest.getApplication().getUpiPay().changeMPIN(inputChangeMPIN
    , new APICallback<ResponseSetCre>() {
                @Override
                public void onSuccess(ResponseSetCre responseSetCre) {
                }

                @Override
                public void onFailure(String sdkErrorCode,String upiErrorCode) {


                }
            });
        } catch (UPISDKException e) {
        }

    }
```

Deregister Profile

Purpose: This API should be called to deregister existing UPI user profile which will not be activated for two years. In VPA deletion user will be able to delete that VPA if no ongoing transaction is associated with that VPA. But later he can add the same VPA again. But for profile deregister user won't be able to access that default VPA that is associated with that user account.

Inputs:

Input Argument	Description

`String userId`(*)	Object containing query inputs for Profile Deragister. DataType: String. Length:20 characters.

	Registration
APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<*** `String` ***> :*** *On successful execution of the API, this callback function returns success as string.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Anexture-A****.*

Example Usage:

```
try {


    ApplicationClassUpiTest.getApplication().getUpiPay().deregisterProfile(<USER_ID
    >, new APICallback<String>() {
            @Override
            public void onSuccess(String s) {
            }

            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {
                    });

    } catch (UPISDKException e) {
        }
```

Raise Dispute

Purpose: This API should be called to raise dispute against failure transactions.

Inputs:

Input Argument	Description
`RaiseDispute`	Object containing query inputs for raise dispute

`txnId`(*)-`Transaction Id`
`issue`(*)-`Issuer Message`
`ownerId`(*)-User Id
	DataType: String. Length: 40 characters.
	DataType: String. Length: 100 characters.
	DataType: String. Length: 20 characters.

APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<*** `RaiseDispute`

	> : *On successful execution of the API, this callback function returns disputeId,status,creation_ts* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Example Usage:

```
try {

    ApplicationClassUpiTest.getApplication().getUpiPay().raiseDispute
 (RaiseDispute raisedispute, new APICallback<String>() {
            @Override
            public void onSuccess(RaiseDispute s) {
            }

            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {
                    });

    } catch (UPISDKException e) {
        }
```

Transaction Details

Purpose: This API should be called to get a transaction details of a transaction.

Inputs:

Input Argument	Description
`InputTxnHistoryDetail`	Object containing query inputs for transaction details
`txnId`(*)-UPI Transaction Id `ownerId`(*)-User Id	DataType: String. Length: 40characters. DataType: String. Length: 20 characters.

APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<*** `TxnHistoryDetails` `Added minAmount,expiryDate,errorCode` ***> :*** *On successful execution of the API, this callback function returns details of transaction with* `TxnHistoryDetails object containing the whole transaction details of a transaction.` ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of*

	function onError (String sdkErrorCode,String upiErrorCode): *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Example Usage:

```
try {

    ApplicationClassUpiTest.getApplication().getUpiPay().transactionHistoryetails

 (InputTxnHistoryDetails
inputTxnHistDetails, new APICallback<String>() {
            @Override
            public void onSuccess(TxnHistoryDetails s) {
            }

            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {
                    });

    } catch (UPISDKException e) {
        }
```

Change password

Purpose: This API should be called to Change the User password.

Inputs:

Input Argument	Description
changePassword	Object containing query inputs for change user password
setUserId(*)	User id in string. DataType: String. Length: 20 characters.

`setUserPassword(*)`	Old password value in string. DataType: String. Length: 6 characters.
`setNewPassword(*)`	New password value in string. DataType: String. Length: 6 characters.
APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess< String>*** ***> :*** *On successful execution of the API, this callback function returns a string message.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Example Usage:

```
ChangePassword changePassword=new ChangePassword();

   changePassword.setUserId("");

   changePassword.setUserPassword("");

   changePassword.setNewPassword("");



try {
ApplicationClassUpiTest.getApplication().getUpiPay().changePassword(changePassword, new
APICallback<String>() {

                @Override

                public void onSuccess(String s) {

                }

                @Override

                }

            });

        }catch (UPISDKException e){
```

Get Transaction details By RefID

Purpose: This API should be called to get the Transaction details of a particular transaction by RefId

Inputs:

Input Argument	Description
refId(*)	String refId of that transaction.
Mode(*)	This is a Enum value. Values are PAY,COLLECT,ALL

APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<*** `TxnHistoryDetails` ***>:*** *On successful execution of the API, this callback function returns details of transaction with* `TxnHistoryDetails object` ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error*

Example Usage:

```
try{

ApplicationClassUpiTest.getApplication().getUpiPay().getTransactionDetailsByRefId(Stri
ng refId, Mode mode, new APICallback<TxnHistoryDetails>() {
            @Override
            public void onSuccess(TxnHistoryDetails txnHistoryDetails) {

            }

            @Override
            public void onFailure(String sdkErrorCode,String upiErrorCode) {

            }
        });
    }catch (Exception e){
    }
         }
```

ActivateVPA

Purpose: This API should be called activate a VPA

Inputs:

Input Argument	Description
`ownerId(*)`	String userId
`vpa(*)`	String VPA
`vpaId(*)`	String VpaId

APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess< `String`*** ***> :*** *On successful execution of the API, this callback function returns string message.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code*

Example Usage:

```
InputActivateVPA inputActivateVPA=new InputActivateVPA();
inputActivateVPA.setOwnerId(<userId>);
inputActivateVPA.setVpa(<vpa>);
inputActivateVPA.setVpaId(<vpaId>);
try{
    ApplicationClassUpiTest.getApplication().getUpiPay().activateVPA(inputActivateVPA,
new APICallback<String>() {
        @Override
        public void onSuccess(String result) {

        }

        @Override
        public void onFailure(String sdkErrorCoe, String upiErrorCode) {

        }
    });

}catch (UPISDKException e){
}
```

DeActivateVPA

Purpose: This API should be called de-activate a VPA

Inputs:

Input Argument	Description
`ownerId(*)`	String userId
`vpa(*)`	String VPA
`vpaId(*)`	String VpaId

APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess< `String`*** ***> :*** *On successful execution of the API, this callback function returns string message.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code*

Example Usage:

```
InputActivateVPA inputActivateVPA=new InputActivateVPA();
inputActivateVPA.setOwnerId(<userId>);
inputActivateVPA.setVpa(<vpa>);
inputActivateVPA.setVpaId(<vpaId>);
try{
    ApplicationClassUpiTest.getApplication().getUpiPay().deActivateVPA(inputActivateVPA,
new APICallback<String>() {
        @Override
        public void onSuccess(String result) {

        }

        @Override
        public void onFailure(String sdkErrorCoe, String upiErrorCode) {

        }
    });

}catch (UPISDKException e){
    }
```

ListNotificationMessage

Purpose: This API should be called to get all the notification messages.

Inputs:

Input Argument	Description
`userId(*)`	String userId
`fromDate(*)`	long milliseconds

`toDate(*)`	long milliseconds
APICallback<N> where T = Notifications[ ] that contains these objects- Message(String) Status(String) Date(String) UserId(String) Event Type(String)	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<*** `Notifications[]>` ***>*** *: On successful execution of the API, this callback function returns* `Notifications array.` ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Example Usage:

```
InputNotification inQuery = new InputNotification();
inQuery.setUserId(<userId>);
inQuery.setFromDate(<milliseconds>);
inQuery.setToDate(<milliseconds>);

try {

ApplicationClassUpiTest.getApplication().getUpiPay().listNotificationMessages(inQuery,
new APICallback<Notifications[]>() {
        @Override
        public void onSuccess(Notifications[] result) {

        }

        @Override
        public void onFailure(String sdkErrorCode, String upiErrorCode) {
        }
    });
} catch (UPISDKException e) {

}
```

AddSchedule

Purpose: This API should be called to add a schedule.

Inputs:

Input Argument	Description

schdType(*)	Enum ScheduleTYpe
schdName(*)	String schedule name
ownerId(*)	String userId
ownVpa(*)	String userVpa
otherVpa(*)	String othersVpa
amount(*)	Double amount
frequncy(*)	Enum ScheduleFrequency
recurDate(*)	String yyyy-mm-dd
startDate(*)	String yyyy-mm-dd
alertBefore(*)	String alerBefore days
recurCount(*)	int recurCount. If recur date is present recur count must be 0.
APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess< String*** ***> :*** *On successful execution of the API, this callback function returns string message.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code*

Example Usage:

```
InputSchedule inQuery = new InputSchedule();
inQuery.setSchdType(ScheduleType.C);
inQuery.setSchdName(<name>);
inQuery.setOwnerId(<userId>);
inQuery.setOwnVpa(<ownVpa>);
inQuery.setOtherVpa(<othersVpa>);
inQuery.setAmount(<amount>);
inQuery.setFrequncy(ScheduleFrequency.ONCE);
inQuery.setRecurDate(<YYYY-MM-DD>);
inQuery.setStartDate(<YYYY-MM-DD>);
inQuery.setRecurCount(<recur_count>);
inQuery.setAlertBefore(<alert_before_days>);
try {
    ApplicationClassUpiTest.getApplication().getUpiPay().addSchedule(inQuery, new
APICallback<String>() {
        @Override
        public void onSuccess(String result) {



        }

        @Override
        public void onFailure(String sdkErrorCode, String upiErrorCode) {

        }
    });
} catch (UPISDKException e) {
}
```

UpdateSchedule

Purpose: This API should be called to update a schedule.

Inputs:

Input Argument	Description
schdType(*)	Enum ScheduleTYpe
schdName(*)	String schedule name
ownerId(*)	String userId
ownVpa(*)	String userVpa
otherVpa(*)	String othersVpa
amount(*)	Double amount

`frequncy(*)`	Enum ScheduleFrequency
`recurDate(*)`	String yyyy-mm-dd
`startDate(*)`	String yyyy-mm-dd
`alertBefore(*)`	String alerBefore days
`recurCount(*)`	int recurCount. If recur date is present recur count must be 0.
`schdId(*)`	int scheduleId
`status`	String status
APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess< `String`*** ***> :*** *On successful execution of the API, this callback function returns string message.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code*

Example Usage:

```
Schedule inQuery = new Schedule();
inQuery.setSchdId(<scheduleId>);
inQuery.setSchdType(ScheduleType.C);
inQuery.setSchdName(<name>);
inQuery.setOwnerId(<userId>);
inQuery.setOwnVpa(<ownVpa>);
inQuery.setOtherVpa(<othersVpa>);
inQuery.setAmount(<amount>);
inQuery.setFrequncy(ScheduleFrequency.ONCE);
inQuery.setRecurDate(<YYYY-MM-DD>);
inQuery.setStartDate(<YYYY-MM-DD>);
inQuery.setRecurCount(<recur_count>);
inQuery.setAlertBefore(<alert_before_days>);
inQuery.setStatus(<status>);

try {
    ApplicationClassUpiTest.getApplication().getUpiPay().updateSchedule(inQuery, new
APICallback<String>() {
        @Override
        public void onSuccess(String result) {



        }

        @Override
        public void onFailure(String sdkErrorCode, String upiErrorCode) {

        }
    });
} catch (UPISDKException e) {
}
```

Delete Schedule

Purpose: This API should be called delete a schedule.

Inputs:

Input Argument	Description
`ownerId(*)`	String userId
`schdId(*)`	`String schdId`

APICallback<T> where T = String	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess< String*** ***> :*** *On successful execution of the API, this callback function returns string message.* ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code*

Example Usage:

```
InputDeleteSchedule inQuery = new InputDeleteSchedule();
inQuery.setOwnerId(<userId>);
inQuery.setSchdId(<scheduleId>);

try {
    ApplicationClassUpiTest.getApplication().getUpiPay().deleteSchedule(inQuery, new
APICallback<String>() {
        @Override
        public void onSuccess(String result) {

        }

        @Override
        public void onFailure(String sdkErrorCode, String upiErrorCode) {
        }
    });
} catch (UPISDKException e) {

}
```

List Schedule

Purpose: This API should be called to get all the schedule created by user.

Inputs:

Input Argument	Description
userId(*)	String userId

APICallback<N> where T = Schedule[ ] that contains these Schedule Object that described earlier.	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<*** `Schedule[]>` ***>*** *: On successful execution of the API, this callback function returns* `Schedule array.` ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Example Usage:

```
try {
    ApplicationClassUpiTest.getApplication().getUpiPay().listSchedule(<userId>,
new APICallback<Schedule[]>() {
        @Override
        public void onSuccess(Schedule[] schedules) {

        }

        @Override
        public void onFailure(String s, String s1) {

        }
    });
} catch (UPISDKException e) {
    }
```

SetLanguageForCL

Purpose: This API should be called to set the language of CL page.

Input: `Enum CLLanguage.`

SetAPIKey

Purpose: This API should be called to set the api key.

Input: `String apikey. This is not a mandatory parameter. User may put null on api key if not known. SDK is capable to fetch the same by App Id.`

VerifyVAE

Purpose: This API should be called to verify the merchant VPA.

Inputs:

Input Argument	Description
`vpa(*)`	String vpa
APICallback<VaeDetails> where VaeDetails contains these parametres. vaeAddr(String) vaeName(String) vaeLogo(String) vaeUrl(String)	A generic Callback Interface (having callback functions like OnSuccess and OnFailure) that needs to be implemented within the App. ***function onSuccess<*** `Schedule[]` ***>:*** *On successful execution of the API, this callback function returns* `Schedule array.` ***function onError (String sdkErrorCode,String upiErrorCode):*** *In case of error this callback method returns the error code inform of string SDK errorcode or UPI errorcode.If any UPI error occurs SDK error code is ERR00096 and corresponding UPI error code will be there in UPI errorcode.* ***Annexure-A****.*

Example Usage:

```
try {
    ApplicationClassUpiTest.getApplication().getUpiPay().verifyVAE(<VPA>, new
APICallback<VaeDetails>() {
            @Override
            public void onSuccess(VaeDetails vaeDetails) {


            }

            @Override
            public void onFailure(String s, String s1) {

            }
        });
    }
} catch (UPISDKException e) {
    e.printStackTrace();
    LogFile.writeLog(CommonUtils.getStacktrace(e));
}
```

Destroy SDK

Purpose: This API should be called to destroy the SDK instance and NPCI's common library instance. When the user try to exit from app on activity's onDetroy() method it's preferable to be called.

Example Usage:

```
if (ApplicationClassUpiTest.getApplication().getUpiPay() != null)

    ApplicationClassUpiTest.getApplication().getUpiPay().destroySDKInstance();
```

3. Anexture – A

Error Code	DESCRIPTION
ERR00000	SYSTEM_ERROR-
ERR00001	INVALID_USER_ID_OR_PASSWORD-
ERR00002	ACCOUNT_LOCKED_CONTACT_YOUR_ADMINSTRATOR-
ERR00003	MOBILE_NOT_VERIFIED-
ERR00004	USER_ALREADY_EXISTS-
ERR00005	VPA_ALREADY_EXISTS-
ERR00006	MOBILE_NUMBER_ALREADY_EXISTS-
ERR00007	EMAIL_ADDRESS_ALREADY_EXISTS-
ERR00008	MAX_PAY_LIMIT_SHOULD_BE_LESS_THAN_SPECFIED_LIMIT
ERR00009	OTP_NOT_VALID-
ERR00010	NICK_NAME_ALREADY_EXISTS-
ERR00011	MOBILE_NUMBER_REQUIRED-
ERR00012	MMID_REQUIRED-
ERR00013	ACCOUNT_NUMBER_REQUIRED-
ERR00014	IFSC_CODE_REQUIRED-
ERR00015	DUPLICATE_ACCOUNT_LINK-
ERR00016	NULL_VALUE//not found-
ERR00017	VALID_UPTO_SHOULD_BE_GREATER_THAN_THE_CURRENT
ERR00018	OTP_FOR_SELECT_PAYEE_BUT_MISSING_PAYEE-
ERR00019	OTP_FOR_ALL_PAYEE_BUT_PAYEE_PROVIDED-
ERR00020	INVALID_PAY_ACCOUNT-
ERR00021	INVALID_COLLECT_ACCOUNT-
ERR00022	INVALID_BENEFICIARY-
ERR00023	VPA_REQUIRED-
ERR00024	INVALID_USER_ID-
ERR00025	USER_REQUIRED-
ERR00026	BENEFICIARY_NAME_REQUIRED-
ERR00027	BENEFICIARY_VPA_REQUIRED-
ERR00028	BENEFICIARY_NAME_EXISTS_FOR_THE_USER-
ERR00029	BENEFICIARY_VPA_EXISTS_FOR_THE_USER-
ERR00030	NO_ACCOUNT_INFORMATION_PROVIDED-
ERR00031	NICK_NAME_REQUIRED-
ERR00032	IDENTITY_REQUIRED-
ERR00033	IDENTITY__TYPE_REQUIRED-
ERR00034	IDENTITY__NUMBER_REQUIRED-
ERR00035	ACCOUNT_HAS_LIVE_VPA-

ERR00036	INVALID_IDENTITY_TYPE-
ERR00037	DEVICE_INFO_REQUIRED-
ERR00038	DEVICE_MOBILE_REQUIRED-
ERR00039	DEVICE_MOBILE_NUMBER_NOT_SAME_AS_USER_MOBILE-
ERR00040	DEVICE_BLOCK_MISSING-
ERR00041	DEVICE_FIGERPRINT_REQUIRED-
ERR00042	PASSWORD_REQUIRED-
ERR00043	INVALID_PASSOWORD-
ERR00044	REQUIRE_NEW_PASSOWORD-
ERR00045	NEW_PASSWORD_SAME_AS_OLD-
ERR00046	BENEFICIARY_ID_REQUIRED-
ERR00047	LINK_ACCOUNT_ID_REQUIRED-
ERR00048	ACCOUNT_DOESNOT_EXISTS-
ERR00049	ACCOUNT_DOESNOT_EXISTS-
ERR00050	VPA_ID_REQUIRED-
ERR00051	OTP_REQUIRED-
ERR00052	OTP_NOT_SEND-
ERR00053	OTP_NOT_VALID-
ERR00054	RULE_NAME_REQUIRED-
ERR00055	PAY_TYPE_REQUIRED-
ERR00056	INVALID_PAY_TYPE-
ERR00057	RULE_DETAILS_REQUIRED-
ERR00058	MAX_PAY_LIMIT_REQUIRED-
ERR00059	SCHD_NAME_REQUIED-
ERR00060	SCHD_TYPE_REQUIED-
ERR00061	INVALID_SCHD_AMT-
ERR00062	INVALID_SCHD_FREQ-
ERR00063	INVALID_SCHD_DATE-
ERR00064	INVALID_SCHD_CUR_FREQ-
ERR00065	INVALID_SCHD_START_DATE-
ERR00066	INVALID_SCHD_START_TIME-
ERR00067	SCHEDULE_ID_REQUIRED-
ERR00068	VPA_NOT_EXISITS-
ERR00069	INVALD_SCHEDULE_TYPE-
ERR00070	NULL_VALUE-
ERR00071	EMPTY_STRING-
ERR00072	INVALID_FORMAT-
ERR00073	MIN_LENGTH_REQUIRED-
ERR00074	MAX_LENGTH_REQUIRED-
ERR00075	MIN_VALUE_REQUIRED-
ERR00076	MAX_VALUE_REQUIRED-
ERR00077	NOT_NEUMERIC-
ERR00078	RULE_ID_REQUIRED-
ERR00079	RULE_HAS_ACTIVE_VPA-
ERR00080	VPA_MARKED_AS_DEFAULT-
ERR00081	ONETIME_NOT_ALLOWED-
ERR00082	QUESTION_ID_REQUIRED-
ERR00083	ANSWER_REQUIRED-
ERR00084	INVALID_QUESTION_ID-
ERR00085	VPA_NAME_SAME_AS_DEFAULT-
ERR00086	EXPIRY_DATE_NOT_ALLOWED-
ERR00087	INVALID_DEFAULT_VPA_NAME-
ERR00088	DEFAULT_VPA_NOT_ALLOWED-
ERR00089	TRANSACTION_IN_PROGRESS-

ERR00090	MOBILE_NUMBER_VERIFICATION_FAILED-
ERR00091	ACCOUNT_HAS_RULE-

ERR000103	SDK_NOT_INITIALIZED
ERR000104	INVALID_IFSC
ERR000105	MAXLIMIT_PER_TRANS_REQUIRED
ERR000106	MISSING_PAYEE_LIST
ERR000107	VPAID_REQUIRED
ERR000108	INVALID_PAYER
ERR000109	INVALID_PAYEELIST
ERR000110	TXN_AMOUNT_MISSMATCH
ERR000111	INVALID_PAYMENT_ACC
ERR000112	INVALID_COLLECTION_ACC

ERR00092	VPA_HAS_SCHEDULE-
ERR00093	BENEFICARY_SCHEDULE_EXISTS-
ERR00094	SYSTEM_ERROR-
ERR00095	INVALID_BANKACC
ERR00096	UPI_ERROR
ERR00097	REGISTRATION_ERROR
ERR00098	ILLEGAL_INPUTS
ERR00099	INVALID_USER
ERR000100	LOGIN_ERROR
ERR000101	TXN_AUTH_FAILED
ERR000102	INVALID_NICKNAME
ERR000113	MERCHANT_ADDRESS_IS_NOT_WHITE_LISTED

NOTE:

For all kind of UPI errors SDK error code will be ERR000096 and corresponding UPI error codes are in the format ERR000XX i.e ERR000ZM, ERR000XC, etc. SDK will return the UPI error code as in same format like SDK error codes.

For NPCI's back button handle user have to implement **iStateListener** by which he will be able to get state as IDLE and ACTIVE according to the user scenario which he has to maintained in his own application.