JAMBOPAY SDK FOR NODEJS

> This is the unofficial node.js integration module for integration with the Jambopay API PDF DOCUMENTATION

All of the methods called return promises and can therefore be easily chained

INSTALLATION

```
npm i jambopay
```

---

1. GETTING STARTED

```
 Object.assign( global, { JP: require('jambopay')});
```

```
The above makes the {jambopay} module globally available as 'JP'
in your project
```

```
APP_KEY = "YOUR_JAMBOPAY_ISSUED_APP_KEY";
```

```
The above identifies your application to Jambopay
```

---

2. SET THE ENDPOINT URL (`.set_endpoint_url(remote_host_address)`)

(Not necessary if using 192.168.11.22 [for tests] via VPN)

Example:

```
 JP.set_endpoint_url( 'http://192.168.11.22' );
```

---

3. LOGIN (*`.login(parmeters_object,header)`*)

parameters_object = {username,password,grant_type}

Example:

```
JP.login({
            username: req.body.email,
            password: req.body.password,
            grant_type: "agency"
        })
```

```
.then(d => {

    //@ All subsequent operations are placed here

})
.catch(e => {

    //@ Handle the error

})
```

This module handles basic authentication re-negotiation with Jambopay so that you don't have to. **`This fetches an authentication token for use with each request.`**

HEADERS PARAMETER

The `headers` parameter is ***`optional`*** but should generally be in the format :

```
{
    "authorization": JAMBOPAY_TOKEN,
    "app_key":"Your-Application-Key"
}
```

The above applies for each of the methods that expect a `headers` parameter.

USER METHODS

1. NAIROBI COUNTY COUNCIL PAYMENTS (`.ncc`)

A). UNIVERSAL BUSINESS PERMIT (`.ubp`)

Available methods

---

a). ***.get_sbp_classes({PhoneNumber,ActivityID},headers)***

This method gets all the business categories for all businesses as classified by the Nairobi County Government.

The ***ActivityID*** (basically a class/category into which the business falls) in the response will be used on the ***.get_sbp_sub_classes*** call.

Example:

```
JP
.ncc
.ubp
.get_sbp_classes({
            PhoneNumber : "...",
            ActivityID  : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). ***.get_sbp_sub_classes({PhoneNumber,ID},headers)***

The above request gets you the business sub-Classes related to the Activity ID.

The ***sub-ClassID*** on the response will be used in the ***.get_sbp_activity_charges*** call.

Example:

```
JP
.ncc
.ubp
.get_sbp_sub_classes({
```

```
            PhoneNumber : "...",
            ID          : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

c). *.get_sbp_activity_charges({PhoneNumber,subclassID},headers)*

The *ChargeID* parameter in the response can be used as a filtering parameter when making subsequent calls

Example:

```
JP
.ncc
.ubp
.get_sbp_activity_charges({
            PhoneNumber : "...",
            subclassID  : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

d). *.get_sbp_activity_sub_counties(headers)*

This call lists all the sub-counties within Nairobi county. The client chooses this and the subsequent call in order to help determine

his/her business' approximate location.

Example:

```
JP
.ncc
.ubp
.get_sbp_activity_sub_counties()
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

e). *.get_sbp_wards({id},headers)*

This call lists all the wards within a given Sub-county in the County.

Example:

```
JP
.ncc
.ubp
.get_sbp_wards({
    id : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

f). *.prepare_payment({Year,Names,PhoneNumber,PhonePin,IDNumber,ActivityCode,ActivityName,SBP*
}, headers)

Acceptable Parameters

Parameter	Type	Required	Description
BusinessID	String	-	
Year	Int	R	
TransactionID	String	-	
Names	String	R	Isaac Juma
BillNumber	String	-	
BillStatus	String	-	
PhoneNumber	String	R	254724659560
PhonePin	Int	R	1234
IDNumber	String	R	1
ActivityCode	Int	R	
ActivityName	String	R	Painting
SBPSubClassID	Int	R	1114
RelativeSize	String	R	1
WardName	String	R	
ZoneName	String	R	
AcceptedTnC	Boolean	R	1
WardCode	Int	R	4
ZoneCode	Int	R	136
BusinessName	String	R	Dcyntech
BusinessRegistrationNumber	String	-	
Pin	String	R	JJITYTU767868
VAT	String	-	JHJH2323
IDDocumentNumber	String	R	28734420
BusinessClassificationDetails	String	R	1212
OtherBusinessClassificationDetails	String	-	2121
PremisesArea	String	R	121
NumberOfEmployees	Int	R	121212
AnnualSBPAmount	Double	-	0
POBox	String	R	8782 Nairobi
PostalCode	String	R	00100
Town	String	R	Nairobi
Telephone1	String	R	254717803383
Telephone2	String	R	254717803383
FaxNumber	String	-	0717803383
Email	String	R	ijuma@webtribe.com
PhysicalAddress	String	R	dsfsdfsd
PlotNumber	String	R	343ffdf
Building	String	R	34634
BuildingCategory	String	R	Non-Storey
Floor	String	R	n/a
RoomStallNumber	String	R	12
ContactPersonName	String	R	Simiyu Robert

Parameter	Type	Required	Description
ContactPersonDesignation	String	R	
ContactPersonPOBox	String	R	281 Soy
ContactPersonPostalCode	String	R	30105
ContactPersonTown	String	R	
ContactPersonTelephone1	String	R	0717803383
ContactPersonTelephone2	String	R	0717803383
ContactPersonFaxNumber	String	-	3434
PaidBy	String	R	SAMUEL KAHARA
ApprovalStatus	Int	-	0
PaymentTypeID	Int	R	1
SBPPaymentType	Int	R	1
ReceiptNumber	String	-	
LocalReceiptNumber	String	-	
ReferenceNumber	String	-	
BankName	String	-	
BranchName	String	-	
Exclusions[0].ID	String	-	ChargeID excludedfrom payment
Exclusions[1].ID	String	-	ChargeID excluded from payment

The *transactionID* in the response is used to commit the transaction on the `.commit_payment` call.

Example:

```
JP
.ncc
.ubp
.prepare_payment({
            Year        : "...",
            Names       : "...",
            PhoneNumber : "...",
            PhonePin    : "...",
            IDNumber    : "...",
            ActivityCode: "...",
            ActivityName: "...",
            SBPSubClassID: "...",
            RelativeSize: "...",
            WardName    : "...",
            ZoneName    : "...",
            AcceptedTnC : "...",WardCode,ZoneCode,BusinessName: "...",
            Pin         : "...",
            IDDocumentNumber: "...",
            BusinessClassificationDetails: "...",
            PremisesArea: "...",
```

```
            NumberOfEmployees: "...",
            AnnualSBPAmount: "...",
            PostalCode  : "...",
            Town        : "...",
            Telephone1  : "...",
            Telephone2  : "...",
            FaxNumber   : "...",
            Email       : "...",
            PhysicalAddress: "...",
            PlotNumber  : "...",
            Building    : "...",
            BuildingCategory: "...",
            Floor       : "...",
            RoomStallNumber: "...",
            ContactPersonName: "...",
            ContactPersonDesignation: "...",
            ContactPersonPOBox: "...",
            ContactPersonPostalCode: "...",
            ContactPersonTown: "...",
            ContactPersonTelephone1: "...",
            ContactPersonTelephone2: "...",
            PaidBy      : "...",
            ApprovalStatus: "...",
            PaymentTypeID: "...",
            SBPPaymentType: "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

g). *.commit_payment({PhoneNumber,SBPPaymentType,Tendered,TransactionID,PaymentTypeID,PaidBy,*

Parameter	Type	Required	Description
MerchantID	String	R	Merchant identifier e.g. NCC
PhoneNumber	String	R	Agent Phone number

Parameter	Type	Required	Description
SBPPaymentType	Int	R	1
Tendered	Double	R	Confirmation amount of the SBP total bill
TransactionID	String	R	Unique ID from the POST response.
PaymentTypeID	Int	R	1
PaidBy	String	R	Person Paying for the registration
PhonePin	String	R	Pin of the wallet being debited

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.ncc
.ubp
.commit_payment({
            PhoneNumber : "...",
            SBPPaymentType: "...",
            Tendered    : "...",
            TransactionID: "...",
            PaymentTypeID: "...",
            PaidBy      : "...",
            PhonePin    : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

UBP RENEWAL

---

a). *.get_business({LicenseID,Year,PhoneNumber}, headers)*

This call gets the business details for the business as registered using the `LicenseID` (*`BusinessID`*).

The *`ActivityID`* provided in the response is to be used in the subsequent calls.

Example:

```
JP
.ncc
.ubp
.get_business({
    LicenseID       : "...",
    Year            : "...",
    PhoneNumber     : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). ***.get_business_class_id({ActivityID,PhoneNumber} ,headers)***

Gets the business Classes related to the `ActivityID`.

The `ClassID` on the response will be used in the subsequent calls.

Example:

```
JP
.ncc
.ubp
.get_business_class_id({
    ActivityID : "...",
    PhoneNumber: "..."
})
.then(d => {

    //@ Do something

})
```

```
.catch(e => {

    //@ Handle the error

})
```

---

c). ***.get_business_sub_class_id({ID,PhoneNumber},headers)***

The returned ***SubClassID*** is used in the `.prepare_payment` call.

Example:

```
JP
.ncc
.ubp
.get_business_sub_class_id({
    ID          : "...",
    PhoneNumber : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

d). ***.prepare_payment({MerchantID,BusinessID,Year,PhoneNumber,PaymentTypeID,SBPSubClassID},he***

Acceptable Parameters

Parameter	**Type**	**Required**	**Description**
MerchantID	String	R	Merchant Identifier
BusinessID	String	R	License ID
Year	Int	R	2017
PhoneNumber	String	R	PhoneNumber of the wallet
PaymentTypeID	Int	R	1
SBPSubClassID	Int	R	ID from the ***getsbpsubclasses*** call

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.ncc
.ubp
.prepare_payment({
    MerchantID          : "...",
    BusinessID          : "...",
    Year                : "...",
    PhoneNumber         : "...",
    PaymentTypeID       : "...",
    SBPSubClassID       : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

e). *.commit_payment({PhoneNumber,MerchantID,TransactionID,Pin,Year,AcceptedTnC,ContactPerson*

Acceptable Parameters

Parameter	Type	Required	Description
PhoneNumber	String	R	Wallet PhoneNumber
MerchantID	String	R	Merchant identifier
TransactionID	String	R	Unique ID from the POST response
Pin	String	R	Wallet Pin
Year	Int	R	Year of business permit renewal
AcceptedTnC	Boolean	R	Accept terms and conditions, e.g. True
ContactPersonName	String	R	Contact Person
IDDocumentNumber	String	R	Number of the ID document
Telephone1	String	R	Telephone1
Building	String	-	
Floor	String	-	Floor

Parameter	Type	Required	Description
RoomStallNumber	String	-	Room Stall Number
ZoneCode	Int	R	Business location Zone Code
WardCode	Int	R	Business location Ward Code
ZoneName	String	R	Business location Zone Name
WardName	String	R	Business location Ward Name
PlotNumber	String	R	Business location Plot Number

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.ncc
.ubp
.commit_payment({
    PhoneNumber         : "...",
    MerchantID          : "...",
    TransactionID       : "...",
    Pin                 : "...",
    Year                : "...",
    AcceptedTnC         : "...",
    ContactPersonName   : "...",
    IDDocumentNumber    : "...",
    Telephone1          : "...",
    ZoneCode            : "...",
    WardCode            : "...",
    ZoneName            : "...",
    WardName            : "...",
    PlotNumber          : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

B). LAND RATES (`.land_rates`)

Available methods

---

a). ***`.prepare_payment({PlotNumber,PhoneNumber,PaymentTypeID},headers)`***

Acceptable Parameters

Parameter	Type	Required	Description
MerchantID	String	R	Merchant identifier
PlotNumber	String	R	Plot number
PhoneNumber	String	R	Customers phone number
PaymentTypeID	Int	R	Mode of payment e.g. 1 for cash, etc

The *`transactionID`* from the response is used in the *`.commit_payment`* call.

Example:

```
JP
.ncc
.land_rates
.prepare_payment({
    PlotNumber : "...",
    PhoneNumber: "...",
    PaymentTypeID: "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

b). ***`.commit_payment({TransactionID,Amount},headers)`***

Acceptable Parameters

Parameter	Type	Required	Description
MerchantID	String	R	Merchant identifier
TransactionID	String	R	Unique transaction reference number
Amount	decimal	R	transaction amount

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.ncc
.land_rates
.commit_payment({
    TransactionID: "...",
    Amount      : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

C). PARKING (.parking)

Available methods

DAILY PARKING

---

a). *.get_daily_parking_items (headers)*

This call pulls the required information to determine the amount to be paid.i.e Vehicle types and Zones.

Example:

```
JP
.ncc
.parking
.get_daily_parking_items()
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). ***.prepare_payment({MerchantID,IsDaily,PlateNumber,PaymentTypeID,PhoneNumber,ZoneCodeID,Ch***

Acceptable Parameters

Parameter	Type	Required	Description
MerchantID	String	R	Merchant identifier
IsDaily	bool	R	Whether parking is daily or not e.g 1 or 0; true or false
PlateNumber	string	R	Car registration Number e.g. KDF888W
PaymentTypeID	Int	R	Mode of payment e.g. 1
PhoneNumber	string	R	Customer phone number
ZoneCodeID	Int	R	Unique id describing the parking zone e.g.10
ChargeID	Int	R	Unique id describing the parking charges e.g.10
DurationID	Int	O	Unique id describing the parking duration

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.ncc
.parking
.prepare_payment({
    MerchantID      : "...",
    IsDaily         : "...",
    PlateNumber     : "...",
```

```
    PaymentTypeID   : "...",
    PhoneNumber     : "...",
    ZoneCodeID      : "...",
    ChargeID        : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

c). ***.commit_payment({TransactionID,Amount},headers)***

Acceptable Parameters

Parameter	Type	Required	Description
MerchantID	String	R	Merchant identifier
TransactionID	String	R	Transaction id
PhoneNumber	String	R	Client Phone Number
PaidBy	String	R	Client Name
Amount	Decimal	R	Amount paid for parking

The response parameters will contain a valid ***ReceiptNumber*** which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.ncc
.parking
.commit_payment({
    TransactionID: "...",
    Amount      : "..."
})
.then(d => {

    //@ Do something
```

```
})
.catch(e => {

    //@ Handle the error

})
```

SEASONAL PARKING

---

a). ***.get_seasonal_parking_items (headers)***

This gets initialization data (fees and charges) to determine the amount to be paid. This includes: vehicle type and Parking duration.

Example:

```
JP
.ncc
.parking
.get_seasonal_parking_items()
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). ***.prepare_payment({MerchantID,IsDaily, PlateNumber,PaymentTypeID,PhoneNumber,ZoneCodeID,C***

Acceptable Parameters

Parameter	Type	Required	Description
MerchantID	String	R	Merchant identifier
IsDaily	Bool	R	Whether parking is daily or not

Parameter	Type	Required	Description
PlateNumber	String	R	Car plate no.
PaymentTypeID	Int	R	Mode of payment
PhoneNumber	string	R	Customer phone number
ZoneCodeID	Int	R	Unique id describing the parking zone
ChargeID	Int	R	Unique id describing the parking charges
DurationID	Int	R	Unique id describing the parking duration

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.ncc
.parking
.prepare_payment({
    MerchantID  : "...",
    IsDaily     : "...",
    PlateNumber : "...",
    PaymentTypeID: "...",
    PhoneNumber : "...",
    ZoneCodeID  : "...",
    ChargeID    : "...",
    DurationID  : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

c). *.commit_payment({MerchantID,TransactionID},headers)*

Acceptable Parameters

Parameter	Type	Required	Description
MerchantID	String	R	Merchant identifier

Parameter	Type	Required	Description
TransactionID	String	R	Transaction id

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.ncc
.parking
.commit_payment({
    TransactionID   : "...",
    MerchantID      : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

D). SACCO PARKING (.sacco_parking)

Available methods

---

a). *.get_sacco_parking_items (headers)*

This call pulls the required information to determine the amount to be paid. This include Vehicle types and duration types.

Example:

```
JP
.ncc
.sacco_parking
```

```
.get_sacco_parking_items()
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). *.prepare_payment({SaccoName,Vehicles: [{VehicleType,Duration,RegistrationNumber}],Paymen*

Acceptable Parameters

Parameter	Required	Type	Description
SaccoName	R	String	name of sacco
Vehicles	***Array fields***		
	VehicleType	R	int
	Duration	R	int
	RegistrationNumber	R	string
PaymentTypeID	R	int	Mode of payment e.g. 1 - cash

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
let params = {
            SaccoName : "..." ,
            Vehicles: [
                {
                    vehicleType :"...",
                    Duration    :"...",
                    RegistrationNumber:"..."
                }
            ],
            PaymentTypeID   :"..."
        };
```

JP

```
.ncc
.sacco_parking
.prepare_payment(
    params
)
.then(d=>{
    //@ Do something with the response
})
.catch(e => {
    //@ Handle the error
});
```

---

c). *.commit_payment({},headers)*

Acceptable Parameters

The response parameters will contain a valid ***ReceiptNumber*** which denotes that the transaction has been successfully commited and completed.

Parameter	Type	Required	Description
TransactionID	String	R Transaction id	

Example:

```
let params = {
            TransactionID   :"..."
        };
```

```
JP
.ncc
.sacco_parking
.commit_payment(
    params
)
.then(d=>{
    //@ Do something with the response
})
.catch(e => {
    //@ Handle the error
});
```

D). HOUSE RENT (`.house_rent`)

Available methods

---

a). *`.get_estates ( headers )`*

This call gets a list of estates with houses and markets that belong to Nairobi County

Example:

```
JP
.ncc
.house_rent
.get_estates()
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). *`.get_residences ( { ID }, headers ))`*

This call gets the list of residences within an estate using the EstateID

Example:

```
JP
.ncc
.house_rent
.get_residences({
    ID          : "..."
})
.then(d => {
```

```
    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

c). *.prepare_payment({TransactionID,Names,Amount,Adjustment,Currentbalance,CustomerNames,Las*

Acceptable Parameters

Output	Type	Description
TransactionID	string	Unique transaction reference number
Names	string	Tenant names
Amount	decimal	Amount to pay
Adjustment	decimal	any adjustment accrued in payment
Currentbalance	decimal	Current balance
CustomerNames	string	Customer names
LastBillMonth	DateTime	Last month bill was paid
MonthlyOtherCharges	decimal	Other monthly charges
MonthlyRent	decimal	Monthly rent
OtherArrears	decimal	Other arrears
PhysicalAddress	decimal	Physical address
RentArrears	decimal	Rent arrears
RentDueDate	DateTime	Rent due date
UHN	string	Unique house number

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.ncc
.house_rent
.prepare_payment({
                    TransactionID   : "...",
                    Names           : "...",
                    Amount          : "...",
                    Adjustment      : "...",
                    Currentbalance  : "...",
```

```
                    CustomerNames   : "...",
                    LastBillMonth   : "...",
                    MonthlyOtherCharges     : "...",
                    MonthlyRent     : "...",
                    OtherArrears    : "...",
                    PhysicalAddress : "...",
                    RentArrears     : "...",
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

d). ***.commit_payment({TransactionID,Amount},headers)***

Acceptable Parameters

Parameter	Type	Required	Description
TransactionID	String	R	Unique transaction reference number
Amount	decimal	R	Amount to pay

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.ncc
.house_rent
.commit_payment({
    TransactionID   : "...",
    Amount          : "..."
})
.then(d => {

    //@ Do something
```

```
})
.catch(e => {

    //@ Handle the error

})
```

D). E-CONSTRUCTION (`.e_construction`)

Available methods

---

a). ***.prepare_payment({InvoiceNumber,PaymentTypeID,PhoneNumber,PaidBy},headers)***

Acceptable Parameters

Parameter	Type	Required	Description
InvoiceNumber	string	R	invoice number
PaymentTypeID	int	R	Mode of payment
PhoneNumber	string	R	The client's phone number
PaidBy	string	R	The name of the client

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.ncc
.e_construction
.prepare_payment({
    InvoiceNumber: "...",
    PaymentTypeID: "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {
```

```
    //@ Handle the error

})
```

---

b). *.commit_payment({TransactionID,PhoneNumber},headers)*

Acceptable Parameters

Parameter	Type	Required	Description
TransactionID	string	R	Unique transaction reference number
PhoneNumber	string	R	User phone number

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.ncc
.e_construction
.commit_payment({
    TransactionID   : "...",
   PhoneNumber      : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

E). LIQUOR (.liquor)

Available methods

---

a). ***.prepare_payment({ Date,LicenseNumber,PhoneNumber,PaymentTypeID},headers)***

Acceptable Parameters

Parameter	Type	Required	Description
Date	DateTime	R	date
LicenseNumber	String	R	License number
PhoneNumber	String	R	Customer
PaymentTypeID	int?	R	Mode of payment

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.ncc
.liquor
.prepare_payment( Date          : "...",
                  LicenseNumber : "...",
                  PhoneNumber   : "...",
                  PaymentTypeID : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). ***.commit_payment({TransactionID},headers)***

Acceptable Parameters

Parameter	Type	Required	Description
TransactionID	String	R	Unique transaction reference number

Parameter	Type	Required	Description

The response parameters will contain a valid *`ReceiptNumber`* which denotes that the transaction has been successfully commited and completed.

`Example:`

```
JP
.ncc
.liquor
.commit_payment({
    TransactionID   : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

E). MISCELLANEOUS (`.misc`)

Available methods

---

a). *`.prepare_payment({PhoneNumber,BillNumber,PaymentTypeID},headers)`*

Acceptable Parameters

Parameter	Type	Required	Description
PhoneNumber	string	`R`	Customer phone number
BillNumber	string	`R`	Unique Bill reference number gotten from county
PaymentTypeID	int	`R`	Mode of payment

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.ncc
.misc
.prepare_payment({
                    PhoneNumber     : "...",
                    BillNumber      : "...",
                    PaymentTypeID   : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). *.commit_payment({TransactionID},headers)*

Acceptable Parameters

Parameter	Type	Required	Description
TransactionID	String	R	Unique transaction reference number

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.ncc
.misc
.commit_payment({
    TransactionID   : "..."
})
.then(d => {
```

```
    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

2. NAIROBI WATER PAYMENTS (`.nairobi_water`)

---

a). *`.get_bill({AccountNumber},headers)`*

This call pulls the applicable charges for Nairobi Water.

Example:

```
JP
.nairobi_water
.get_bill({
    AccountNumber   : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). *`.prepare_payment({AccountNumber,PhoneNumber,Names,Amount,PaymentTypeID},h`*

Acceptable Parameters

Parameter	Type	Required	Description
AccountNumber	String	R	Nairobi Water customer account number
PhoneNumber	String	R	User phone number
Names	String	R	Customer account names
Amount	decimal	R	Transaction amount
PaymentTypeID	int?	R	Mode of payment

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.nairobi_water
.prepare_payment({
            AccountNumber   : "...",
            PhoneNumber     : "...",
            Names           : "...",
            Amount          : "...",
            PaymentTypeID   : "..."

        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

c). *.commit_payment({TransactionID},headers)*

Acceptable Parameters

Parameter	Type	Required	Description
TransactionID	String	R	Unique transaction reference number

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.nairobi_water
.commit_payment({
            TransactionID   : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

3. NATIONAL HOSPITAL INSURANCE FUND PAYMENTS (`.nhif`)

---

a). *.prepare_payment({AccountNumber,IsCorporate,PhoneNumber,Amount,PaymentTyp*

Acceptable Parameters

Parameter	Type	Required	Description
AccountNumber	String	R	Customer account number at NHIF i.e. member number
IsCorporate	Bool	R	Whether transaction is corporate or individual, use **False**

Parameter	Type	Required	Description
PhoneNumber	String	R	Customers phone number
Penalty	Decimal	*O*	Amount to pay as penalty.
Amount	Decimal	R	Transaction amount.
PaymentTypeID	Int	R	Mode of payment e.g. 1 for cash, etc

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.nhif
.prepare_payment({
            AccountNumber   : "...",
            IsCorporate     : "...",
            PhoneNumber     : "...",
            Amount          : "...",
            PaymentTypeID   : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). *.commit_payment({TransactionID,PhoneNumber,Amount},headers)*

Acceptable Parameters

Parameter	Type	Required	Description
TransactionID	String	R	Unique transaction reference number

Parameter	Type	Required	Description
PhoneNumber	String	R	Customers phone number
Amount	decimal	R	Transaction amount

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.nhif
.commit_payment({
            TransactionID   : "...",
            PhoneNumber     : "...",
            Amount          : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

4. KASNEB PAYMENTS (.kasneb)

---

a). *.prepare_payment({PhoneNumber,BillNumber,PaidBy,PaymentTypeID},headers)*

Acceptable Parameters

Parameter	Description
PhoneNumber	The phone number of the individual making the payment

Parameter	Description
BillNumber	The bill number or invoice number to which payment is to be made
PaidBy	The names of the individual making the payment
PaymentTypeID	The type of payment being made. For all agents we request this be defaulted to 1 which stands for cash transactions

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.kasneb
.prepare_payment({
            PhoneNumber : "...",
            BillNumber  : "...",
            PaidBy      : "...",
            PaymentTypeID: "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). *.commit_payment({PhoneNumber,TransactionID,ExaminationCenterID,Tendered},*

Acceptable Parameters

Parameter	Description
PhoneNumber	The phone number of the individual making the payment

Parameter	Description
TransactionID	This is the unique identifier for the payment to be processed. This value comes from the previous output from a property with the same name
ExaminationCenterID	Null able parameter. Only applies to payments that have an exam center applied to them.
Tendered	This is an optional parameter denoting the total amount of money given by the client. It is either same to or more than the Amount field and can be used to compute change owed to the client

The response parameters will contain a valid *`ReceiptNumber`* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.kasneb
.commit_payment({
            TransactionID   : "...",
            PhoneNumber     : "...",
            Tendered        : "...",
            ExaminationCenterID: "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

5. JAMBOPAY WALLET PAYMENTS (`.wallet`)

---

a). *.prepare_payment({PhoneNumber,Amount,PaymentTypeID},headers)*

Acceptable Parameters

Parameter	Type	Required	Description
PhoneNumber	string	R	User phone number
Amount	decimal	R	Transaction amount
PaymentTypeID	Int	R	Mode of payment.e.g. 1 for cash, 2 for cheque, etc

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.wallet
.prepare_payment({
            PhoneNumber   : "...",
            Amount        : "...",
            PaymentTypeID : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). *.commit_payment({TransactionID,PhoneNumber},headers)*

Acceptable Parameters

Parameter	Type	Required	Description
TransactionID	String	R	Unique transaction reference number
PhoneNumber	String	R	User phone number

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.wallet
.commit_payment({
            TransactionID   : "...",
            PhoneNumber     : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

6. JAMBOPAY PAYMENT REPORTS (`new JP.Reports(reporting_stream [ e.g. parking, saccoparking, rent,nhif,ubp])`)

INITIALIZE THE QUERY OBJECT

e.g

```
For NCC Parking

const ParkingReport = new JP.Reports(`parking`);
```

---

a). *.get_transactions({},headers)*

Acceptable parameters

Parameter	Type	Required	Description
PlateNumber	string	*O*	Id of the business
Index	int	*O*	Zero based index of the pages of the transactions to return- each page has 30 transactions
TransactionID	string	*O*	Unique transaction reference number
StartDate	Date	*O*	Least date to fetch transactions from
EndDate	Date	*O*	Latest date to fetch transactions
UserID	string	*O*	Unique user id
TransactionStatus	byte	*O*	State of the transaction (prepared-0 or completed-1)

Sample Usage ***(for ncc parking)***

```
ParkingReport
.get_transactions( {})
.then( d => {
    //@ Do something
})
.catch( e => {
    ///@ Handle the error
})
```

7. KENYA POWER & LIGHTING Co. PAYMENTS (`.kplc`)

---

a). *.prepare_payment({PhoneNumber,Amount,PaymentTypeID,MeterNumber,MerchantID*

Acceptable Parameters

Parameter	Type	Required	Description
PhoneNumber	string	R	User phone number
Amount	decimal	R	Transaction amount
PaymentTypeID	Int	R	Mode of payment.e.g. 1 for cash, 2 for cheque, etc
MeterNumber	Int	R	The client's meter number
MerchantID	Int	R	'KPLC' in this case

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.kplc
.prepare_payment({
            PhoneNumber   : "...",
            Amount        : "...",
            PaymentTypeID : "...",
            MeterNumber   : "...",
            MerchantID    : "KPLC"
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). *.commit_payment({TransactionID,PhoneNumber,Tendered,Pin},headers)*

Acceptable Parameters

Parameter	Type	Required	Description
TransactionID	String	R	Unique transaction reference number
PhoneNumber	String	R	User phone number
Tendered	Int	R	The amount that has been given by the client
Pin	String	R	The initiator's access password/PIN

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.kplc
.commit_payment({
            TransactionID   : "...",
            PhoneNumber     : "...",
            Tendered        : "...",
            Pin             : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

8. OFFSTREET PARKING PAYMENTS (new JP.OffStreetParking(bool liveServer = false))

This endpoint requires initialization before it's use. The main purpose for this is to specify whether or not to use the live endpoint.

To initialize the endpoint use

```
const live_offstreet_parking = new JP.OffStreetParking(true);

or

const test_offstreet_parking = new JP.OffStreetParking();
```

a). *.validate_attendant({PhoneNumber},headers)*

This method helps to check the validity of an attendant by their telephone number

Example:

```
test_offstreet_parking
.validate_attendant({
    PhoneNumber : "..."
})
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). *.book_vehicle({PlateNumber,attPhoneNumber},headers)*

This method helps to book in a vehicle to the system by the plate number

Example:

```
test_offstreet_parking
.book_vehicle({
    PlateNumber : "...",
    attPhoneNumber: "..."
})
.then(d => {

    //@ Do something
```

```
})
.catch(e => {

    //@ Handle the error

})
```

---

9. STARTIMES PAYMENTS (`.startimes`)

---

a). *.prepare_payment({PhoneNumber,Amount,PaymentTypeID,SmartCardCode,Customer*

Acceptable Parameters

Parameter	Type	Required	Description
PhoneNumber	string	R	User phone number
Amount	decimal	R	Transaction amount
PaymentTypeID	Int	R	Mode of payment.e.g. 1 for cash, 2 for cheque, etc
SmartCardCode	Int	R	The client's smartcard number
CustomerCode	Int	R	The designated customer code

The *transactionID* from the response is used in the *.commit_payment* call.

Example:

```
JP
.startimes
.prepare_payment({
            PhoneNumber   : "...",
            Amount        : "...",
            PaymentTypeID : "...",
            SmartcardCode : "...",
            CustomerCode  : "..."
        })
.then(d => {
```

```
    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

---

b). *.commit_payment({TransactionID,PhoneNumber,Tendered},headers)*

Acceptable Parameters

Parameter	Type	Required	Description
TransactionID	String	R	Unique transaction reference number
PhoneNumber	String	R	User phone number
Tendered	Int	R	The amount that has been given by the client

The response parameters will contain a valid *ReceiptNumber* which denotes that the transaction has been successfully commited and completed.

Example:

```
JP
.startimes
.commit_payment({
            TransactionID   : "...",
            PhoneNumber     : "...",
            Tendered        : "..."
        })
.then(d => {

    //@ Do something

})
.catch(e => {

    //@ Handle the error

})
```

GLOBAL METHODS

JP.prepare_payment({},headers)

JP.commit_payment({},headers)

The above methods can be invoked directly for any payment channel (`apart from offstreetparking`) as long as the relevant payment *stream* is passed.

Below is a table of the relevant payment streams

Target	Stream
E-Construction	`econstruction`
House Rent	`rent`
Jambopay Wallet	`wallet`
KASNEB	`kasneb`
Kisumu Water & Sewerage	`kiwasco`
Kenya Power & Lighting	`kplc`
Land Rates	`landrate`
Liquor	`liquor`
Miscellaneous	`misc`
Nairobi Water	`nairobiwater`
NHIF	`nhif`
NCC - OffStreet Parking	`offstreetparking`
Parking	`parking`
Sacco Parking	`saccoparking`
Unified Business Permit	`sbp`

Below is a table of applicable MerchantID's

Target	MerchantID
Nairobi County	`NCC`
Trans Nzoia County	`TNC`

An example sacco_parking prepare_payment request

```
let params = {
        Stream  :"saccoparking"
        SaccoName : "" ,
        Vehicles: [
            {
                vehicleType :"",
                Duration    :"",
                RegistrationNumber:""
            }
        ],
        PaymentTypeID   :""
    };

JP
.prepare_payment(
    params
)
.then(d=>{
    //@ Do something with the response
})
.catch(e => {
    //@ Handle the error
});
```