WebSDK Develop Guide For Merchant Developer (Phase 2)

As an example, this guide uses jQuery framework for frontend and PHP / Java for backend.

Please Note:

Citcon UPI has two kinds of access token, one is client access token, which has inquiry and charge permissions. This is the correct access token for WebSDK / Mobile SDK use.

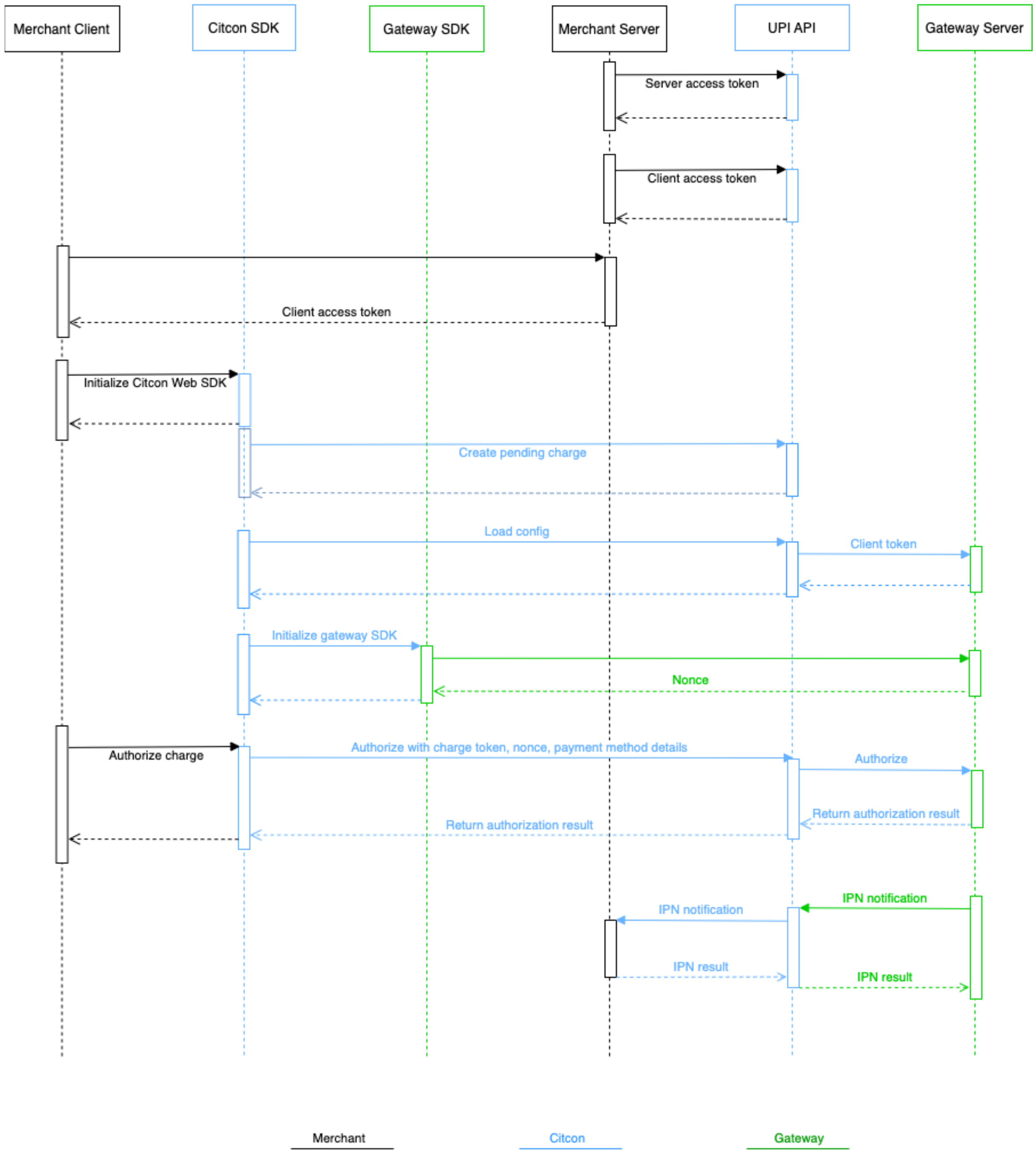
The other one is server access token, which has full permissions: inquiry, charge, refund, capture, vault.

In Citcon WebSDK. Only client access token is needed.

For UPI API Spec, please refer: UPI API Specs

Payment Flow

Payment Initiated from Web SDK



Integration Steps

Step 1: Load Core JS in html page

Load Web SDK core at the bottom of the html page, for example before </body> or after </html>. You can either use the latest version or specify a version

UAT

`<script src="https://cdn.uat01.citconpay.com/latest/core/citconpay.core.js"></script>`

Version base

`<script src="https://cdn.uat01.citconpay.com/v0.2.3/core/citconpay.core.js"></script>`

Production

`<script src="https://cdn.citconpay.com/latest/core/citconpay.core.js"></script>`

Version base

`<script src="https://cdn.citconpay.com/v0.2.3/core/citconpay.core.js"></script>`

For version history, please refer: upi_websdk_demo/history.md at main · long2934/upi_websdk_demo

Step 2: Create access token and pending charge from server back end

Frontend Sample Code

```
  $.ajax({
  url: merchantUrl + '?action=create_transaction',
  type:'post',
  dataType: 'json',
  data: JSON.stringify({
    reference: transaction_reference,
    totalAmount: parseInt($("#txtAmount").val()),
    currency: $("#currency").val(),
    countryCode:$("#country").val()
  }),
  success:function(resp){
    console.log('create pending transaction...' + JSON.stringify(resp));
    if(resp.status === 'success'){
      access_token = resp.data.access_token;
      chargeToken = resp.data.charge_token;
      transactionId = resp.data.transaction_id;
    }else{
      console.log(resp.data);
    }
  },
  async:false
});
```

This only a PHP sample code, you may need to implement by your own

Back End Sample Code

For back end, only thing you need to do is create a http function call Citcon UPI API to create the pending charge, you may need merchant key which you get from Citcon.

UPI API url

Sandbox: `https://api.sandbox.citconpay.com/v1`

Production: `https://api.citconpay.com/v1`

Step one is create an access token from merchant key

PHP code sample

```
function get_access_token(){
    $data = array(
        "token_type" => "client"
    );
    $url = API_URL .'/access-tokens';

    //xxxx is your merchant private key which get from citcon
    $resp = do_http_post_with_token($url,'xxxx',json_encode($data));

    return $resp;
}
```

Java code sample

```
public AccessTokenResponse getAccessToken(){


        String url = Const.getUPIUrl(usingSandbox) + "/access-tokens";
        String data = "{\n  \"token_type\": \"client\"}";
        String resp = callUPIAPI(url,merchantKey,data);
        AccessTokenResponse accessTokenResponse    =  new Gson().fromJson (resp,AccessTokenResponse.class);

        return accessTokenResponse;
    }
```

here the toke type is “client”. there are two kind of token type: client and server. the different is permission. client usually use for WebSDK & MobileSDK

The Response body for this step, will like this sample

```
{
    "status": "success",
    "app": "citcon_upi",
    "version": "v0.1.1",
    "data": {
        "access_token": "UPI_eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiYnJhaW50cmVlIiwiaWF0IjoxNjU4NTc1Nz
        "token_type": "client",
        "expiry": 1658662133487,
        "permission": [
            "inquiry",
            "charge",
            "vault",
            "installment"
        ]
    }
}
```

Step 2 is create the pending charge from the access token

PHP code sample

```
function create_pending_transaction($token){

    $json = file_get_contents('php://input');
    $json_data = json_decode($json,true);
    $urls = array (
        "ipn_url" => "https://dev.citconpay.com",
        "success_url" => "https://dev.citconpay.com",
        "fail_url" =>  "https://dev.citconpay.com",
        "mobile_url" => "https://dev.citconpay.com"
    );
    $transaction = array(
            "reference" => $json_data['reference'],
            "amount"=> $json_data['totalAmount'],
            "currency"=> $json_data['currency'],
            "auto_capture"=> false,
            "country"=> $json_data['countryCode'],
    );
    $data_array['transaction'] = $transaction;
    $data_array['urls'] = $urls;
    $data = json_encode($data_array);
    $url = API_URL .'/charges';
    $resp = do_http_post_with_token($url,$token,$data);
    return $resp;
}
```

Jave code sample

```
    public PendingChargeData getPendingCharge(String reference,String amount,String currency,String countryCode,
        String data = "{";
        data = data +"\"transaction\": {";
        data = data +"\"reference\": \""+reference+"\",";
        data = data +"\"amount\": "+ amount+",";
        data = data +"\"currency\": \""+currency+"\",";
        data = data +"\"country\": \"" + countryCode +"\",";
        data = data +"\"auto_capture\": false";
        data = data +"},";
        data = data +"\"urls\": {";
        data = data +"\"ipn\": \"http://ipn.com\",";
        data = data +"\"success\": \"http://success.com\",";
        data = data +"\"fail\": \"http://fail.com\",";
        data = data +"\"mobile\": \"http://mobile.com\",";
        data = data +"\"cancel\": \"http://cancel.com\"";
        data = data +"}";
        data = data +"}";
        System.out.println(data);
        String url = Const.getUPIUrl(usingSandbox) + "/charges";
        String resp = callUPIAPI(url,token,data);
        PendingChargeRespone pendingChargeRespone = new Gson().fromJson(resp, PendingChargeRespone.class);
        return pendingChargeRespone.getData();
    }
```

here the IPN (Instant Payment Notify) url is the url which you want to get the result from Citcon UPI.

For the IPN Verify Signature, please refer this : How to verify UPI IPN signature

This Step response body will looks like

```
{
   "status":"success",
   "data":{
      "charge_token":"5412428009ca11edbeec5dd5c069f366",
      "transaction_id":"4000033920223962677253",
      "access_token":"UPI_eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiYnJhaW50cmVlIiwiaWF0IjoxNjU4NTAwMDEwLC
   }
}
```

for fully demo code, please download from here GitHub - long2934/upi_websdk_demo: Demo for UPI WebSDK

Step 3: Initialize WebSDK Core with access token

Add a element in the HTML file to host WebSDK widgets, such as

`<div id="citcon-client-container"></div>`

Web SDK then can mount a widget.

Sample code

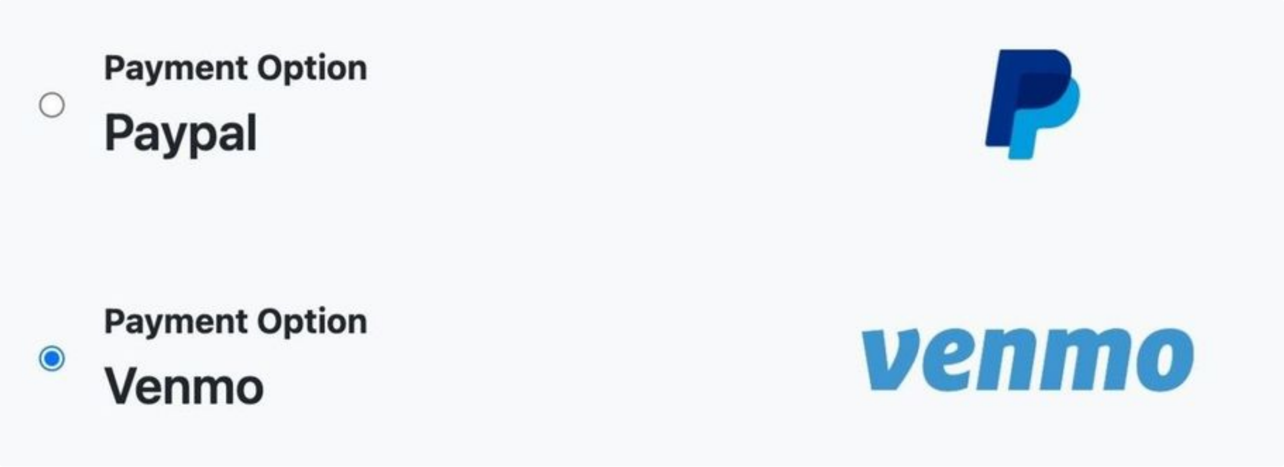
```
// the payment method is what merchant choosed at onboarding.
let paymentMethodArray =['paypal','venmo','card'];
let defaultPaymentMethod = 'paypal';
//Citcon Dropin & Components
let citconDropins = ['NameOnCard','Card','Address','Municipal','Phone','Email'];

const configObj = {
  accessToken: access_token,
  environment:'uat', ///uat/prod,This is optional. default is production
  debug: true, // true/false, This is optional. default is false
  consumerID: "18000",
  languages: "en",//en for English,zh_CN for Mandarin,fr for French,es for Spanish,this is optional, default i
  cardTypes:['VISA','AXP','MA','JCB','DFS'], //For allowed credit card
  threeDSPaymentMethodScope:['debitcard'], // debitcard,creditcard, For Ebanx use only
  availableCountries:['BR'], //This only used for Municipal components
  urls: {
    ipn: "https://api.huiuh.com/notify0",
    mobile: "http://mobile.com",
    success: "https://baidu.com",
    cancel: "https://jd.com",
    fail: "https://qq.com"
  }
};

citconpay.client.core(configObj).then( clientInstance=>{
  console.log(' Init SDK...' + JSON.stringify(clientInstance));
  // mount UI
  // class name is css class, you can define your own class at your style sheet
  citconInstance = clientInstance;
  clientInstance.mount('#citcon-client-container',{
    classname: 'payment-method-select-component',
    paymentMethods: paymentMethodArray,
    selectedPaymentMethod: defaultPaymentMethod,
    citconDropinFormDetail: citconDropins,
  }, sdkUIDidInitialized).then(function(instance) {

      console.log('instance ...' + JSON.stringify(instance));
      //Do Events Register after WebSDK core init complete
      registerEvents();
    }).catch(error=>{
      console.log(' mount error:' + JSON.stringify(error));
    });
  }).catch(error=>{
    console.log(' Init SDK error:' + JSON.stringify(error));
  });

function sdkUIDidInitialized(e){

  console.log(' sdk ui initialized .....' + JSON.stringify(e));
}
```

Step 4: Register event listeners

Supported events include the following:

`1.payment-method-selected`

This event is triggered when consumer clicks on the Radio Button



`2.payment-method-submitted`

This event is triggered when consumer clicks the “Pay Now” button

`3.payment-status-changed`

This event is triggered when payment is completed. It calls back the listener with payment result.

Sample Return Object in JSON:

Success:

```

{
   "status":"success",
   "data":{
         "object":"charge",
         "id":"da949a90cb1011ecb9a3550ac676a90f",
         "reference":"0wqd3i0s5p8483q62f5x",
         "amount":180,
         "amount_captured":null,
         "amount_refunded":null,
         "currency":"USD",
         "time_created":1651603379000,
         "time_captured":null,
         "auto_capture":false,
         "status":"authorized",
         "country":"US",
         "payment":{
            "method":"paypal"
         }
         ...
    }
}

```

fail:

```
{
   "status":"fail",
   "data":{
      "code":xxxx,
      "message":"xxxxx",
   }
}


```

Sample Code:

```
function registerEvents(){

  citconInstance.on('payment-method-selected', function (e) {
    console.log('Inside `payment-method-select` .........' + JSON.stringify(e));
    //Do you code here
  });

  citconInstance.on('payment-method-submitted', function(e) {
    // UI Event
    // For those payment method Popup Or Redirect, Sucha as Paypal, Venmo
    // merchant handles selection of payment methods through Citcon's UI
    console.log('....paynow..click.....');
    $("body").addClass("loading");
    const requestOptions ={
          payment: {
            chargeToken:chargeToken,
            countryCode:$("#country").val(),
            transactionReference: transaction_reference
          },
          billingAddress: {
             street: $("#address").val(),
             street2:$("#address2").val(),
             city: $("#txtCity").val(),
             state: $("#state").val(),
             zip: $("#zip").val(),
             country: $("#country").val()
          },
          consumer:{
            id:"18000",
            reference: "consumer_test_1",
            firstName: $("#firstName").val(),
            lastName: $("#lastName").val(),
            phone: $("#phone").val(),
            email: $("#email").val(),
          }
      }
      citconInstance.onPaymentMethodSubmitted(e.paymentMethod,requestOptions).then(rest=>{
          console.log('pay now click, return..' + JSON.stringify(rest));
        }).catch(error=>{
          $("body").removeClass("loading");
          console.log('pay now click, error..' + JSON.stringify(error));
        });
  });

  //on payment status change, this is the event for charge result
  //status: status, status will return "success" or "failed"
  //retObj: retObj
  citconInstance.on('payment-status-changed', function(e) {
     const status = e.status;
     const res = e.retObj;

     console.log('payment-status-change status..' + status + JSON.stringify(res));
     if(status == 'success'){
       //payment success
       // do your code here....
     }else{
        $("body").removeClass("loading");
        if (res.code === 'VENMO_CANCELED') {
          console.log('App is not available or user aborted payment flow');
        } else if (res.code === 'VENMO_APP_CANCELED') {
          console.log('User canceled payment flow');
        } else {
          console.log('An error occurred:', res.message);
        }
     }
  });
}

```

`4.vault-item-selected`

This is when customer choose one of the vault item

Sample code:

```
citconInstance.on('vault-item-selected', function(e) {
  // vault item selected, do anything you want for this event
  console.log('....vault item selected.......' + JSON.stringify(e));
});
```

Additional functions

Charge inquiry

You can use this function to inquire charge status

Sample code

```
$("body").addClass("loading");
console.log('inquire order... id=' + transactionId);
citconInstance.inquire(transactionId).then(resp=>{
  console.log('inquire...return' + JSON.stringify(resp));
  $("body").removeClass("loading");
}).catch(error=>{
  console.log('inquire order error...' + JSON.stringify(error));
  $("body").removeClass("loading");
});
```

Parameters

Init SDK

Sample code:

```
let paymentMethodArray =['paypal','venmo','card'];
//Citcon Dropin & Components
let citconDropins = ['NameOnCard','Card','Address','Municipal','Phone','Email'];

const configObj = {
  accessToken: access_token,
  environment: 'prod', //dev/qa/uat/prod,
  debug:true,
  consumerID:"18000",
  languages: "en",//en for English,zh_CN for Mandarin,fr for French,es for Spanish,this is optional, default i
  cardTypes:['VISA','AXP','MA','JCB','DFS'], //For allowed credit card
  threeDSPaymentMethodScope:['debitcard'], // debitcard,creditcard, For Ebanx use only
  availableCountries:['BR'], //This only used for Municipal components
  urls: {
    ipn: "https://api.huiuh.com/notify0",
    mobile: "http://mobile.com",
    success: "https://baidu.com",
    cancel: "https://jd.com",
    fail: "https://qq.com"
  }
};

```

```
console.log(' citconpay...' + JSON.stringify(citconpay));

citconpay.client.core(configObj).then( clientInstance=>{
```

Payment Method:

Phase 2 support the following payment method:

Braintree:

`'vault','paypal','venmo','card'`

Ebanx:

`'vault','card','oxxo','oxxopay','spei','mercadopago'`

TOSS:

`'card', 'banktransfer','toss','lpay','lgpay','samsungpay'`

PPCP

`'paypal','venmo'`

FOMO

`'grabpay','shopeepay','alipay','upop','wechatpay','atome','paynow','netspay','paypal'`

Xendit

`'vault','card','gcash','grabpay','paymaya','shopeepay','bpi','ubp','billease','cashalo','7eleven','ecpay','cebuana','mlhuillier','palawan','ecpayloan','lbc','rdpawnshop','cvm'`

Dropin and Component

Component include the followings

`'FirstName','LastName','Phone','Email','Address','Municipal','NameOnCard','Card','AllInOne','DocumentID'`

Parameters

Parameter	Required	Description
`accessToken`	M	This is the access token which create by backend code at step 2, use to communication with UPI API
`environment`	O	This is a enum , values will be `uat/prod`, default is prod
`debug`	O	This is whether show log . suggest set to true during developing. and remove it when live. default is false
`consumerID`	M	This is the consumer ID which store in Merchant’s database
`languages`	O	This is a enum, values will be `'auto',"en-US", "zh-CN",'ko-KR','fr-CA','es-MX','pt-BR'`, `'ja-JP','tl-AU'`,default is auto, detect by browser, if browser language not in above values , WebSDK will use English ,for fully language code, please see Langauges Code section

`cardTypes`	O	This is a enum, values will be `['VISA','AXP','MA','JCB','DFS']`, default is all the credit card.
		For Ebanx Brazil, this value can be
		`['VISA','AXP','MA','Diners','Hiper']`
		For Ebanx Mexico, this value can be
		`['VISA','AXP','MA','Banamex','Banorte','Bbva','HSBC','Santander','CARNET']`

`threeDSPaymentMethodScope`	O	This is a enum, values will be `debitcard,creditcard, this is only for Ebanx Brazil 3DS validation`
`availableCountries`	O	`This is a country iso2 array, such as ['US','CA']. only used for Municipal components`

`countryCode`	C	Country Code, This is ISO2, such as US, CA, This is for Ebanx Credit Card
`urls`	C	This is the urls, For some Payment Gateway use, such as TOSS, PPCP, FOMO

Payment Method Selected

This is for event `payment-method-selected`, that means customer choose this payment method, and this payment method need some parameter to initialize.

Sample code:

```
citconInstance.on('payment-method-selected', function (e) {
 console.log('Inside `payment-method-selected` .........' + JSON.stringify(e));
 let selectedPaymentMethod = e.paymentMethod;
 //when customer select card, show the card input UI
 let options ={
   consumer:{
    id:"115646448"
   },
   payment: {
    totalAmount: parseInt( $("#txtAmount").val()),
    currency:$("#currency").val(),
    countryCode:$("#country").val(),
    transactionReference: transaction_reference,
    chargeToken:chargeToken
   },
   //whether need 3DS
   request3DSecureVerification: true,
   requestInstallment: true,
  }
  citconInstance.onPaymentMethodSelected(e.paymentMethod,options).then((rest)=>{
   console.log('onPaymentMethodSelected , result' + JSON.stringify(rest));
  }).catch(error=>{
   console.log('onPaymentMethodSelected error:' + JSON.stringify(error));
  });
});
```

Parameters the same as `payment-method-submitted`. see following

Submit Payment

This is for event `payment-method-submitted`.

Sample Code will be

```
citconInstance.on('payment-method-submitted', function(e) {
 console.log('....paynow..click.....');
 $("body").addClass("loading");
 const paypalOptions ={
  payment: {
   totalAmount: parseInt( $("#txtAmount").val()),
   currency:$("#currency").val(),
   countryCode:$("#country").val(),
   transactionReference: transaction_reference,
   chargeToken:chargeToken,
   autoCapture:true,
  },
  billingAddress: {
   street: $("#address").val(),
   street2:$("#address2").val(),
   city: $("#txtCity").val(),
   state: $("#state").val(),
   zip: $("#zip").val(),
   country: $("#country").val()
  },
  consumer:{
   id:"18000",
   reference: "consumer_test_1",
   firstName: $("#firstName").val(),
   lastName: $("#lastName").val(),
   phone: $("#phone").val(),
   email: $("#email").val(),
  },
  tax:{
   taxExemptAmount:20
  },
  request3DSecureVerification:true,
  liabilityShiftPossibleContinue:false,
  bothFailedContinue:false
 }
 switch(e.paymentMethod){
  case 'venmo':
  case 'paypal':
   citconInstance.onPaymentMethodSubmitted(e.paymentMethod,paypalOptions).then(rest=>{
     console.log('pay now click, return..' + JSON.stringify(rest));
   }).catch(error=>{
     $("body").removeClass("loading");
     console.log('pay now click, error..' + JSON.stringify(error));
   });
   break;
 }

});
```

Parameters

Parameter	UPI Parameter *	MobileSDK Parameter (Android)	Required	Description
payment.totalAmount	payment.amount	UpiOrderBuilder.amount	M	Total amount of the order, In Cents, such as 100 is $1.00
payment.currency		UpiOrderBuilder.currency	M	Currency Code, such as USD, CAD,…
payment.countryCode	transaction.country	UpiOrderBuilder.country(Locale)	M	Country Code, This is ISO2, such as US, CA
payment.`transactionId`	N/A		C	UPI transaction ID. （For PPCP）
payment.transactionReference	transaction.reference	UpiOrderBuilder.reference	M	This is the transaction reference
payment.chargeToken	N/A		M	This is the charge token which create by backend at step 2
payment.`autoCapture`	transaction.auto_capture	this parameter is set in merchant backend when create charge token	O	whether auto capture (paid) or not
payment.`vertical`	transaction.vertical		O	(For PPCP)

payment.billingAddress.street	payment.billing_address.street	`BillingAddressBuilder.street`	O	Billing address street address
payment.billingAddress.street2	payment.billing_address.street2	`BillingAddressBuilder.street` 2	O	Billing address street address2, like Apartment number,..
payment.billingAddress.city	payment.billing_address.city	`BillingAddressBuilder.city`	O	Billing address city name
payment.billingAddress.state	payment.billing_address.state	`BillingAddressBuilder.state`	O	Billing address state or province name
payment.billingAddress.zip	payment.billing_address.zip	`BillingAddressBuilder.zip`	O	Billing address zip or post code
payment.billingAddress.country	payment.billing_address.country	`BillingAddressBuilder.country`	O	Billing address country code, this also ISO2, ex. US.
`consumer` .id	N/A		C	This is the consumer ID which store in Merchant's database, the same as Init SDK. It is mandatory for Braintree
`consumer` . `reference`			M	This is the consumer reference
`consumer` . `firstName`		`ConsumerBilder.firstName`	O	This is the consumer first name
`consumer` . `lastName`		`ConsumerBilder.lastName`	O	This is the consumer last name
`consumer` . `phone`		`ConsumerBilder.phone`	O	This is the consumer phone number
`consumer` . `email`		`ConsumerBilder.email`	O	This is the consumer email address
`consumer.registrationTime`			O	Timestamp in seconds (For PPCP)
`consumer.registrationIp`			O	(For PPCP)
`consumer.riskLevel`			O	Possible value: `high` , `medium` and `low` (For PPCP)
`consumer.firstInteractionTime`			O	Timestamp in seconds (For PPCP)
`consumer.totalTransactionCount`			O	(For PPCP)
`tax` . `taxExemptAmount`	goods.data[index].tax_exempt_amount		O	For TOSS use
`request3DSecureVerification`	N/A		O	For Braintree Card and Ebanx Brazil Card, whether need 3DS verification
`requestInstallment`	N/A		O	For Ebanx use, whether need instalment function. For TOSS, when total amount bigger or equal 10000, the instalment button will auto show
`liabilityShiftPossibleContinue`	N/A		O	This two parameter only for Braintree card 3DS For liabilityShiftPossible & liabilityShifted, please refer to https://developer.paypal.com/braintree/docs/guides/3d-secure/legacy-3d-secure/client-side liabilityShiftPossibleContinue means when Braintree 3DS return liabilityShiftPossible = false whether continue bothFailedContinue means Braintree 3DS return liabilityShiftPossible = false AND liabilityShifted = false whether continue ,default is false, if you set to true, means you will accept the risk
`bothFailedContinue`	N/A		O	
goods.data			C	goods.data is a json array. will including the following items
goods.data[index].productType			C	value: `physical` , `digital` , `donation` (For PPCP)
goods.data[index].name			C	(For PPCP)
goods.data[index].sku			O	(For PPCP)
goods.data[index].url			O	(For PPCP)
goods.data[index].quantity			C	(For PPCP)
goods.data[index].unitAmount			C	(For PPCP)
goods.data[index].unitTaxAmount			O	(For PPCP)
goods.data[index].totalAmount			O	(For PPCP)
goods.data[index].totalTaxRate			O	(For PPCP)
goods.data[index].totalTaxAmount			O	(For PPCP)
goods.data[index].totalDiscountAmount			O	(For PPCP)
goods.data[index].totalDiscountCode			O	(For PPCP)
goods.data[index].taxableAmount			O	(For PPCP)

goods.data[index].taxExemptAmount			O	(For PPCP)
goods.shipping.country			C	(For PPCP)
goods.shipping.state			O	(For PPCP)
goods.shipping.city			C	(For PPCP)
goods.shipping.zip			C	(For PPCP)
goods.shipping.type			O	value: `shipping` , `pickup_in_persion` (For PPCP)
goods.shipping.firstName			O	(For PPCP)
goods.shipping.lastName			O	(For PPCP)
goods.shipping.phone			O	(For PPCP)
goods.shipping.email			O	(For PPCP)
goods.shipping.street			O	(For PPCP)
goods.shipping.street2			O	(For PPCP)
goods.shipping.amount			O	(For PPCP)

*For *UPI Parameter* column: blank means the parameter is consistent with WebSDK parameter, red color means the parameter is inconsistent with WebSDK parameter

the goods sample will be

```
"goods": {
        "shipping": {
            "firstName": "fist",
            "lastName": "last",
            "phone": "string",
            "email": "test@citcon.cn",
            "street": "3 Main St",
            "street2": "",
            "city": "CA",
            "state": "San Jose",
            "zip": "95132",
            "country": "US"
        },
        "data": [{
            "name": "shoes",
            "sku": "shoes",
            "url": "string",
            "quantity": 1,
            "unitAmount": 15,
            "unitTaxAmount": 10,
            "totalDiscountAmount": 10,
            "category" : "PHYSICAL_GOODS"
        }]
}
```

Langauges Code

WebSDK Phase 2 support the following languages

Languages Code (ISO 639-1)	Language
`auto`	Detect by Browser, if not support , will use English
`en-US`	English
`zh-CN`	Simple Chinese

`ko-KR`	Korean
`fr-CA`	French Canada
`es-MX`	Spanish for Mexico
`pt-BR`	Portuguese for Brazil
ja-JP	Japanese
tl-PH	Tagalog

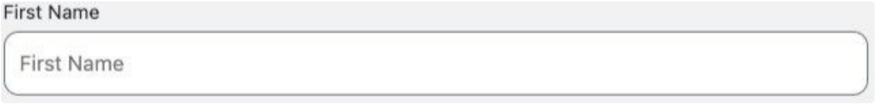
Component and Drop-ins

Component

There are following components:

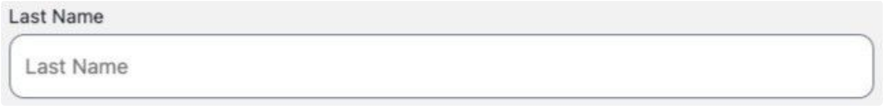
`FirstName`

This is Consumer’s first name, using for UPI `consumer` object, it will show a component like this



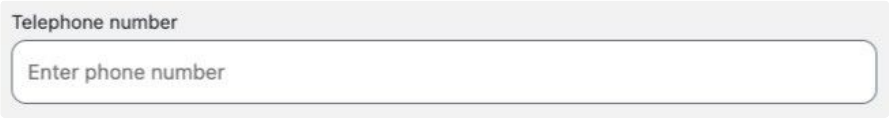
`LastName`

This is Consumer’s last name, using for UPI `consumer` object, it will show a component like this



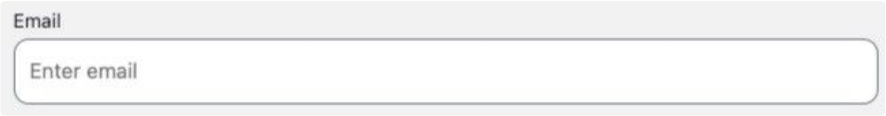
`Phone`

This is Consumer’s contact phone number, using for UPI `consumer` object, it will show a component like this



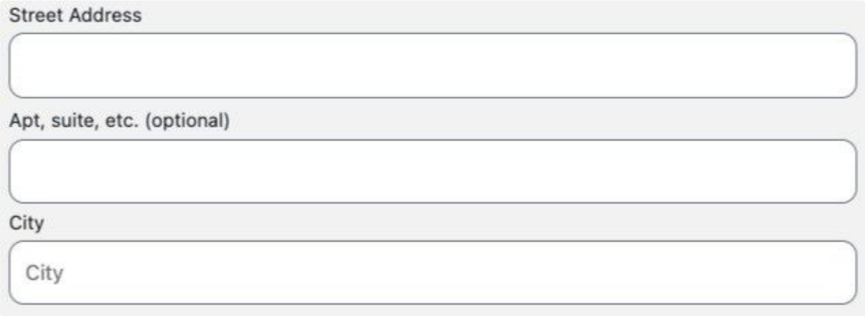
`Email`

This is Consumer’s contact e-mail address, using for UPI `consumer` object, it will show a component like this



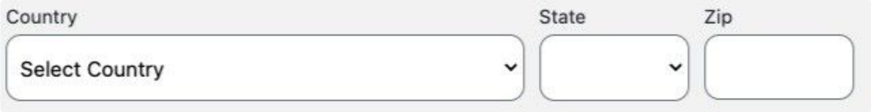
`Address`

This is Consumer’s address, using for billing address , including street address , street2, city. It will show a component like this



Municipal

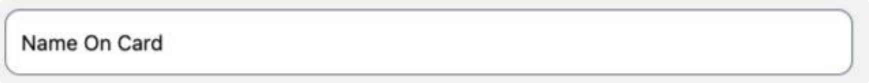
This is Consumer’s address, using for billing address , including Country, State / Province and Zip / Post Code. It will show a component like this



NameOnCard

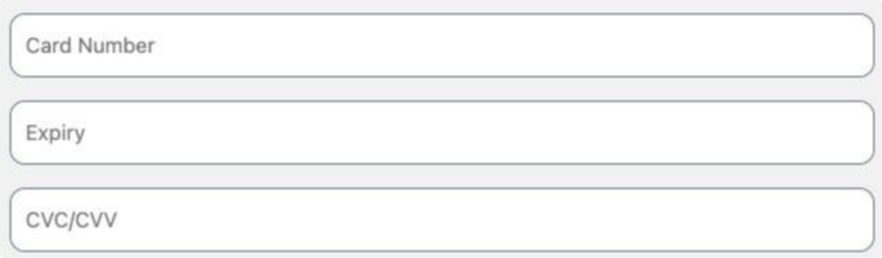
This is Name on the credit card, including First name and Last name, using for card information.

It will show a component like this



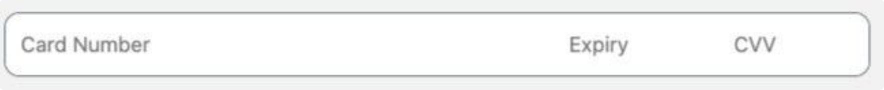
Card

This is traditional credit card widget, has card number, exp date and cvv, It will show a component like this



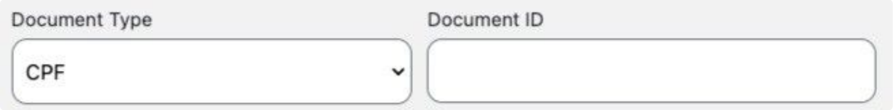
AllInOne

This is tiny credit card widget, it will combine 3 elements together. It will show a component like this



`DocumentID`

This is only for Ebanx Brazil use . It will show a component like this



Document Type: CPF

Document ID:

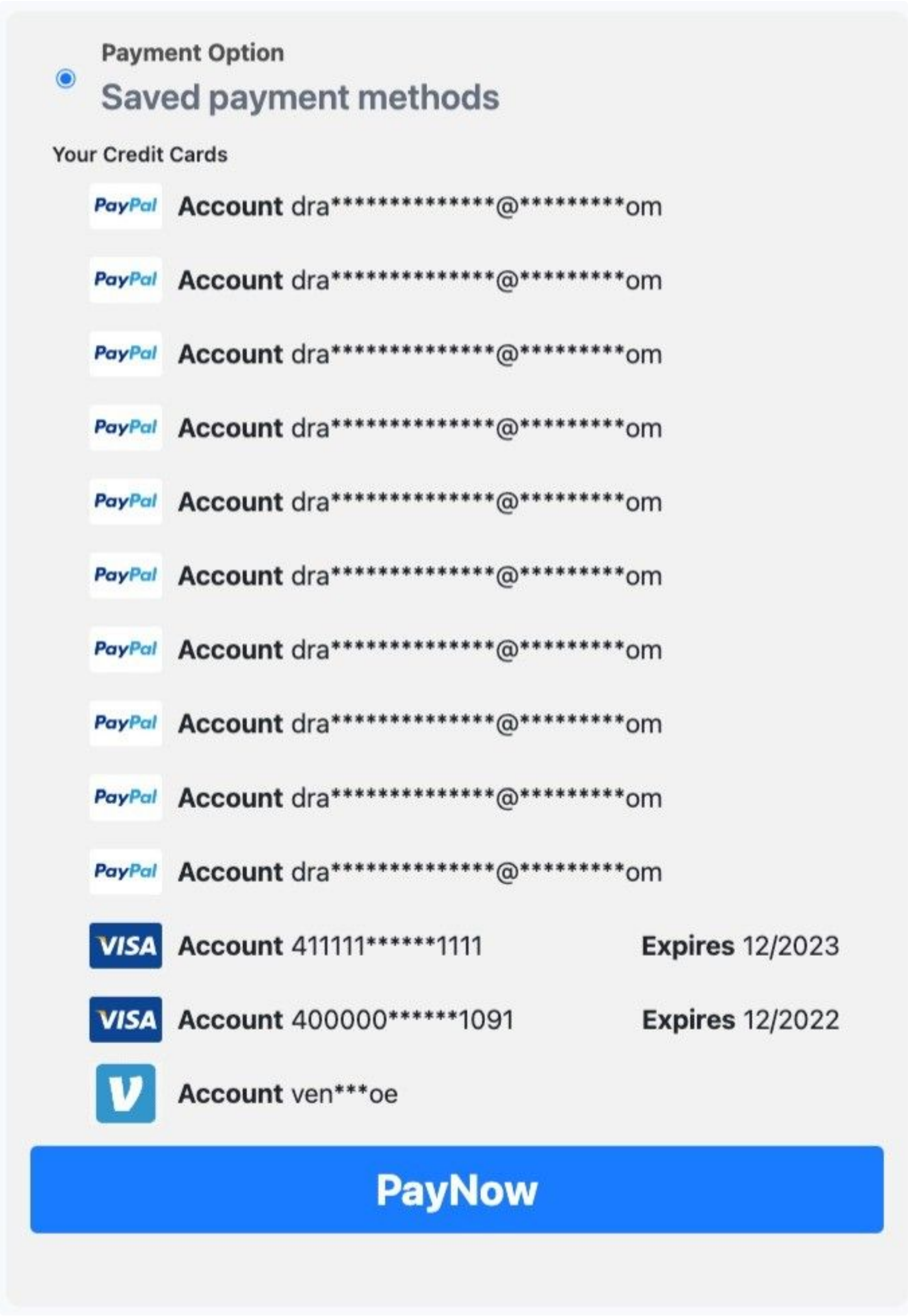
Drop-ins

There are following drop-ins

Vault

Vault is saved payment method. which will let consumer easy to pay, not need enter credit card information or login to PayPal etc. Vault has two widgets. one is “Saved payment methods”

like this



○ Payment Option

Saved payment methods

Your Credit Cards

PayPal Account dra**************@*********om

PayPal Account dra**************@*********om

PayPal Account dra**************@*********om

PayPal Account dra**************@*********om

PayPal Account dra**************@*********om

PayPal Account dra**************@*********om

PayPal Account dra**************@*********om

PayPal Account dra**************@*********om

PayPal Account dra**************@*********om

PayPal Account dra**************@*********om

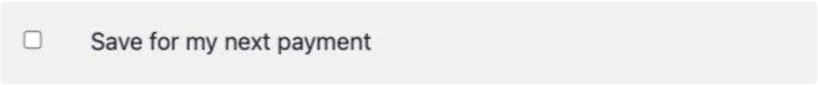
VISA Account 411111******1111 Expires 12/2023

VISA Account 400000******1091 Expires 12/2022

Account ven***oe

another is a checkbox show under the payment method.

like this



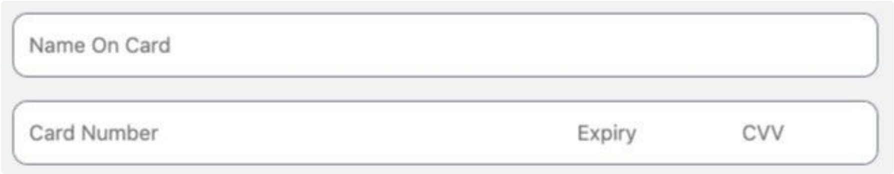
□ Save for my next payment

this two widgets will show if `paymentMethodArray` add ‘vault’ , also the payment method support vault

TinyCard

This drop-in include NameOnCard and AllInOne two components

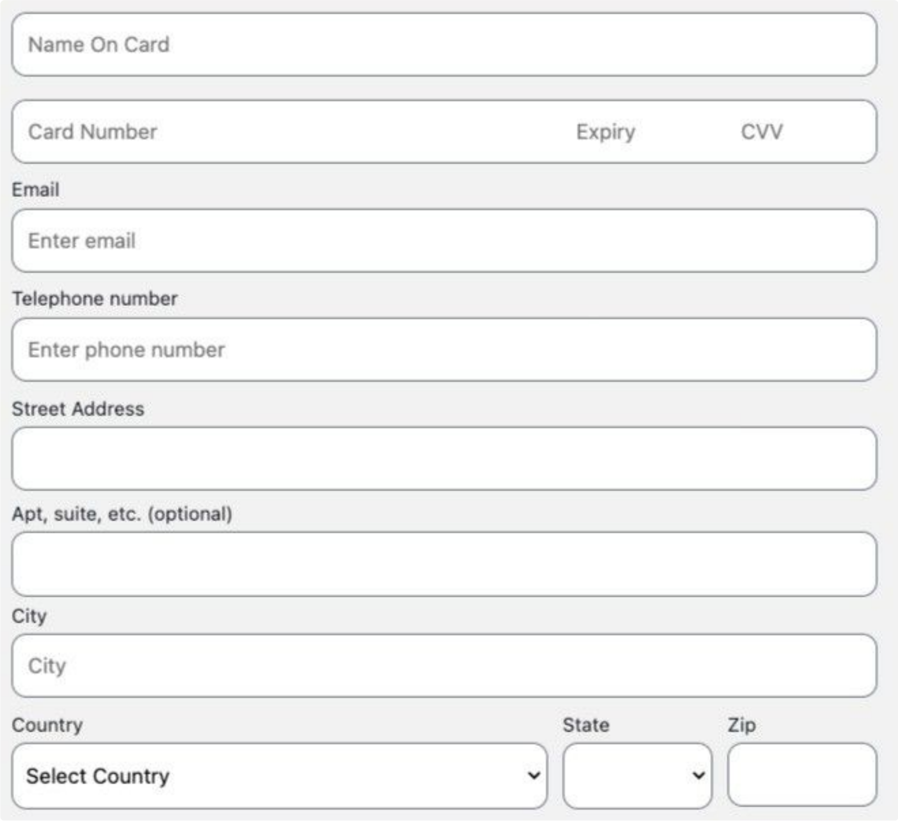
the drop-in looks like this



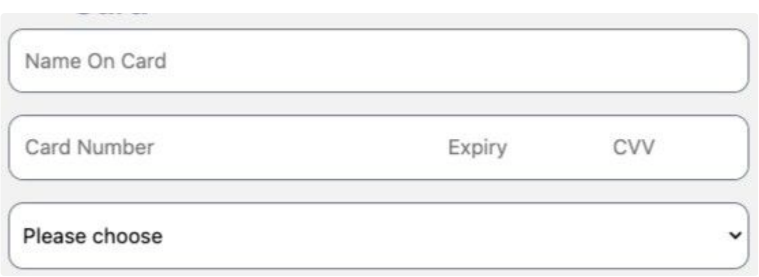
CardWithBillingAddress

This drop-in include NameOnCard , AllInOne , email, phone, address and Municipal six components

the drop-in looks like this



CardWithInstallments



Resources

1.Demo code repositories

please checkout at here GitHub - long2934/upi_websdk_demo: Demo for UPI WebSDK

2. Ebanx 3DS testing card

please checkout at here: Cards integration with 3DS 2 through EBANX Direct API | EBANX Developers Academy | EBANX Developers Academy

3. Braintree 3DS testing card

please checkout at here: https://developer.paypal.com/braintree/docs/guides/3d-secure/testing-go-live/php

That’s everything, happy coding!