Payment Service - System Design

CyberSource Integration



October 11th, 2024

Submitted to:

Icon

Description automatically generated

Prepared by TransSIGHT

**DOCUMENT VERSION CONTROL**

|  |  |  |
| --- | --- | --- |
| **DOCUMENT NAME** | **SUBMITTAL DATE** | **VERSION NO.** |
| **Payment Service – System Design -- CyberSource Integration**  - First Draft | 10/11/2024 | 1.0 |

**TABLE OF CONTENTS**

[1.0 Overview 2](#_Toc179386753)

[2.0 Payment Service Actions 3](#_Toc179386754)

[Generate Client Token 4](#_Toc179386755)

[Create Customer 6](#_Toc179386756)

[Add Payment Method 9](#_Toc179386757)

[Delete Payment Method 11](#_Toc179386758)

[Find Payment Method 12](#_Toc179386759)

[Sale Transaction 13](#_Toc179386760)

[Hold Authorization Transaction 15](#_Toc179386761)

[Void Transaction 17](#_Toc179386762)

[Refund Transaction 18](#_Toc179386763)

[Transaction Search 19](#_Toc179386764)

[Find Expiring and Expired Credit Cards 21](#_Toc179386765)

[3.0 Microform User Interface 22](#_Toc179386766)

[4.0 Payments Schema 24](#_Toc179386767)

[Merchant Accounts 24](#_Toc179386768)

[Payment Methods 24](#_Toc179386769)

[5.0 Testing 25](#_Toc179386770)

[Dave’s Sandbox Account 25](#_Toc179386771)

[Test Credit Card Numbers 25](#_Toc179386772)

[Refund or Cancellation 25](#_Toc179386773)

[MicroForm Process Flow 26](#_Toc179386774)

[6.0 Developer Notes 30](#_Toc179386775)

[Changes in latest version of cybersource-rest-client 30](#_Toc179386776)

[7.0 References 31](#_Toc179386777)

[Sequence Diagram for Tokenization and Payment Processing 31](#_Toc179386778)

[8.0 Settlements 32](#_Toc179386779)

[Braintree 32](#_Toc179386780)

[Cybersource 32](#_Toc179386781)

[9.0 Issues and Questions 33](#_Toc179386782)

[Billing information required when vaulting each payment method 33](#_Toc179386783)

[10.0 Summary 34](#_Toc179386784)

# Overview

The objective of this document is to outline the technical design for integrating CyberSource as a payment gateway while ensuring continuity with the existing functionalities implemented using Braintree in both mobile applications and website.

**TODO:** Cybersource does not support saved payment methods for PayPal or Venmo, so will need to include a section on notifying customers with these active payment methods.

**TODO:** Need a configuration section and how the payment service will identify between Braintree and Cybersource requests.

**SEE:** <https://github.com/evadremlab/cybersource-demo> for my sample code used to test various features.

# Payment Service Actions

These are the actions currently supported by the payment service.

|  |  |  |
| --- | --- | --- |
| **Action** | **Braintree Command** | **Comments** |
| Generate Client Token | gateway.clientToken.generate | Get client token for drop-in UI |
| Create Customer | gateway.customer.create | Called when adding payment method if users.payment\_processor\_customer\_id is null |
| Add Payment Method | gateway.paymentMethod.create |  |
| Delete Payment Method | gateway.paymentMethod.delete |  |
| Find Payment Method | gateway.paymentMethod.find | Used to validate delete payment method request |
| Sale Transaction and  Hold Authorization Transaction | gateway.transaction.sale |  |
| Refund Transaction | gateway.transaction.refund |  |
| Void Transaction | gateway.transaction.void |  |
| Transaction Search | gateway.transaction.search | Used by settlement process |
| Find Expired and Expiring Credit Cards | gateway.creditCard.expiringBetween | Used by Parking backend scheduled task notify customers of expiring or expired credit cards |

## Generate Client Token

The Flex Microform API will be used to generate the context of the customer payment information that is to be captured and tokenized. The capture context request contains all the merchant-specific parameters that tell the front-end JavaScript library how to behave within your payment experience.

The capture context response is a signed JSON Web Token (JWT) containing this information:

* Merchant-specific parameters: Manages the customer payment experience for the current payment transaction.
* A one-time public key: Secures the information flow during the current payment transaction.

The capture context is signed so that the receiving party can verify it is coming from CyberSource, including automated verification by Microform.

Example request:

|  |
| --- |
| const path = require('path');  const cyberSourceConfig = require(path.resolve('config/cybersource-config.js'));  const {  ApiClient,  GenerateCaptureContextRequest  } = require('cybersource-rest-client');  const requestData = GenerateCaptureContextRequest.constructFromObject({  clientVersion: 'v2.0',  targetOrigins: ['http://localhost:3000'], **🡨** url hosting the microform  allowedCardNetworks: ['VISA', 'MASTERCARD', 'AMEX', 'DISCOVER']  });  instance.generateCaptureContext(requestData, (err, data, response) => {  // handle response here  } |

Example decoded capture context from response:

|  |
| --- |
| {  "flx": {  "path": "/flex/v2/tokens",  "data": "OnNUTif50p3N6pgB51VBZBAAEKd2sZ/Vi4swG2FjLCDuNf4VpoHp47Yfede94N3yR/ehab0RHEIS5cVp/m80Otc40J+mLQXecnRxPBue9n6F3ZN6ptXRmWnX53FtWPs0TP3ClkEbte7nZdGIu8LmUaiLEQ==",  "origin": "https://testflex.cybersource.com",  "jwk": {  "kty": "RSA",  "e": "AQAB",  "use": "enc",  "n": "3fqFjmYzm-48wEtYL4lBJKDPLJGCgBJ3CBygBqqZHcXylcC7vWiUBaLINgqpVUsE8QIuPjlkntB-RHiI8J-x7vIdHNRfBe6FEjxVMCP7N9jK49\_r6m6XB1jJGdNFYiAmVqGgbuj3MLcQH701wF\_hQXCBwuxZbCbWLgdSG7eZcdOQm9Xr4ahYhZVITD0OMjn\_qWrsCWlpXt9lICpVS4cy7VatQaTPXCSFVbLa69-X2z13O6kxjR02g26\_4POMoYaGoQUsTHCimLtIeosRFGN5XZRjsEr6f9bivaez8soNvoZNThQPYmvhS89CuA8\_hqFgQ8frVTVDOduHrbkW\_JnGTw",  "kid": "089U7tM9jWI0Lcbm5IiQPmS0fDwKmGzV" **🡨** decoded to get public key for verification  }  },  "ctx": [  {  "data": {  "clientLibrary": "https://testflex.cybersource.com/microform/bundle/v2.0/flex-microform.min.js", **🡨** use instead of hardcoded path  "allowedCardNetworks": [  "VISA",  "MASTERCARD",  "AMEX",  "DISCOVER"  ],  "targetOrigins": [  "http://localhost:3000"  ],  "mfOrigin": "https://testflex.cybersource.com"  },  "type": "mf-2.0.0"  }  ],  "iss": "Flex API",  "exp": 1727991541,  "iat": 1727990641,  "jti": "gLeb73YVSfrkJs9g"  } |

## Create Customer

We know whether a user has a Braintree account from the payment\_processor\_customer\_id in the “payments”.users record and this column is defined as varchar(50), so no change is required to use it to store the CyberSource customer token value eg “23AC2720514EA950E063AF598E0AF5C1”.

*“In CyberSource it is recommended that a customer be created with a zero amount Payment Authorization.”*

*“Cybersource can automatically verify that a payment card or electronic check account is valid prior to tokenization by authorizing a zero or low value amount, depending on the card type. There is no additional charge from Cybersource for this service.”*

A CyberSource customer and payment method will be created at the same time when adding a user’s first payment method. This requires a transient token (JWT) generated by the Flex API from the Microform used to enter the payment information. See the [Microform User Interface](#_Microform_User_Interface) sections for details on creating the transient token.

The payment service currently receives a “nonce” generated from the Braintree drop-in UI, and we would use this same parameter to receive the transient token from the Cybersource UI in the website or app.

Example request:

|  |
| --- |
| {  "tokenInformation": {  "transientTokenJwt": "<from microform.createToken()>",  "paymentInstrument": {  "default": true 🡨 required to be true for first payment method  }  },  "processingInformation": {  "commerceIndicator": "internet",  "actionList": [  "TOKEN\_CREATE" **🡨** create the following token types  ],  "actionTokenTypes": [ **🡨** customer and payment method at the same time  "customer",  "paymentInstrument"  ],  "capture": false **🡨** auth only  },  "orderInformation": {  "amountDetails": {  "totalAmount": "0.00",  "currency": "USD"  },  "billTo": { **🡨** all fields required  "email": "david.balmer@transsight.com",  "locality": "Alameda",  "firstName": "David",  "lastName": "Balmer",  "address1": "100 Main Street",  "country": "US",  "administrativeArea": "CA",  "postalCode": "94501"  }  },  "clientReferenceInformation": {  "code": "new-customer-order-id" **🡨** order ID  }  } |

Example response data: (not all properties included)

|  |
| --- |
| {  "status": "AUTHORIZED",  "paymentAccountInformation": {  "tokenInformation": {  "instrumentidentifierNew": false,  "customer": {  "id": "23FE2439AD65BBCFE063AF598E0A66A9" **🡨** customer id  },  "paymentInstrument": {  "id": "23FE0306218C3D37E063AF598E0A7E90" **🡨** payment token  },  "instrumentIdentifier": {  "id": "7031410000201134444",  "state": "ACTIVE"  }  }  } |

Following a successful CyberSource response, call [Find Payment Method](#_Find_Payment_Method) using the **tokenInformation.paymentInstrument.id** to get these payment method properties:

|  |  |
| --- | --- |
| **payment\_methods** | **createPayment response** |
| token | tokenInformation.paymentInstrument.id |
|  | **Find Payment Method response** |
| source | from card.type eg: “visa” |
| nickname | from card.type and \_embedded instrumentIdentifier.card.number |
| credit\_card\_expiration\_date | from card.expirationMonth and card.expirationYear |
| is\_default | true (when creating new customer) |

The “token” property in the “payments”.payment\_methods record is defined as varchar(64), so no change is required to use it to store the CyberSource value eg: “23FE0306218C3D37E063AF598E0A7E90”.

**NOTE:** When adding a payment method with a zero dollar authorization, the Cybersource dashboard shows this transaction as having a successful “Subscription Creation”:

A screenshot of a computer

Description automatically generated

According to the Cybersource docs, a subscription includes:

* Customer contact information, such as billing and shipping information.
* Customer payment information, such as card type, masked account number, and expiration date.
* Customer order information, such as the transaction reference number and merchant defined data fields.

There are two types of subscriptions but neither seems to apply to our use cases, and I did not provide the “required” information during my testing so not sure how a subscription was created…

**Installment Subscriptions** which will charge a customer’s preferred payment method on a fixed number of scheduled payments. The number of payments, the amount and frequency of each payment, and the start date for processing the payments are required.

**Recurring Subscriptions** which will charge a customer’s preferred payment method on a recurring payment basis with no specific end date. The amount and frequency of each payment and the start date for processing the payments are required.

**NOTE:** I’m assuming that no recurring payment will happen because no schedule or amount was provided.

<https://developer.cybersource.com/library/documentation/dev_guides/Token_Management/SO_API/html/Topics/Automatically_Preauthorizing_an_Account.htm>

<https://developer.cybersource.com/library/documentation/dev_guides/Recurring_Billing/UBC_Deprecated/html/Topics/Customer_Subscriptions.htm>

## Add Payment Method

This requires a transient token (JWT) generated by the Flex API from the Microform used to enter the payment information. See the [Microform User Interface](#_Microform_User_Interface) sections for details on creating the transient token.

The payment service currently receives a “nonce” generated from the Braintree drop-in UI, and we would use this same parameter to receive the transient token from the Cybersource UI.

*“You can also*[*add additional Payment Instruments to a Customer via a Payment Authorization*](https://developer.cybersource.com/api-reference-assets/index.html#payments_payments_process-a-payment_samplerequests-dropdown_authorization-with-token-create_authorization-create-default-payment-instrument-shipping-address-for-existing-customer_liveconsole-tab-request-body)*.”*

<https://developer.cybersource.com/docs/cybs/en-us/tms/best-practices/all/rest/tms-best-practices/wallet-intro/3-add-pymt-method.html>

Example request data:

|  |
| --- |
| {  "paymentInformation": {  "customer": {  "id": "23FE2439AD65BBCFE063AF598E0A66A9" 🡨 existing customer  }  },  "clientReferenceInformation": {  "code": "add-new-payment-method" **🡨** order id  },  "processingInformation": {  "actionList": [  "TOKEN\_CREATE" **🡨** create the following token type  ],  "actionTokenTypes": [  "paymentInstrument"  ],  "capture": false, **🡨** auth only  "commerceIndicator": "internet"  },  "orderInformation": {  "amountDetails": {  "totalAmount": "0",  "currency": "USD"  },  "billTo": {  "email": "cybersource\_test\_002@yopmail.com",  "firstName": "David",  "lastName": "Balmer",  "address1": "3355 Geary Blvd.",  "locality": "San Francisco",  "country": "US",  "administrativeArea": "CA",  "postalCode": 94118  }  },  "tokenInformation": {  "transientTokenJwt": "<from microform.createToken()>",  "paymentInstrument": {  "default": false **🡨** don’t replace their default payment method  }  }  } |

Example response data: (not all properties included)

|  |
| --- |
| {  "status": "AUTHORIZED",  "paymentAccountInformation": {  "tokenInformation": {  "instrumentidentifierNew": false,  "customer": {  "id": "23FE2439AD65BBCFE063AF598E0A66A9"  },  "paymentInstrument": {  "id": "23FE2439B3F7BBCFE063AF598E0A66A9" **🡨** payment token  },  "instrumentIdentifier": {  "id": "7037870000055421111",  "state": "ACTIVE"  }  }  } |

## Delete Payment Method

**DELETE** [https://apitest.cybersource.com/tms/v1/paymentinstruments/{{token}}](https://apitest.cybersource.com/tms/v1/paymentinstruments/%7b%7btoken%7d%7d)

**NOTE:** Cannot delete if it’s the customers default payment method.

|  |
| --- |
| {  "errors": [  {  "type": "conflict",  "message": "Action cannot be performed as the PaymentInstrument is the customers default"  }  ]  } |

Would need to add a new default payment method, or update an existing one to default, before deleting.

Update Payment Method

<https://developer.cybersource.com/docs/cybs/en-us/tms/best-practices/all/rest/tms-best-practices/wallet-intro/9-defaults.html>

Customer 23FE2439AD65BBCFE063AF598E0A66A9 has these payment methods:

* 23FE0306218C3D37E063AF598E0A7E90 (default)
* 23FE2439B3F7BBCFE063AF598E0A66A9
* 23FE2439B056BBCFE063AF598E0A66A9 🡨 tried to make this the default, but doesn’t work

**PATCH** [https://apitest.cybersource.com/tms/v2/customers/{{customer}}/payment-instruments/{{token}}](https://apitest.cybersource.com/tms/v2/customers/%7b%7bcustomer%7d%7d/payment-instruments/%7b%7btoken%7d%7d)

|  |
| --- |
| {  "default": true  } |

returns this error:

|  |
| --- |
| {  "errors": [  {  "type": "conflict",  "message": "Action cannot be performed as the if-match header value does not match the token etag"  }  ]  } |

## Find Payment Method

GET [https://apitest.cybersource.com/tms/v2/customers/{{customer}}/payment-instruments/{{token}}](https://apitest.cybersource.com/tms/v2/customers/%7b%7bcustomer%7d%7d/payment-instruments/%7b%7btoken%7d%7d)

Example response: (not all properties included)

|  |
| --- |
| {  "id": "23FE0306218C3D37E063AF598E0A7E90", **🡨** token  "default": true, **🡨** this is their default payment method  "state": "ACTIVE",  "card": {  "expirationMonth": "12",  "expirationYear": "2024",  "type": "002"  },  "buyerInformation": {  "currency": "USD"  },  "billTo": {  "firstName": "DAVID",  "lastName": "BALMER",  "address1": "3355 Geary Blvd.",  "locality": "San Francisco",  "administrativeArea": "CA",  "postalCode": "94118",  "country": "US",  "email": "cybersource\_test\_002@yopmail.com"  },  "processingInformation": {  "billPaymentProgramEnabled": false  },  "instrumentIdentifier": {  "id": "7031410000201134444"  }  } |

## Sale Transaction

In Braintree we’re passing **submitForSettlement = true**, but in CyberSource we need to pass **capture=true** to initiate the settlement at the same time as the hold authorization.

A capture is a follow-on transaction to an authorization. It is used to transfer the authorized funds from the customer's account to the merchant account.

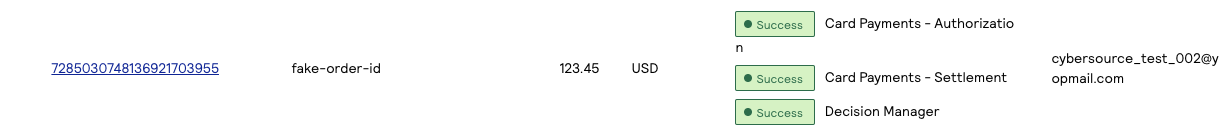
Example request data:

|  |
| --- |
| {  "paymentInformation": {  "paymentInstrument": {  "id": "23FE0306218C3D37E063AF598E0A7E90" **🡨** payment token  }  },  "orderInformation": {  "amountDetails": {  "totalAmount": "105.00",  "currency": "USD"  }  },  "processingInformation": {  "capture": true, **🡨** request auth and submit for settlement  "commerceIndicator": "internet"  },  "clientReferenceInformation": {  "code": "fake-order-id" **🡨** order id  }  } |

Example response data: (not all properties included)

|  |
| --- |
| {  "id": "7285030748136921703955", **🡨** transaction id  "submitTimeUtc": "2024-10-09T19:44:35Z",  "status": "AUTHORIZED",  "reconciliationId": "79518956",  "clientReferenceInformation": {  "code": "fake-order-id"  },  "processorInformation": {  "authIndicator": "1",  "approvalCode": "831000",  "transactionId": "0602MCC603474", **🡨** what is this? cannot search in dashboard  "networkTransactionId": "0602MCC603474",  "responseCode": "00",  "avs": {  "code": "Y",  "codeRaw": "Y"  }  } |

The Cybersource dashboard shows this transaction as having both and Authorization and Settlement:



Following a successful CyberSource response, create either a direct\_payment\_transactions or wallet\_refill\_transactions record using these properties:

|  |  |
| --- | --- |
| **transaction record** | **createPayment response** |
| payment\_processor\_transaction\_id | id |
| payment\_status | mapped from status |
| payment\_failure\_code | tbd |
| payment\_failure\_message | tbd |

The transaction id property in both tables is defined as varchar(50), so no change is required to use it to store the CyberSource value eg: “7285030748136921703955”.

**TODO:** get list of possible failure statuses and messages

## Hold Authorization Transaction

In Braintree we’re passing **submitForSettlement = false**, but in CyberSource omitting the **capture** property will have the same effect and only perform the hold authorization.

A hold authorization transaction can later be submitted for settlement using a Capture transaction, but these are typically not performed in real time. They are placed in a batch file and sent to the processor and the processor settles all the captures at one time.

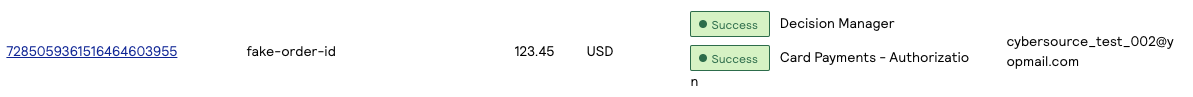
Example request data:

|  |
| --- |
| {  "paymentInformation": {  "paymentInstrument": {  "id": "23FE0306218C3D37E063AF598E0A7E90" **🡨** payment token  }  },  "orderInformation": {  "amountDetails": {  "totalAmount": "105.00",  "currency": "USD"  }  },  "processingInformation": {  "capture": false, **🡨** request auth only (can also omit this property)  "commerceIndicator": "internet"  },  "clientReferenceInformation": {  "code": "fake-order-id" **🡨** order id  }  } |

Example response data: (not all properties included)

|  |
| --- |
| {  "id": "7285059361516464603955", **🡨** transaction id  "submitTimeUtc": "2024-10-09T20:32:17Z",  "status": "AUTHORIZED",  "reconciliationId": "W6SFV21YR3RA",  "clientReferenceInformation": {  "code": "fake-order-id"  },  "processorInformation": {  "authIndicator": "1",  "approvalCode": "831000",  "transactionId": "0602MCC603474",  "networkTransactionId": "0602MCC603474",  "responseCode": "00",  "avs": {  "code": "Y",  "codeRaw": "Y"  }  }  } |

The Cybersource dashboard shows this transaction as having only an Authorization:

****

Following a successful CyberSource response, create an authorization\_transactions record using these properties:

|  |  |
| --- | --- |
| **transaction record** | **createPayment response** |
| payment\_processor\_transaction\_id | id |
| payment\_status | mapped from status |
| payment\_failure\_code | tbd |
| payment\_failure\_message | tbd |
| expiration\_date | tbd |

The transaction id property in this table is defined as varchar(50), so no change is required to use it to store the CyberSource value eg: “7285059361516464603955”.

**TODO:** get list of possible failure statuses and messages

## Void Transaction

*“A void cancels a capture or credit request that was submitted but not yet processed by the processor.*

*Capture and credit requests are usually submitted once a day. A void request is declined when the capture or credit request has already been sent to the processor.”*

*“After a void is processed, you cannot credit or capture the funds. You must perform a new transaction to capture or credit the funds. Further, when you void a capture, a hold remains on the authorized funds. If you are not going to re-capture the authorization, and if your processor supports authorization reversal after void (ARAV), you should request an authorization reversal to release the hold on the unused funds.”*

<https://developer.cybersource.com/docs/cybs/en-us/payments/developer/vital/sa/payments/payments-intro/payments-services-intro/payments-intro-processing-void.html>

Example request data:

|  |
| --- |
| {  "clientReferenceInformation": {  "code": "some-order-id" **🡨** order id recommended but not required  }  } |

Example success response data: (not all properties included)

|  |
| --- |
| {  "id": "7285186975596936503954", **🡨** transaction id being voided  "submitTimeUtc": "2024-10-10T00:04:57Z",  "status": "VOIDED", **🡨** success  "clientReferenceInformation": {  "code": "fake-order-id"  },  "voidAmountDetails": {  "voidAmount": "12.34",  "currency": "usd"  }  } |

Example failure response data: (transaction cannot be voided)

|  |
| --- |
|  |

## Refund Transaction

TBD

Example request data:

|  |
| --- |
|  |

Example success response data: (not all properties included)

|  |
| --- |
|  |

Example failure response data: (transaction cannot be refunded)

|  |
| --- |
|  |

## Transaction Search

<https://developer.cybersource.com/docs/cybs/en-us/txn-search/developer/all/rest/txn-search/txn-search-intro.html>

*“To search for settled transactions in Cybersource, you can use the Transaction Search API, and the request body should include the search name, time zone, and search parameters. A single search can return up to 2,000 transactions.”*

Example request data (for settled transactions):

|  |
| --- |
| {  "save": false, **🡨** could save a named search if required  "name": "test",  "timezone": "America/Los\_Angeles",  "searchType": "pendingSettlement", **🡨** Settlements Pending Batch  "query": "submitTimeUtc:[NOW/DAY-1DAYS TO NOW/DAY+1DAY}", **🡨** date range  "offset": 0,  "limit" 100, **🡨** got a 404 error when trying 2600, but 2500 was ok  "sort": "submitTimeUtc:asc"  } |

<https://developer.cybersource.com/docs/cybs/en-us/apple-pay/developer/six/rest/applepay/applepay-txn-reports/transactionby_phase.html>

***“Settlements Pending Batch:*** *Use this filter to find transactions that were settled by the merchant in the past 48 hours, but that are not yet forwarded by Business Center to the backend processors to fulfill a transaction.”*

Example response data:

|  |
| --- |
| {  "save": false,  "name": "test",  "timezone": "America/Los\_Angeles",  "query": "submitTimeUtc:[NOW/DAY-1DAYS TO NOW/DAY+1DAY}",  "offset": 0,  "limit": 100,  "sort": "submitTimeUtc:asc",  "count": 12,  "totalCount": 12,  "\_embedded": {  "transactionSummaries": [  {  "id": "7285030748136921703955", **🡨** transaction id  "submitTimeUtc": "2024-10-09T19:44:34Z",  "applicationInformation": {  "reasonCode": "100",  "rCode": "1",  "rFlag": "SOK",  "applications": [  {  "name": "ics\_auth", **🡨** card authorization  "reasonCode": "100", **🡨** successful transaction  "rCode": "1",  "rFlag": "SOK",  "reconciliationId": "W1HKS21YLSOR",  "rMessage": "Request was processed successfully.",  "returnCode": 1010000  },  {  "name": "ics\_bill", **🡨** card settlement  "reasonCode": "100", **🡨** successful transaction  "rCode": "1",  "rFlag": "SOK",  "reconciliationId": "79518956",  "rMessage": "Request was processed successfully.",  "returnCode": 1260000  }  ]  }  }  ]  }  } |

See <https://developer.cybersource.com/docs/cybs/en-us/apple-pay/developer/amexbrighton/rest/applepay/reason_code_descriptions.html> for the list of “reasonCode” values.

## Find Expiring and Expired Credit Cards

CyberSource doesn’t return expiration info when creating a payment method, so we would just use the expirationYear and expirationMonth entered in the UI and returned from “Retrieve Customers Payment Instrument” response when creating the “payments”.payment\_method record.

# Microform User Interface

<https://developer.cybersource.com/docs/cybs/en-us/digital-accept-flex/developer/all/rest/digital-accept-flex/ctp-intro/ctp-tokens-intro.html>

*“The transient token is a reference to the payment data collected on your behalf. Tokens enable secure card payments without risking exposure to sensitive payment information. The transient token is a short-term token with a duration of 15 minutes.”*

**TODO:** what’s required to use the website drop-in UI …

**TODO:** can we use Click to Pay to create a new user or add a payment method?

For mobile applications, the card acceptance fields of PAN and CVV must be hosted on a web page and the native application must load the hosted card entry form web page in a web view.

* iOS sample: <https://github.com/cybersource/flex-v2-ios-sample>
* Android sample: <https://github.com/cybersource/flex-v2-android-sample>

Sample decoded transient token (JWT):

|  |
| --- |
| {  "iss": "Flex/07",  "exp": 1728068879,  "type": "mf-2.0.0",  "iat": 1728067979,  "jti": "1E36E8UME8403QX532YNKVO56QZQ6JZV46WSNFLIESC7S0XHUHJC67003D0F523A",  "content": {  "paymentInformation": {  "card": {  "expirationYear": {  "value": "2024"  },  "number": {  "maskedValue": "XXXXXXXXXXXX1111",  "bin": "411111"  },  "securityCode": {},  "expirationMonth": {  "value": "12"  },  "type": {  "value": "001"  }  }  }  }  } |

# Payments Schema

**REQUIREMENT:** The payment module shall support both Braintree and CyberSource based on configuration

gateway = new braintree.BraintreeGateway({ // v3

environment: braintree.Environment[process.env.BRAINTREE\_ENVIRONMENT],

merchantId: process.env.BRAINTREE\_MERCHANT\_ID,

publicKey: process.env.BRAINTREE\_PUBLIC\_KEY,

privateKey: process.env.BRAINTREE\_PRIVATE\_KEY

});

## Merchant Accounts

TODO: how do we set the payment provider for

**Q:** do we know the merchant account when adding a payment method for a new/existing user?

**A.** no, only when creating a wallet to get the wallet type

agency\_id and agency\_program\_id are only passed when making payment from parking-backend using ??? appConstants.PAYMENT\_MODULE.AGENCY\_NAME and appConstants.PAYMENT\_MODULE.AGENCY\_PROGRAM\_NAME

|  |  |
| --- | --- |
| **Column** | **Comments** |
| payment\_source | use “**CyberSource**” and change “default” to “Braintree”??? |
| merchant\_account\_id | the CyberSource **merchantID** eg: “transsightdev\_1718140723” |
| threshold\_amount | null |
| threshold\_comparison | null |
| merchant\_account\_type | eg: “Parking” |

## Payment Methods

**NOTE:** payment method is not associated with a Merchant Account

|  |  |
| --- | --- |
| **Column** | **Comments** |
| token | varchar(64) |
| source | varchar(100) eg: “American Express” |
| nickname | varchar(100) eg: “American Express ending in 1003” |
| is\_default | true for first payment method created |
| credit\_card\_expiration\_date | timestamp |
| **payment\_provider** | **“Braintree” or “CyberSource”** |

# Testing

## Dave’s Sandbox Account

<https://ebc2test.cybersource.com/ebc2>

Your Account ID: TransSightDev\_1718140723\_nt

Your Organization ID: transsightdev\_1718140723

username: evadremlab

password: SYq9dAdi43!!

Generated REST certificate and downloaded as **transsightdev\_1718140723.p12**

REST – Certificate key password: Zaq12wsx!!!

|  |
| --- |
| MerchantId = 'transsightdev\_1718140723';  MerchantKeyId = 'ad81163c-aa36-471d-be21-9ac7c7ebbe99';  MerchantSecretKey = 'H/P7ehPswD5Y2xr/kUTIWG2mk37G63rsI5tvfCtZ34g='; |

## Test Credit Card Numbers

<https://developer.cybersource.com/hello-world/testing-guide.html>

## Refund or Cancellation

**TODO:** how often do Braintree and CyberSource settle transactions?

**TODO:** how can we let someone cancel a transaction that has not started settlement?

## MicroForm Process Flow (OUTDATED)

<https://developer.cybersource.com/docs/cybs/en-us/digital-accept-flex/developer/all/rest/digital-accept-flex/microform-integ-v2.html>

<https://github.com/cybersource/cybersource-flex-samples-node>

**Creating the Server-Side Capture Context**

<https://developer.cybersource.com/docs/cybs/en-us/digital-accept-flex/developer/all/rest/digital-accept-flex/microform-integ-v2/microform-integ-getting-started-v2/creating-server-side-context-v2.html>

Send an **authenticated** POST request to https://apitest.cybersource.com/microform/v2/sessions. Include the target origin URL and at least one accepted card type in the content of the body of the request.

**NOTE:** done by the /checkout example, and sets the **keyInfo** property before rendering the Payment Form defined in express-microform/views/index.ejs:

|  |
| --- |
| const {  ApiClient,  MicroformIntegrationApi,  GenerateCaptureContextRequest  } = require('cybersource-rest-client');  const apiClient = new ApiClient();  const instance = new MicroformIntegrationApi(cyberSourceConfig, apiClient);    const requestData = GenerateCaptureContextRequest.constructFromObject({  clientVersion: 'v2.0',  targetOrigins: ['http://localhost:3000'],  allowedCardNetworks: ['VISA', 'MASTERCARD', 'AMEX', 'DISCOVER']  });  instance.generateCaptureContext(requestData, (err, data, response) => {…} |

**NOTE:** when I added only “VISA” to the allowedCardNetworks it did not prevent me from entering a MasterCard number or getting MasterCard payment token.

**NOTE:** In the FLEX sample code, selecting VISA from the dropdown but entering a Mastercard number returned a 400 error but no error message or details.

**Setting Up the Client Side**

<https://developer.cybersource.com/docs/cybs/en-us/digital-accept-flex/developer/all/rest/digital-accept-flex/microform-integ-v2/microform-integ-getting-started-v2/setting-up-client-side-v2.html>

This is done by the Payment Form defined in express-microform/views/index.ejs

**Validating the Transient Token**

<https://developer.cybersource.com/docs/cybs/en-us/digital-accept-flex/developer/all/rest/digital-accept-flex/microform-integ-v2/microform-integ-getting-started-v2/setting-up-client-side-v2/validating-the-transient-token-v2.html>

**NOTE:** this is not done by the example code

**Example Transient Token (displayed by /token)**

eyJraWQiOiIwOFQ4REU5Q3N3QzRJeHZFaVBEc1BBN1NZWHh2Q3hDeiIsImFsZyI6IlJTMjU2In0..as7VMrDMtdFmhjieVSNT6w5Rc-6DCOawT1pYhot4UagErYfXkahwzwiDdgeTDinUuN9i\_FCpotnL-JKBGKUbFkgi2uZOcTCBt6TZPh67Us9JmHDiV0ZQgdja-T1gQg42SbhLbuffo3KWENmoUI59bKD9sAHQ3-Oc54ir-3aEm89756N8hhCxE7DcURh5MFXH6FX6Ni-sZvK6UqwU5ANZSG83MVwkvkWpxGTpjHQL0Nh3aR\_ZxR8aDY6xCA0dIvOz\_DUFqtSrCMo0tZ605vrnzW2LbQJJFJVzxqpdaJa9yxfmBtbOC4dRRMT8pdXgD9Ss4FKp65XDIVbsOW23MDJ5Dg

decoded (using jwt.io):

|  |
| --- |
| {  "data": {  "expirationYear": "2024",  "number": "555555XXXXXX4444",  "expirationMonth": "12",  **"type": "002" 🡨 MasterCard**  },  "iss": "Flex/07",  "exp": 1727900466,**🡨 add 3 trailing zeros to convert to JS Date (below)**  "type": "mf-0.11.0",  "iat": 1727899567,  "jti": "1E21545B6X5WBVHRMPFIIEFPAMF99RNUU4J0JXDOLZW0C1QLSRY866FDAB3217DB",  "content": {  "paymentInformation": {  "card": {  "expirationYear": {  "value": "2024"  },  "number": {  "maskedValue": "XXXXXXXXXXXX4444",  "bin": "555555"  },  "securityCode": {},  "expirationMonth": {  "value": "12"  },  "type": {  "value": "002"  }  }  }  }  } |

“exp” converted to JS Date is incorrect: “1/20/1970, 3:58:20 PM”:

|  |
| --- |
| const date = new Date(1727900466);  console.log(date.toLocaleString()); |

“exp” converted to JS Date by adding 3 trailing zeroes is correct “10/2/2024, 1:35:54 PM”, and is 15 minutes after the converted “iat” “10/2/2024, 1:20:54 PM”:

|  |
| --- |
| const date = new Date(1727900466**000**);  console.log(date.toLocaleString()); |

**Create a customer with validated payment details (from transient token)**

<https://developer.cybersource.com/docs/cybs/en-us/tms/developer/ctv/rest/tms/tms-cust-tkn/tms-cust-tkn-create-valid-pay-intro.html>

<https://developer.cybersource.com/docs/cybs/en-us/digital-accept-flex/developer/all/rest/digital-accept-flex/da-payments/da-auth-token-create-tms-task.html>

At the end of a successful request:

|  |
| --- |
| {  "status": "AUTHORIZED",  "submitTimeUtc": "2024-10-02T23:54:43Z",  "tokenInformation": {  "instrumentidentifierNew": true,  "instrumentIdentifier": {  "state": "ACTIVE",  "id": "7031410000201134444"  },  "paymentInstrument": {  "id": "238812209805C8ABE063AF598E0A2101"  },  "customer": {  "id": "2387E7B265C60B82E063AF598E0AB911"  }  } |

**Create a customer payment instrument token (from transient token)**

<https://developer.cybersource.com/docs/cybs/en-us/tms/developer/ctv/rest/tms/tms-cust-pi-tkn/tms-cust-pi-tkn-create-intro.html>

For and existing customer with a $0.00 authorization

**Authorize a Payment with an Instrument Identifier**

**QUESTION:** is this just an auth or a payment? Try sending “capture”: false for an auth

<https://developer.cybersource.com/docs/cybs/en-us/payments/developer/ctv/rest/payments/payment-tms-process-intro/tms-ii-tkn-pay-intro.html>

# Developer Notes

## Changes in latest version of cybersource-rest-client

The <https://github.com/cybersource/cybersource-rest-samples-node> repo does have the latest usage, but

<https://github.com/cybersource/cybersource-flex-samples-node> has outdated code, so have a look at this doc for some of the changes:

<https://stackoverflow.com/questions/77962470/cybersource-rest-client-webpack-imported-module-0-keygenerationapi-is-not-a-c>

# References

## Sequence Diagram for Tokenization and Payment Processing

A document with text and arrows

Description automatically generated

Figure 1 Sequence diagram for tokenization and payment processing

# Settlements

## Braintree

<https://developer.paypal.com/braintree/articles/au/transactions/settlement-funding-timeline>

## Cybersource

???

# Issues and Questions

## Billing information required when vaulting each payment method

|  |
| --- |
| billTo: { // all are required  email: 'david.balmer@transsight.com',  firstName: 'David',  lastName: 'Balmer',  address1: '3355 Geary Blvd.',  locality: 'San Francisco', // city  country: 'US', // two character ISO code  administrativeArea: 'CA', // state code  postalCode: '94118' // zipcode  } |

If these required properties are missing, the request is rejected with a validation error:

|  |
| --- |
| {  "status": "INVALID\_REQUEST",  "reason": "MISSING\_FIELD",  "message": "Declined - The request is missing one or more fields",  "details": [  {  "field": "orderInformation.billTo.locality",  "reason": "MISSING\_FIELD"  },  {  "field": "orderInformation.billTo.lastName",  "reason": "MISSING\_FIELD"  },  {  "field": "orderInformation.billTo.email",  "reason": "MISSING\_FIELD"  },  {  "field": "orderInformation.billTo.address1",  "reason": "MISSING\_FIELD"  },  {  "field": "orderInformation.billTo.country",  "reason": "MISSING\_FIELD"  }  ]  } |

**TODO:** need to find out from Cybersource if there is any way to specify which fields are required, because

Braintree only requires the card number, expiration date and zipcode, **but not security code**.

# Summary

Cybersource does not support saved payment methods for PayPal or Venmo.

No schema changes required to store the Cybersource customer ID, payment method token, or transaction ID.

Adding a Cybersource payment method requires more information than Braintree:

* security code (CCV)
* billing information
  + email
  + first and last name
  + address line 1
  + city
  + state and country codes