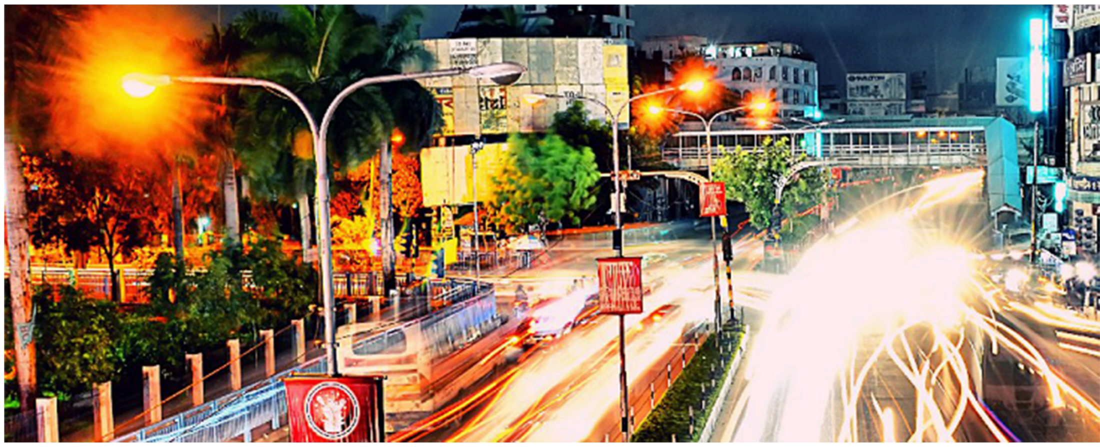
Unified Checkout

REST API

Implementation Guide

cybersource
A Visa Solution



Version: 21.01

Contents

Recent Revisions to This Document

21.01

Initial version of the Unified Checkout Implementation Guide

About This Guide

This section provides you with information about Unified Checkout REST API Implementation guide.

Audience and Purpose

This document is written for merchants who want to enable the Unified Checkout SDK to accept payments on their ecommerce page. This document provides an overview for integrating the Unified Checkout SDK and describes how to request the Cybersource API to process an authorization. Merchants must use the Unified Payments Javascript SDK to place a digital button widget on their ecommerce site, allowing Cybersource to capture payment data on their behalf.

Conventions

The following special statements are used in this document:

Important: An *Important* statement contains information essential to successfully completing a task or learning a concept.

Warning: A *Warning* contains information or instructions, which, if not heeded, can result in a security risk, irreversible loss of data, or significant cost in time or revenue or both.

Related Documentation

Refer to the Support Center for complete technical documentation:

http://www.cybersource.com/support_center/support_documentation

Customer Support

For support information about any service, visit the Support Center:

http://www.cybersource.com/support

Unified Checkout

Secure Acceptance Unified Checkout is a web acceptance technology built to provide a single interface for the acceptance of a multitude of digital payment options within your ecommerce experience.

Unified Checkout provides you with the ability to place a multitude of payment options within your ecommerce experience with a single integration.

How it works

Unified Checkout allows you to focus on your shopping experience and encapsulates a multitude of payment options within a single integration.

The provided JavaScript library enables you to place a button widget within your ecommerce experience. This widget offers a multitude of payment options to your customers.

When a customer selects a payment option from the button widget, Unified Checkout will handle all the interactions with the digital payment selected and will provide a common response to your ecommerce system.

Button Widget

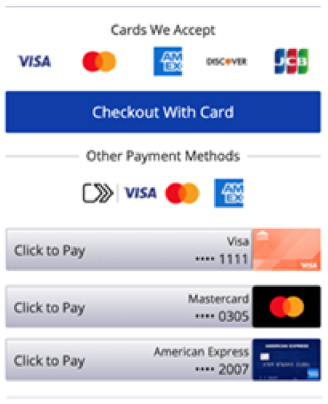


Image is currently displaying a recognized secure remote commerce experience & guest checkout payment options

Getting Started

Secure Acceptance Unified CHeckout consists of a server-side component and a client-side JavaScript library.

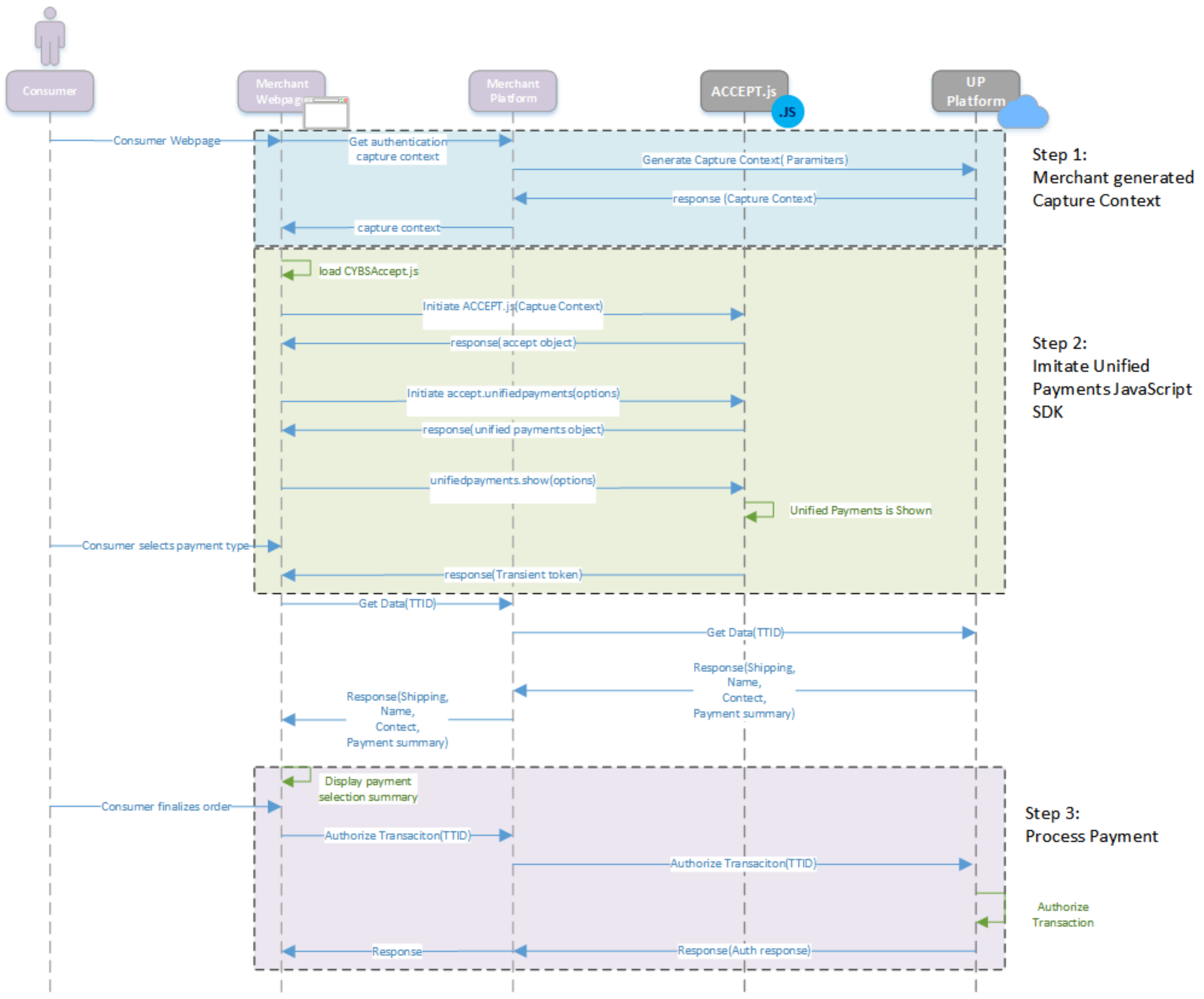
The server-side component authenticates your merchant identity and provides the instructions to the system how to act within your payment experience. The response contains limited use public keys, used for end to end encryption and merchant specific payment information that will drive the interaction of the application. The client-side JavaScript library will dynamically place digital payment options into your e-commerce page in a secure fashion.

Integration Flow

The integration process consists of two basic steps that need to be done in sequance

1. Send a Server to Server request for a capture context. This is a fully authenicated request for the JWT needed to invoke the front end JavaScript library

2. Invoke the Unified Checkout JavaScript Library using the JWT resposne from your capture context request


Consumer
Merchant Webpage
Merchant Platform
ACCEPT.js
.JS
UP Platform
Consumer Webpage
Get authentication capture context
Generate Capture Context(Paramiters)
response (Capture Context)
capture context
Step 1: Merchant generated Capture Context
load CYBSAccept.js
Initiate ACCEPT.js(Captue Context)
response(accept object)
Initiate accept.unifiedpayments(options)
response(unified payments object)
unifiedpayments.show(options)
Unified Payments is Shown
Consumer selects payment type
response(Transient token)
Step 2: Imitate Unified Payments JavaScript SDK
Get Data(TTID)
Get Data(TTID)
Response(Shipping, Name, Contect, Payment summary)
Response(Shipping, Name, Contect, Payment summary)
Display payment selection summary
Consumer finalizes order
Authorize Transaciton(TTID)
Authorize Transaciton(TTID)
Authorize Transaction
Response
Response(Auth response)
Step 3: Process Payment

Server-Side Setup

The server-side component provides the following:

- A transaction specific public key. This key is used by the customer's browser to protect the Unified Checkout transaction.
- An authenticated context description package that manages the payment experience on the client. This includes payment options available, payment interface styling, interaction methods.

The functions are compiled in Json Web Token (JWT) object referred to as the capture context.

Generate Capture Context

The capture context request contains all of the merchant specific parameters that tell the front-end JavaScript library how to behave within your payment experience.

The capture context is a signed JSON Web Token (JWT) containing the following 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 with long lived keys, allowing for validation of their authenticity.

Capture context is available with the following networks:

- AMEX
- DINERSCLUB
- DISCOVER
- JCB
- MASTERCARD
- VISA

Required Fields

Your capture context request must include the following fields:

- **targetOrigins**
- **clientVersion**
- **allowedCardNetworks**
- **allowedPaymentTypes**
- **country**
- **locale**
- **captureMandate.billingType**
- **capturemandate.requestEmail**

- **captureMandate.requestPhone**
- **captureMandate.requestShipping**
- **captureMandate.shipToCountries**
- **captureMandate.showAcceptedNetworkIcons**
- **orderInformation.amountDetails.totalAmount**
- **orderInformation.amountDetails.currency**

Requesting the Capture Context

To request the capture context:

1. Create a request payload that includes the required and optional customer transaction fields.
2. Send the request payload using a `POST` request to `/up/v1/capture-contexts`.
   A JavaScript Web Token (JWT) called the capture context is returned.
3. Use the capture context to invoke the JavaScript SDK on the client.

Example Request

The following is an example of a Capture Context request:

```
{

  {
    "targetOrigins": [
      "https://unified-payments.appspot.com"
    ],
    "clientVersion": "0.5",
    "allowedCardNetworks" : [ "VISA", "MASTERCARD", "AMEX" ],
    "allowedPaymentTypes" : [ "PANENTRY", "SRC" ],
    "country": "US",
    "locale": "en_US",
    "captureMandate": {
      "billingType": "FULL",
      "requestEmail": true,
      "requestPhone": true,
      "requestShipping": true,
      "shipToCountries": [
        "US",
        "UK"
      ],
      "showAcceptedNetworkIcons": true
    },
    "orderInformation": {
      "amountDetails": {
        "totalAmount": "21.00",
        "currency": "USD"
      },
      "billTo": {
        "address1": "277 Park Avenue",
        "address2": "50th Floor",
        "address3": "Desk NY-50110",
        "address4": "address4",
        "administrativeArea": "NY",
        "buildingNumber": "buildingNumber",
        "country": "US",
        "district": "district",
        "locality": "New York",
        "postalCode": "10172",
        "company": {
          "name": "Visa Inc",
          "address1": "900 Metro Center Blvd",
          "address2": "address2",
          "address3": "address3",
          "address4": "address4",
          "administrativeArea": "CA",
          "buildingNumber": "1",
          "country": "US",
          "district": "district",
          "locality": "Foster City",
          "postalCode": "94404"
        },
        "email": "al.kelly@visa.com",
        "firstName": "Alfred",
        "lastName": "Kelly",
        "middleName": "F",
        "nameSuffix": "Jr",
        "title": "Mr",
        "phoneNumber": "1234567890",
        "phoneType": "phoneType"
      },
      "shipTo": {
        "address1": "CyberSource",
        "address2": "Victoria House",
        "address3": "15-17 Gloucester Street",
        "address4": "string",
        "administrativeArea": "CA",
        "buildingNumber": "string",
        "country": "UK",
        "district": "string",
        "locality": "Belfast",
        "postalCode": "BT1 4LS",
        "firstName": "Joe",
        "lastName": "Soap"
      }
    }
  }

}
```

Client-Side Setup

The Secure Acceptance Unified Checkout Javascript library is used to integrate with your ecommerce site.

Secure Acceptance Unified Payments Javascript library has two primary components:

- The Button widget; Lists the payment options available to the customer.
- The payment acceptance screen: Captures payment information from the consumer. The payment acceptance screen can be set up to be integrated with your webpage or added as a sidebar.

Integration Steps

To set up the client:

1. Load JavaScript library
2. Initialize Accept object with capture context
3. Initialize Unified payment object with optional parameters
4. Show the button list and/or payment acceptance screen

The response to these interactions is a transient token that can be used to complete payments in a secure manner.

Loading the Javascript Library and Invoking the Accept Function

The Javascript library is used to invoke unified payments on your page.

Javascript Library URL:

This URL is retrieved from the Capture Context response. When decoded this can be found in the JSON parameter **clientLibrary**.

`https://stageup.cybersource.com/up/v1/assets/0.6.0/SecureAcceptance.js`

Once the library is loaded, the Accept function should be invoked using the Capture Context received from your initial serverside request.

Example Request

The following shows how to initialize the SDK:

```
<script
 src="https://apitest.cybersource.com/up/v1/assets/0.5/SecureAcceptance.js"></script>
<script>
  Accept('header.payload.signature').then(function(accept) {
    // use accept object
  });
</script>
```

Adding the Button Widget and Payment Acceptance

After you initialize the Unified Checkout object, you can now add the button widget and payment acceptance screens to your webpage. The Unified Checkout widget and Payment Acceptance screens can be attached to any named element within your HTML.

Typically they are attached to explicit div named components intended to be replaced with the iframes.

Important:

If no location is specified for the payment acceptance screens than it will default to the side bar experience.

Example

Basic Setup

The following shows a basic setup with a full sidebar experience:

```
up.show().then(transientToken => console.log(transientToken));
```

Options

The following shows payment selection options in a container:

```
var options = { containers: { paymentSelection: '#paymentSelectionContainer' } };

up.show(options).then(transientToken => console.log(transientToken));

All screens embedded in containers
var options = {
  containers: {
    paymentSelection: document.querySelector('.my-payment-selectors'),
    paymentScreen: document.querySelector('.my-payment-screen'),
  }
};

up.show(options).then(transientToken => console.log(transientToken));
```

Transient Tokens

The transient token is a reference to the payment data collected upon your behalf. They enable secure card payments without risking exposure to sensitive payment information.

Transient Token Format

The transient token is issued as a JSON Web Token (JWT) (RFC 7519). A JWT takes the form of a string, consisting of three parts separated by dots ., which are:

- Header
- Payload
- Signature

An example JWT may look like:

```
xxxxx.yyyyy.zzzzz
```

The payload portion of the token is a Base64Url encoded JSON string and contains various claims. An example payload looks like:

```
{
    "iss": "Flex/07",
    "exp": 1597084897,
    "type": "gda-0.1.1",
    "iat": 1597083997,
    "jti": "1C26VZRFEIQMOO5H0504KH47I0AM2IZFC43V5L541HBLN9COI3L75F3194E196A1"
}
```

Token Verification

Upon receiving the transient token within your systems, its integrity should be cryptographically verified using the public key embedded within the capture context created at the beginning of this flow. Doing so verifies that Cybersource issued the token and that no tampering of the data has occurred in transit.

Payment Authorizations with Tokens

The Unified Checkout transitent token can be used to submit an authorization (or additional services). This eliminates the need to send sensitive payment data along with the request. The JWT ID (jti) within the transient token response or the entire token can be submitted depending on your integration path.

To send the transitient token with a request, use the **tokenInformationtokenSource_transientTokentransient_token** field. For example:

```
"tokenInformation": {
    "transientTokenJwt":
 "eyJraWQiOiIwOG4zUnVsRTJGQXJDRktycVRkZFlkWGZSWFhMNXFoNSIsImFsZyI6IlJTMj

 U2In0.eyJpc3MiOiJGbGV4LzA3IiwiZXhwIjoxNTk3MDg0ODk3LCJ0eXBlIjoiZ2RhLTAuMS4xIiwiaWF
0IjoxNTk3MD

 gzOTk3LCJqdGkiOiIxQzI2VlpSRkVJUU1PTzVIMDUwNEtINDdJMEFNMklaRkM0M1Y1TDU0MUhCTE45Q09
JM0w3NUYzMT

 k0RTE5NkExIn0.SNm1VZaZr3DkTqUg9CdV0F5arRe-uQU9oUWPKfWIpbIzIPZutRokv5DSDcM7asZIKNJ
yNIBx5DLsl_

 yQPrKgzhwQxZ8qbhto7cu3t-v8DHG2yO951plPQVQnj7x-vEDcXkLUL1F8sqY23R5HW-xSDAQ3AFLawCc
kn7Q2eudRGe

 uMhLWH742Gflf9Hz3KyKnmeNKA3o9yW2na16nmeVZaYGqbUSPVITdl5cMA0o9lEob8E3OQH0HHdmIsu5u
MA4x7DeBjfT

 KD1rQxFP3JBNVcv30AIMLkNcw0pHbtHDVzKBWxUVxvnm3zFEdiBuSAco2uWhC9zFqHrrp64ZvzxZqoGA
"
}
```

```
transient_token=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
<tokenSource><transientToken>XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX</transientToken></to
kenSource>
```

Example: Authorization Request with a Transient Token

```
{
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "processingInformation": {
    "commerceIndicator": "internet"
  },
  "tokenInformation": {
    "transientTokenJwt": "eyJraWQiOiIwOG4zUnVsRTJGQXJDRktycVRkZFlkWGZSWFhMNXFoNSIsImFs
ZyI6IlJTMjU2In0.eyJpc3MiOiJGbGV4LzA3IiwiZXhwIjoxNTk3MDg0ODk3LCJ0eXBlIjoiZ2RhLTAuMS4xIi
wiaWF0IjoxNTk3MDgzOTk3LCJqdGkiOiIxQzI2VlpSRkVJUU1PTzVIMDUwNEtINDdJMEFNMklaRkM0M1Y1TDU0
MUhCTE45Q09JM0w3NUYzMTk0RTE5NkExIn0.SNm1VZaZr3DkTqUg9CdV0F5arRe-uQU9oUWPKfWIpbIzIPZutR
okv5DSDcM7asZIKNJyNIBx5DLsl_yQPrKgzhwQxZ8qbhto7cu3t-v8DHG2yO951plPQVQnj7x-vEDcXkLUL1F8
sqY23R5HW-xSDAQ3AFLawCckn7Q2eudRGeuMhLWH742Gflf9Hz3KyKnmeNKA3o9yW2na16nmeVZaYGqbUSPVIT
dl5cMA0o9lEob8E3OQH0HHdmIsu5uMA4x7DeBjfTKD1rQxFP3JBNVcv30AIMLkNcw0pHbtHDVzKBWxUVxvnm3z
FEdiBuSAco2uWhC9zFqHrrp64ZvzxZqoGA"
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "21.00",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1Market St",
      "address2": "Address 2",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "4158880000"
    }
  }
}
```