# Service Provider Simulator

## Overview

The Service Provider Simulator works like a processor and allows you to test your transactions in a sandbox environment based on your linked Payment Provider Contract. See here how to get started.

The Simulator is designed to assist with integration testing and troubleshooting. If the simulator processor is not available to you, please contact your Verifone administrator. The Simulator can be configured without any need to provide your acquiring merchant ID or any other credentials.

## Availability

Available to all merchants who want to test in a sandbox environment.

## Authorizing and settling transactions

Once your Payment Provider Contract (PPC) is set up in the sandbox, you can start testing, mocking transaction processing and the settlement process as well.

A transaction may be captured either by using the `capture_now` flag during the authorization or the capture API.

Captured transactions are automatically updated to `Settled` status shortly after 23:45 PM (UTC).

This simulates the process in the production environment, where settlement takes place at a fixed schedule per acquirer (usually at the end of the day; visit the Supported acquirers documentation for more details). **Check the contract with your acquirer for settlement period details.**

Placing an order in the amount of 76.51 via the simulator will trigger the settlement process instantaneously (without having to wait for the automatic process) and all previously captured transactions will be settled.

> The Service Provider Simulator does not support pre-authorizations.

## Wallets testing

- To test Google Pay, you need to add your email to Google's test card suite group. For additional information and mock test cards data, read Google's Test card suite documentation.
- To test Apple Pay, follow Apple's instructions for Sandbox Testing
- You need to set up a wallet for your demo account. Refer to Advanced Payment Methods (APMs) documentation for more information.

Example of a Checkout request payload for a hosted payment page with a google wallet

```
{
"amount": 116,
"currency_code": "EUR",
"entity_id": "d5323bad-dc4d-4077-a6c9-25a705f4dd6a",
"customer": "62cb3527-e0a5-4ef6-afcc-d156b666c77d",
"configurations": {
"card": {
"mode": "PAYMENT",
"payment_contract_id": "13e295b6-af21-46ca-93b1-8f7d812e3aff"
},
```

```
"google_pay": {
"card": {
"sca_compliance_level": "WALLET",
"payment_contract_id": "13e295b6-af21-46ca-93b1-8f7d812e3aff",
"threed_secure": {
"threeds_contract_id": "6fc092d7-79b7-4729-8d30-4ab082b04791",
"transaction_mode": "S"
}
}
}
},
"merchant_reference": "1d35078c-1964-47ac-a50f-846ef28ebb53",
"i18n": {
"default_language": "en",
"show_language_options": true
}
}
```

By setting `"sca_compliance_level": "NONE"`, you can test the wallet without the `customer` ID and the `threed_sercure` object.

## Test cases

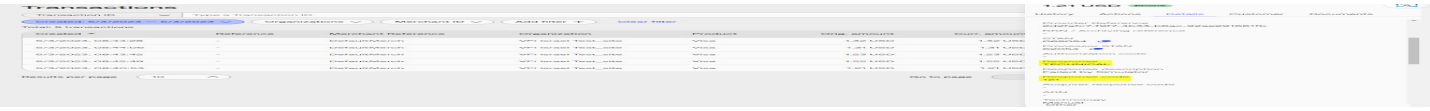
### Trigger specific error responses

Generally, all transactions initiated against the simulator would be approved with a code `0000`. However, in production, you may encounter failed transactions.

There are two ways to trigger unhappy flows:

- you can use a specific value for the `amount` and, in this case, when the amount ends in the values below, a relevant error code will be invoked as described in the table.
- you can use the `merchant_reference` parameters. When the `merchant_reference` is equal to the values below, a relevant error code will be invoked as described in the table.

| **`amount` last 3 digits** | **`merchant_reference`** | **`transaction_status` ( readTransaction API)** | **Response** | **`reason_code` ( readTransaction API)** |
|---|---|---|---|---|
| 121 | please 121 | FAILED | TECHNICAL | 121 |
| 122 | please 122 | FAILED | UNKNOWN | 122 |
| 123 | please 123 | FAILED | FAILED | 123 |
| 131 | please 131 | FAILED | MISSING | 131 |
| 132 | please 132 | AUTHORISED | PARTIAL | 132 |



Failed transactions' status is updated to SETTLED after settlement with the Simulator processor.

### Card verification value (CVV) check

You can simulate various CVV check results by including any of the following CVV values in a payment request.

| card.cvv | cvv_result | Meaning |
|---|---|---|
| 111 | 1 | Matched |
| 222 | 2 | Not matched |
| 333 | 3 | Not checked |
| any other value | 0 | Unavailable |

Simulating different CVV results for AMEX cards with 4-digit CVVs is currently unavailable.

### Address verification service check

You can simulate various AVS check results by including any of the following values in `address_line1` of `billing_address` in a payment request.

Read AVS results codes for additional information on the various applicable codes.

| customer.billing.address_1 | avs_result |
|---|---|
| 100 | A |
| 101 | B |
| 102 | C |
| 103 | D |
| 104 | E |
| 105 | F |
| 106 | G |
| 107 | I |
| 108 | K |

| customer.billing.address_1 | avs_result |
|---|---|
| 109 | L |
| 110 | M |
| 111 | N |
| 112 | O |
| 113 | P |
| 114 | R |
| 115 | S |
| 116 | T |
| 117 | U |
| 118 | W |
| 119 | X |
| 120 | Y |
| 121 | Z |

### 3-D Secure tests

To combine transactions with the 3-D Secure flow, simply use the cards listed for 3DS 1.0 and for 3DS 2.0 to trigger the relevant behavior.

## Test in production

After your account was set up and you received all your credentials to be used in the Production environment, you are ready to start testing.

> Note that transactions in the Production environment are real and will reflect into your bank account. Refunding the transactions at the end of your testing may be a good practice.