Card

This guide contains the required steps to process card payments through the API & Inject. The main payment actions that can be applied to card transactions will each be described here. A prerequisite is added to each section which is required in order to perform the action.

All examples are done with the minimum required fields, please view the API reference documentation to check up-to-date required fields. Always check the page of the connection you are using to process your payment, most connections have exceptions regarding which payment actions can be performed and other exceptions regarding the currencies, dynamic descriptor formatting, customer object requirements, etc.

Authorize

Action	Authorize
Description	Authorizing a card transaction will reserve the amount on the users card account at the issuer. Money will not yet be deducted from the bank account. It will defer per acquirer and issuer when the authorization is not valid anymore.
Link to API docs	https://sandbox.omni.verifone.cloud/docs/api#operation/createCardTransa...
Prerequisite 1	A tokenized card for the `card` field.
Prerequisite 2	An account with a card processor attached to it. The account ID will need to be used for the `account` field.
Prerequisite 3	Check the connection page for the acquirer you are using for any exceptions regarding the dynamic descriptor, currency, customer requirements or other requirements.

Post the following body to the `/transaction` endpoint:

```
{
"account": "account.id",
"amount": 1000,
"card": "card.id",
"capture_now": false,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
}
```

Notice that in the above example the `capture_now` field is set to `false`. This is the flag that is used to authorize a transaction. Setting this field to `true` would attempt to immediately capture the transaction, otherwise known as a sale.

If the transaction is authorized the response should contain the following field:

```
{
...
"status": "AUTHORIZED"
```

```
...
}
```

The transaction has now been authorized and can be voided or captured.

Void authorization

Action	Void authorization
Description	Voiding an authorized transaction means that the authorization will be reversed, freeing up the reserved funds on the cardholder account.
Link to API docs	https://sandbox.omni.verifone.cloud/docs/api#operation/voidAuthCardTran...
Prerequisite 1	A transaction that can be voided for use in the `id` field. This can be obtained by performing an authorization (see previous step) and using the returned `id` from the transaction in the previous step.
Prerequisite 2	Check the connection page for the acquirer you are using for any exceptions regarding the dynamic descriptor, currency, customer requirements or other requirements.

Voiding authorizations requires you to include the transaction id in the POST URL. The url should be formatted like so: https://sandbox.omni.verifone.cloud `/transaction/id/void` where `id` is replaced with the transaction id.

If the void is accepted by the acquirer the response should contain the following field:

```
{
...
"status": "AUTHORIZATION_VOIDED"
...
}
```

Your authorization has been successfully voided.

Capture

Action	Capture
Description	Capturing a card transaction will deduct the amount from the cardholders account at the issuer. Settlement typically follows within 1-3 days. This will depend on your acquirer.

Action	Capture
Link to API docs	https://sandbox.omni.verifone.cloud/docs/api#operation/captureCardTrans...
Prerequisite 1	A transaction id that can be captured for use in the `id` field. This can be obtained by performing an authorization (see previous step) and using the returned `id` from the transaction in the previous step.
Prerequisite 2	Check the connection page for the acquirer you are using for any exceptions regarding the dynamic descriptor, currency, customer requirements or other requirements.

Capturing transactions requires you to include the transaction id in the POST URL. The url should be formatted like so: `https://sandbox.omni.verifone.cloud/transaction/id/capture` where `id` is replaced with the transaction id.

The body should contain the `amount` you would like to capture on the transaction. It is only possible to capture less than the authorized amount, it is currently not possible to perform multiple captures on the same authorization.

```
{
...
"amount": "1000
...
}
```

If the capture is accepted by the acquirer the response should contain the following field:

```
{
...
"status": "SETTLEMENT_REQUESTED"
...
}
```

The transaction has now been captured. Once the settlement files are received from the acquirer the transaction will change to the status `SETTLEMENT_COMPLETED`. It usually takes 1-3 days before the settlement files are received, this timeframe differs per acquirer.

With some acquirers it is also possible to void the capture request.

Void capture

Action	Void capture

Description	Voiding a captured transaction means that the capture request will be reversed, freeing up the reserved funds on the cardholder account. This is only possible after the transaction reaches the status `SETTLEMENT_REQUESTED` and before it is processed.
Link to API docs	https://sandbox.omni.verifone.cloud/docs/api#operation/voidCaptureCardT...
Prerequisite 1	A transaction that can be voided for use in the `id` field. This can be obtained by performing an authorization (see previous step) and using the returned `id` from the transaction in the previous step.
Prerequisite 2	Check the connection page for the acquirer you are using for any exceptions regarding the dynamic descriptor, currency, customer requirements or other requirements.

Voiding authorizations requires you to include the transaction id in the POST URL. The url should be formatted like so: https://sandbox.omni.verifone.cloud `/transaction/id/void_capture` where `id` is replaced with the transaction id.

If the void is accepted by the acquirer the response should contain the following field:

```
{
...
"status": "SETTLEMENT_CANCELLED"
...
}
```

Your capture has been succesfully voided.

Authorize & Capture (Sale)

Action	**Authorize & Capture**
Description	Directly authorize and capture a transaction.
Link to API docs	https://sandbox.omni.verifone.cloud/docs/api#operation/createCardTransa...
Prerequisite 1	A tokenized card for the `card` field.
Prerequisite 2	An account with a card processor attached to it. The account ID will need to be used for the `account` field.
Prerequisite 3	Check the connection page for the acquirer you are using for any exceptions regarding the dynamic descriptor, currency, customer requirements or other requirements.

Capturing and authorizing a transaction works the same as a regular authorization with one small difference: The `capture_now` flag needs to be set to `true`.

Post the following body to the `/transaction` endpoint:

```
{
"account": "account.id",
"amount": 1000,
"card": "card.id",
"capture_now": true,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
}
```

If the transaction is authorized the response should contain the following field:

```
{
...
"status": "SETTLEMENT_REQUESTED"
...
}
```

Account Verification ($0 Auth)

Action	Account Verification
Description	Verify that customer's card details are valid
Link to API docs	https://sandbox.omni.verifone.cloud/docs/api#operation/createCardTransa...
Prerequisite 1	A tokenized card for the `card` field.
Prerequisite 2	An account with a card processor attached to it. The account ID will need to be used for the `account` field.
Prerequisite 3	Check the connection page for the acquirer you are using for support of Account Verification transactions and any exceptions regarding the dynamic descriptor, currency, customer requirements or other requirements.

Account verification transaction works the same as a regular authorization with one small difference: The `amount` needs to be set to `0`.

Account verification does not have amount, so there is no money to be moved out of the customer's card, which means that `capture_now` **has to be** set to `false`. Attempting to capture a previously processed account verification transaction or sending it as a sale would fail.

Account verification transactions cannot handle 3DS data provided with them.

Post the following body to the `/transaction` endpoint:

```
{
"account": "account.id",
"amount": 0,
"card": "card.id",
"capture_now": false,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
}
```

Credentials on File (COF), storing credentials for a first time

Action	Initial transaction storing credentials for first time
Description	Process a payment, storing the shopper's credentials to be used in the future
Link to API docs	https://sandbox.omni.verifone.cloud/docs/api#operation/createCardTransa...
Prerequisite 1	A tokenized card for the `card` field.
Prerequisite 2	An account with a card processor attached to it. The account ID will need to be used for the `account` field.
Prerequisite 3	Check the connection page for the acquirer you are using for support of Credentials on File transactions and any exceptions regarding the dynamic descriptor, currency, customer requirements or other requirements.

A Credential on File transaction is a transaction in which a cardholder has explicitly authorised a card acceptor to store the cardholder's account information for future use. COF transaction can be sent as Authorisation, Sale or Account Verification. It is also possible to provide 3DS data to it. Flagging a transaction as initial COF is done by providing an additional parameter `recurring` in the createCardTransaction POST. This parameter indicates the type of contract and processing model to be used, which controls the way the stored credentials can be used in the future. The schema for initial COF transaction is as:

```
recurring: {
"contract": "string",
"processing_model": "string"
}
```

After successfully processing such transaction, the response will contain an additional field `reference` under the `recurring` object. This reference needs to be provided for all subsequent transactions in order to use the credentials stored by this initial one. A `reference` will only be generated when the transaction is processed successfully.

Usage

Business model	Shopper present	Transaction	shopper_interaction	contract	processing_model	CVV
Online purchase	yes	Online purchase where shopper agrees to store card details for future use	ecommerce	ONE_CLICK	COF	optional
Subscriptions	yes	Transaction to sign up for a subscription	ecommerce	RECURRING	SUBSCRIPTION	optional
Shopper absent transactions	yes	Transaction to sign up for the terms and conditions of later subsequent charges.	ecommerce	RECURRING	UNSCHEDULED_COF	optional

- Initial COF transactions can also be sent as shopper_interaction set to `mail_order` or `telephone_order`.

Examples

- COF
- SUBSCRIPTION
- UNSCHEDULED_COF

Example 1: initial COF transaction, storing the credentials for future use. Request:

```
{
"account": "account.id",
"amount": 1234,
"card": "card.id",
"capture_now": false,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
```

```
"recurring": {
"contract": "ONE_CLICK",
"processing_model": "COF"
}
}
```

Response:

```
{
"_id": "1234567",
"account": "account.id",
"amount": 1234,
"card": "card.id",
"capture_now": false,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"recurring": {
"contract": "ONE_CLICK",
"processing_model": "COF",
"reference": "1234567"
}
}
```

Example 2: initial SUBSCRIPTION transaction, storing the credentials for future use. Request:

```
{
"account": "account.id",
"amount": 1234,
"card": "card.id",
"capture_now": false,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"recurring": {
"contract": "RECURRING",
"processing_model": "SUBSCRIPTION"
}
}
```

Response:

```
{
"_id": "9876556789",
"account": "account.id",
"amount": 1234,
```

```
"card": "card.id",
"capture_now": false,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"recurring": {
"contract": "RECURRING",
"processing_model": "SUBSCRIPTION",
"reference": "9876556789"
}
}
```

Example 3: initial UNSCHEDULED_COF transaction, storing the credentials for future use. Request:

```
{
"account": "account.id",
"amount": 1234,
"card": "card.id",
"capture_now": false,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"recurring": {
"contract": "RECURRING",
"processing_model": "UNSCHEDULED_COF"
}
}
```

Response:

```
{
"_id": "87654321",
"account": "account.id",
"amount": 1234,
"card": "card.id",
"capture_now": false,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"recurring": {
"contract": "RECURRING",
"processing_model": "UNSCHEDULED_COF",
"reference": "87654321"
}
}
```

Credentials on File (COF), using previously stored credentials

Action	Subsequent transaction with stored credentials
Description	Process a transaction using previously stored shopper credentials
Link to API docs	https://sandbox.omni.verifone.cloud/docs/api#operation/createCardTransa...
Prerequisite 1	A tokenized card for the `card` field.
Prerequisite 2	An account with a card processor attached to it. The account ID will need to be used for the `account` field.
Prerequisite 3	Check the connection page for the acquirer you are using for support of Credentials on File transactions and any exceptions regarding the dynamic descriptor, currency, customer requirements or other requirements.

COF transaction using previously stored credentials works the same way as regular regular Authorisation or Sale. In the case that the shopper is present, 3DS data can also be provided. Flagging a transaction as subsequent COF is done by providing `shopper_interaction` with value `cont_auth` and an additional parameter `recurring` . This parameter is used to indicate the processing model and provide reference, which both need to match the initially processed transaction.

```
{
"shopper_interaction": "cont_auth",
...
"recurring": {
"processing_model": "string",
"reference": "string"
}
}
```

Usage

Business model	Shopper present	Transaction	shopper_interaction	contract	processing_model	CVV
Online purchase	yes	Online purchase where shopper uses previously stored credentials	cont_auth	-	COF	optional

Business model	Shopper present	Transaction	shopper_interaction	contract	processing_model	CVV
Subscriptions	no	Subsequent subscription charge.	cont_auth	-	SUBSCRIPTION	not used
Shopper absent transactions	no	Subsequent charges as agreed upon during the sign-up transaction.	cont_auth	-	UNSCHEDULED_COF	not used

Examples

- Subsequent COF
- Subsequent SUBSCRIPTION
- Subsequent UNSCHEDULED_COF

Example 1: Subsequent COF sale transaction, using previously stored credentials

Subsequent COF transactions indicate that the shopper is present. This means that it is possible to provide CVV and/or 3DS data for this transaction.

(given that an initial COF transaction was successfully processed as in Example 1 of `Credentials on File (COF), storing credentials for a first time`)

Request:

```
{
"account": "account.id",
"amount": 1234,
"card": "card.id",
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"shopper_interaction": "cont_auth",
"recurring": {
"processing_model": "COF",
"reference": "1234567"
}
}
```

Response:

```
{
"_id": "1234567",
"account": "account.id",
"amount": 1234,
"card": "card.id",
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"shopper_interaction": "cont_auth",
"recurring": {
"processing_model": "COF",
"reference": "1234567"
}
}
```

Example 2: Subsequent SUBSCRIPTION sale transaction, using previously stored credentials (given that an initial SUBSCRIPTION transaction was successfully processed as in Example 2 of `Credentials on File (COF), storing credentials for a first time`)

Request:

```
{
"account": "account.id",
"amount": 1234,
"card": "card.id",
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"shopper_interaction": "cont_auth",
"recurring": {
"processing_model": "SUBSCRIPTION",
"reference": "87654321"
}
}
```

Response:

```
{
"_id": "9876556789",
"account": "account.id",
"amount": 1234,
"card": "card.id",
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"shopper_interaction": "cont_auth",
"recurring": {
"processing_model": "SUBSCRIPTION",
```

```
"reference": "9876556789",
}
}
```

Example 3: Subsequent UNSCHEDULED_COF authorisation transaction, using previously stored credentials (given that an initial UNSCHEDULED_COF transaction was successfully processed as in Example 3 of `Credentials on File (COF), storing credentials for a first time`)

Request:

```
{
"account": "account.id",
"amount": 1234,
"card": "card.id",
"capture_now": false,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"shopper_interaction": "cont_auth",
"recurring": {
"processing_model": "UNSCHEDULED_COF",
"reference": "87654321"
}
}
```

Response:

```
{
"_id": "87654321",
"account": "account.id",
"amount": 1234,
"card": "card.id",
"capture_now": false,
"customer_ip": "string",
"dynamic_descriptor": "dynamic_descriptor",
"merchant_reference": "string",
"user_agent": "string",
"shopper_interaction": "cont_auth",
"recurring": {
"processing_model": "UNSCHEDULED_COF",
"reference": "87654321"
}
}
```