API de E-Commerce Pluxee (M-PAL) Autenticação e Serviços

Pluxee / Versão 1.3 | 26/07/2024

Índice

Autenticação – Guia de implementação

pluxee

Introdução

A API de E-Commerce é uma plataforma de e-commerce que fornece uma interface unificada possibilitando que Estabelecimentos e Gateways se conectem e transacionem com a Pluxee.

> **Aviso**
>
> Sempre que o nome técnico **M-PAL** for mencionado neste documento, ele se refere à **API de E-Commerce**, nome comercial da plataforma.

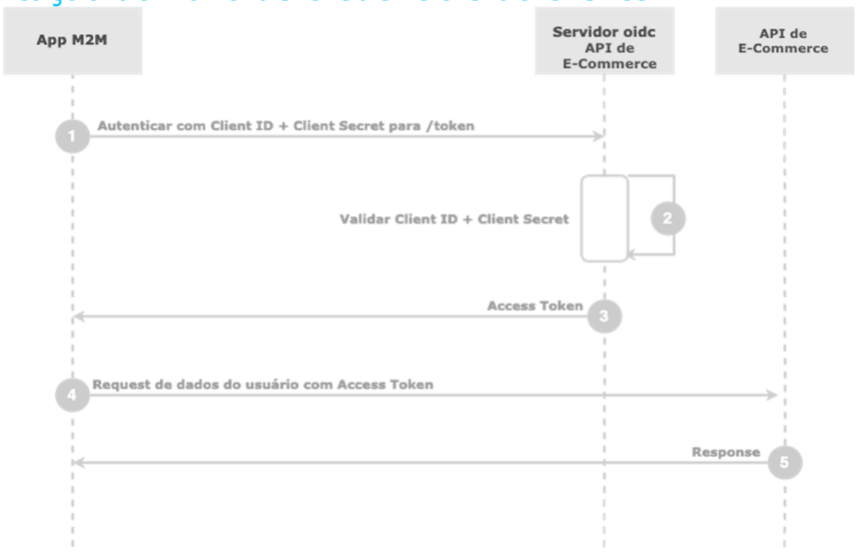
Este guia tem como objetivo descrever passo a passo os devidos procedimentos para a integração bem-sucedida da API de Autenticação.

Métodos de autenticação da aplicação

As aplicações confidenciais, ao contrário das aplicações públicas, armazenam credenciais com segurança. Quando aplicações confidenciais solicitam acesso do endpoint do token, a aplicação deve se autenticar com o servidor de autorização. Durante essa solicitação de tokens, a aplicação fornece credenciais conhecidas por ela. Para obter tokens, sua aplicação deve se autenticar por meio da API de Autenticação da API de E-Commerce Pluxee. Esta realiza a autenticação da sua aplicação utilizando o Client Secret.

Client Secret: A autenticação Client Secret é um método de autenticação simétrica incluído na especificação OAuth2.0. Esse fluxo é mais adequado para aplicativos Machine-to-Machine (M2M), como CLIs, daemons ou serviços de back-end, porque o sistema deve autenticar e autorizar o aplicativo em vez de um usuário.

Implementação do fluxo de credenciais do cliente



Para se integrar com a API de E-Commerce Pluxee será necessário passar por três etapas, conforme descritas abaixo:

1. Inicialmente, você precisa enviar um **request de Client Credential** para:
   UAT --> https://connect.uat.pluxee.app
   PRD --> https://connect.pluxee.app

Detalhes do Request de Credenciais do Cliente

Nome	Descrição
client_id	O identificador único da aplicação na API de E-Commerce Pluxee. É fornecido pela equipe local de Merchant.
client_secret	O identificador Secret da aplicação na API de E-Commerce Pluxee. É fornecido pela equipe local de Merchant.
grant_type	value : client_credentials
resource	O parâmetro "resource" deve conter o URL do recurso protegido. Os URLs de todos os recursos identificados (exceto a API de E-Commerce Pluxee). Lembre-se de que apenas um recurso pode ser chamado por solicitação de credencial de cliente. Um request deve ser enviado para cada recurso.
scope	Se você chamar um recurso protegido por escopos específicos, esse parâmetro deverá conter o(s) escopo(s) que você deseja acessar

Exemplo de Request

```
curl --location --request POST 'https://connect.uat.pluxee.app/op/oidc/token' \
--header 'accept: application/json' \
--header 'content-type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic encodeBase64(client_id:client_secret)' \
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'resource=https://api.brso.sodexonet.com/br/mpal/' \
--data-urlencode 'scope=scopes/card-service/cards/token
scopes/card-service/cards/vault
scopes/transaction-service/transaction''
```

2. No **response**, você receberá um **Access Token** da API de E-Commerce Pluxee. O token recebido deve ser armazenado em um cache até expirar. Não fazer isso pode reduzir o desempenho da API de E-Commerce Pluxee. Portanto, se a cada request for gerado um novo Access Token, o seu Client ID poderá ser bloqueado. A duração máxima recomendada para o cache de token é de 5 minutos, independentemente do tempo de vida do token.

Exemplo de Response

```
{
  "access_token": "U9ZMQVwKYZKap********rXTkUVm8SnvNUH",
  "expires_in": 600,
  "token_type": "Bearer",
}
```

3. Agora, você deve realizar uma **chamada** à API/Servidor que foi passada no parâmetro "resource" com o Access Token recebido. Esse token deve ser passado como **bearer** no header da solicitação HTTP feita à API consumida adicionando o seguinte cabeçalho:

```
--header 'Authorization: Bearer {{access_token}}'
```

Exemplo

```
curl --location '{{mpal root address}}/transaction-service/transaction' \
--header 'Ocp-Apim-Subscription-Key: {{subscriptionKey}}' \
--
header 'Authorization: Bearer eyJhbGciOiJFUzI1NiIsI****************************************a
kJf9f5o7xI1l6p7et-XKQ' \
--header 'Content-Type: application/json' \
--data '{
  "payment": {
    "captureType": "PA",
    "currency": "BRL",
    "amount": 10.58
  },
  "card": {
      "token": {
        "token": "fff951f3-d497-4d2d-915b-3720098dd86c",
        "ownerName": "ELIAS SENA SILVA",
        "cvv": "296",
        "expiryMonth": "05",
        "expiryYear": "2033"
      }
  },
  "merchant": {
    "orderNumber": "0000000000085",
    "mid": "000000001483136"
  },
  "consumer": {
    "documentType": "CPF",
    "documentNumber": "58191828049"
  }
}
```

Serviços - API de E-Commerce Pluxee

pluxee

Introdução

A API de E-Commerce Pluxee é um hub de APIs que permite a conexão com a Pluxee para envio de transações e-commerce. Atualmente, a plataforma possui 2 tipos de serviços principais: Cartões (Cards) e Transação (Transaction).

Aqui serão descritas as funcionalidades e como atuam os serviços da API de E-Commerce Pluxee.

Segue tabela com valores para substituições das variáveis que constam nos exemplos abaixo.

Tabela de Valores em Ambientes UAT e PRD

Variável	UAT	PRD
baseUrl	https://api.brso.uat.sodexonet.com/br/mpal	https://api.brso.sodexonet.com/br/mpal
subscriptionKeyCards	bea7dc34d0f84b10ac162d325a89a9ec	26035f4971094d05b48a04809c3d2052
subscriptionKeyTransaction	c8dc862cbcf24cf4b8ed519f188da2bb	8404ae9d66be472999f3907f4240a627

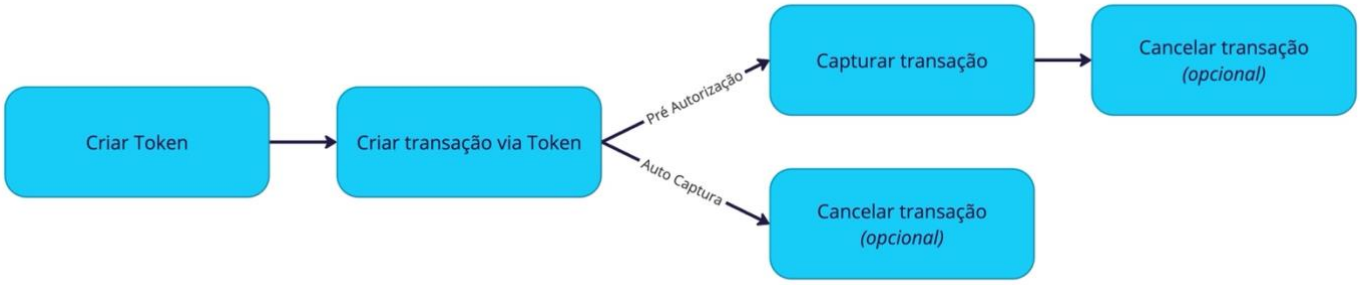
1. Cards

Para realizar uma transação dentro da API de E-Commerce com cartões Pluxee, primeiramente, é necessário tokenizar o número cartão do cliente (PAN) ou criar um cofre (vault) com todos os dados de cartão, para que o mesmo possa ser reutilizado posteriormente de forma mais eficaz. Esse passo existe para aumentarmos o nível de segurança das transações trafegadas.

Abaixo seguem formas de criar o token ou o cofre para iniciar a transação junto à Pluxee.

1.1 Token

O serviço de "tokenização" é usado para adicionar uma camada de segurança para transações realizadas através da API de E-Commerce Pluxee. Ele existe para que não exista o tráfego de todas as informações de cartão de uma só vez, portanto, é necessário que, primeiro, o número do cartão (PAN) seja tokenizado, para depois ser enviado junto com as demais informações do cartão (Data de validade e CVV).



Um token poderá ser utilizado apenas uma vez, após seu uso, mesmo que a transação não seja aprovada, deverá ser gerado um novo token para iniciar uma nova transação.

Além disso o token deverá ser gerado no momento da transação, pois possui um curto tempo de expiração e, após esse período, será inativado.

Exemplo Token: Geração de token

- **Contexto:** O Gateway de Pagamentos / Estabelecimento quer gerar um token de número de cartão
- **Transação:** O Gateway de Pagamentos / Estabelecimento vai chamar o serviço de "tokenização" enviando o número do cartão "6060606060606060". Como resposta ele vai receber um token "514023b7-8ef0-4e21-99ee-60492263d0e6". Posteriormente, esse token será utilizado para ser enviado no Request do Transaction

Request

baseUrl: Segue exemplo abaixo.

Header

X-Mid: Merchant ID que está solicitando a tokenização do cartão.

subscriptionKey: Segue exemplo abaixo.

Body

cardNumber: Número do cartão do portador que deverá ser tokenizado.

Exemplo de Request do Token

```
curl --location '{{baseUrl}}/card-service/cards/token' \
--header 'X-Mid: {{merchantId}} \
--header 'Ocp-Apim-Subscription-Key: {{subscriptionKeyCards}}' \
--header 'Authorization: Bearer {{accessToken}}' \
--header 'Content-Type: application/json' \
--data '{
  "cardNumber": "6060682200477601"
}'
```

Response

Usuário receberá no retorno o token do cartão solicitado e a data de expiração do token.

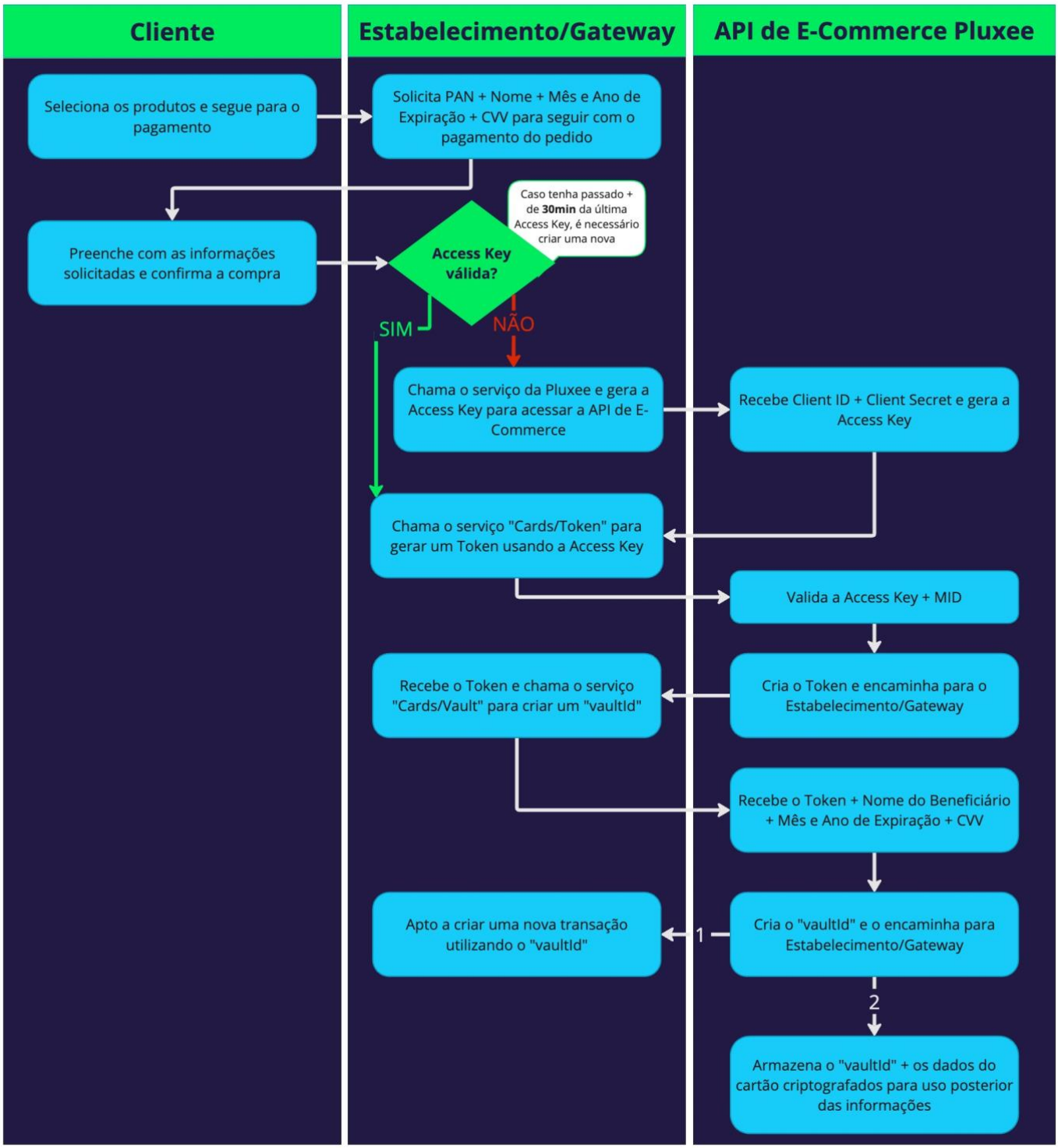
O token recebido na resposta poderá ser utilizado para iniciar uma nova transação ou para gerar um cofre dentro da API de E-Commerce Pluxee.

Exemplo de Response do Token

```
{
    "cardToken": "514023b7-8ef0-4e21-99ee-60492263d0e6",
    "expireAt": "2024-07-03T15:28:12.376166698-03:00"
}
```

1.2 Cofre (Vault)

O serviço de cofre (Vault) armazena os dados do cartão do usuário de maneira segura para permitir transações do tipo One-Click. Com isso, novas transações podem ser feitas com o mesmo cartão sem precisar solicitar os dados do cartão novamente e criar tokens a cada nova compra (será necessário criar um token de cartão somente no processo de criação do cofre).



Exemplo Token: Geração de cofre

- **Contexto:** O Gateway de Pagamentos / Estabelecimento quer gerar um VaultID para os dados de cartão de um usuário.
- **Transação:**
  - O Gateway de Pagamentos / Estabelecimento vai chamar o serviço de "tokenização" enviando o número do cartão "6060606060606060". Como resposta ele vai receber um token "514023b7-8ef0-4e21-99ee-60492263d0e6".
  - Posteriormente, ele precisa chamar o serviço de vault enviando o token gerado anteriormente, junto com o Nome Impresso no Cartão, Data de Validade e CVV. Como resposta, ele vai receber o vaultId "6b5d242a-146d-4267-b8c0-9bf2d8d4646c".
  - Esse mesmo vaultId pode ser utilizado sempre esse usuário fizer uma compra no e-commerce.

Request

baseUrl: Segue exemplo abaixo.

Header

X-Mid: Merchant ID que está solicitando a criação do cofre.
subscriptionKey: Segue exemplo abaixo.

Body

cardToken: Token recebido no retorno da API de tokenização.
Demais campos são obtidos no preenchimento do portador do cartão.

Exemplo de Request do Vault

```
curl --location '{{baseUrl}}/card-service/cards/vault' \
--header 'X-Mid: {{merchantId}}\
--header 'Ocp-Apim-Subscription-Key: {{subscriptionKeyCards}}' \
--header 'Authorization: Bearer {{accessToken}}' \
--header 'Content-Type: application/json' \
--data '
{
  "cardToken": "514023b7-8ef0-4e21-99ee-60492263d0e6",
  "ownerName": "TESTE MIGRACAO 3C 15",
  "expiryMonth": "09",
  "expiryYear": "2033",
  "cvv": "421"
}'
```

Response

vaultId: Campo que deverá ser utilizado para iniciar uma nova transação.

Exemplo de Response do Vault

```
{
    "vaultId": "6b5d242a-146d-4267-b8c0-9bf2d8d4646c"
}
```

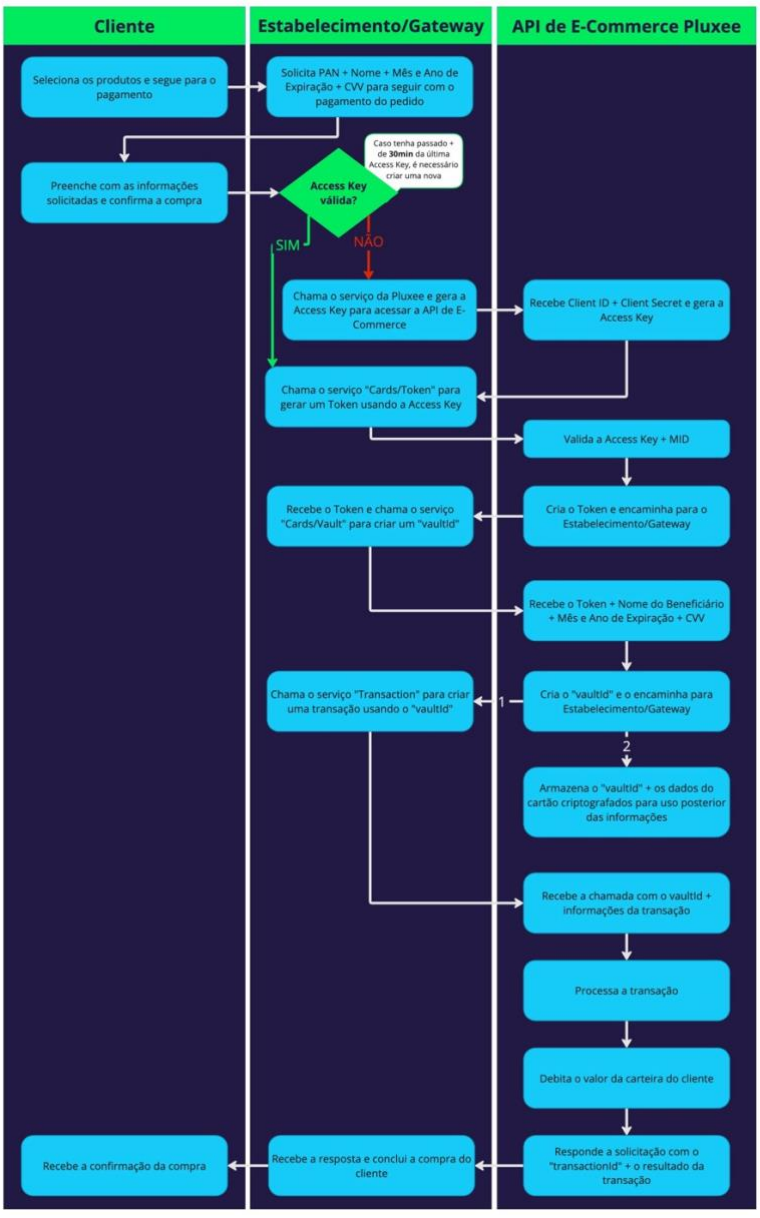
2. Transaction

Conjunto de APIs para enviar transações (criar), capturar, cancelar ou consultar um pagamento.

2.1 Criar

O serviço de criação é utilizado para criar transações de pagamento. Existem dois tipos de transações que podem ser enviadas:

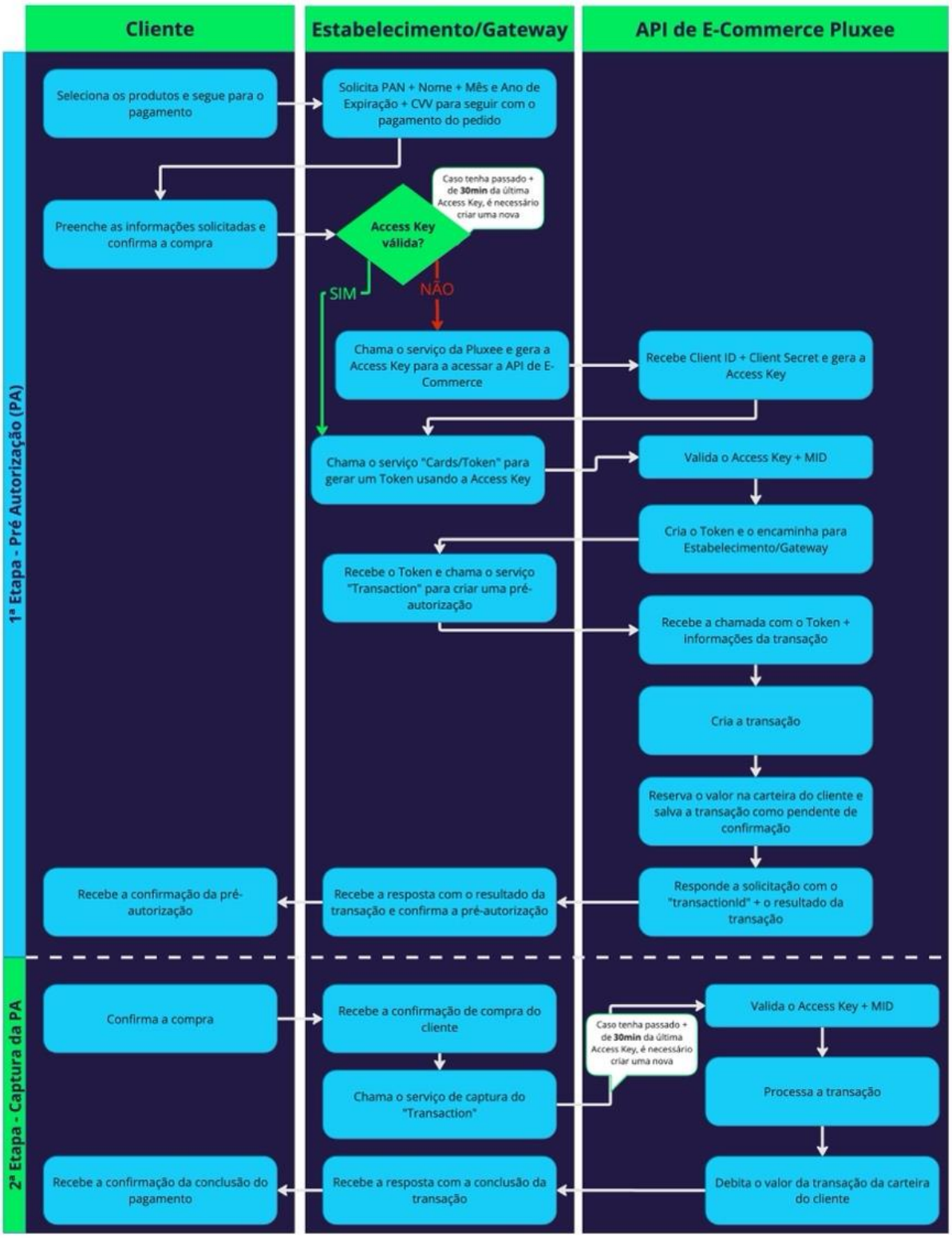
- **Captura Imediata (AC):** Transação de compra/venda comum, nesse caso, o saldo do portador do cartão será sensibilizado imediatamente;



Exemplo Captura Imediata: Compra de Almoço

- **Contexto:** Um usuário utiliza um aplicativo de entrega de comida para pedir o almoço.
- **Transação:** O usuário seleciona os itens do cardápio e paga com seu vale-refeição. Assim que a compra é confirmada, o saldo do vale-refeição é imediatamente debitado do valor da compra.

- **Pré-Autorização (PA):** Transação de duas etapas. Inicialmente é só reservado o saldo da conta do portador do cartão, porém, essa transação precisa de uma confirmação (Captura) para ser finalizada.



Exemplo: Pedido de Supermercado

- **Contexto:** Um usuário faz um pedido de compras de supermercado através de um aplicativo de mercado online. Ele decide pagar com seu vale-alimentação.
- **Transação:**
  - **Etapa 1: Pré-Autorização**
    - O usuário seleciona todos os itens que deseja comprar no aplicativo, como frutas, vegetais, produtos de limpeza etc.
    - O aplicativo calcula o valor total dos itens selecionados e realiza uma pré-autorização no vale-alimentação do usuário. Por exemplo, o valor total dos itens é R$ 200,00.
    - O aplicativo reserva o valor de R$ 200,00 no saldo do vale alimentação do usuário. Isso garante que o valor necessário está disponível para a compra, mas ainda não é debitado da conta do usuário.
  - **Etapa 2: Captura**
    - Os itens são separados e embalados pelo supermercado.
    - Quando o pedido está pronto para ser entregue, o supermercado confirma o valor final do pedido, que pode variar ligeiramente devido a peso de produtos frescos ou disponibilidade de itens.
    - O valor final confirmado é R$ 195,00.
    - O aplicativo então realiza a captura do valor de R$ 195,00, debitando esse montante do saldo do vale alimentação do usuário.
    - A diferença de R$ 5,00 que foi inicialmente pré-autorizada, mas não utilizada,

O Estabelecimento/Gateway de Pagamentos enviará as transações conforme descrição abaixo:

Request

baseUrl: Segue exemplo abaixo.

Header

subscriptionKey: Segue exemplo abaixo.

Body

payment

captureType: "AC" para Auto Captura ou "PA" para Pré-Autorização.

currency: Moeda da transação, para o mercado brasileiro enviar BRL.

amount: Valor da transação em reais.

card: Dentro do card deverá ser enviado o schema vault ou o schema token. Os schemas são condicionalmente exclusivos.

vault

vaultId: ID do vault criado anteriormente no serviço de cofre.

token

token: Token do cartão gerado no serviço de tokenização de cartão.

Demais campos obtidos no preenchimento do portador do cartão.

merchant

orderNumber: Número do pedido, cada pedido deverá ter um número único e esse campo não deverá se repetir no mesmo dia.

mid: Merchant ID do solicitante da transação.

consumer

documentType: Tipo de documento do portador (CPF ou CNPJ)

documentNumber: Número do documento do portador.

Exemplos de Request da Transação

1. Pré-Autorização via Token

```
curl --location '{{baseUrl}}/transaction-service/transaction' \
--header 'Ocp-Apim-Subscription-Key: {{subscriptionKeyTransaction}}’ \
--header 'Authorization: Bearer {{accessToken}}' \
--header 'Content-Type: application/json' \
--data '{
  "payment": {
    "captureType": "PA",
    "currency": "BRL",
    "amount": 0.19
  },
  "card": {
      "token": {
        "token": "f44fe30a-ddf8-41be-96bd-949319c9889e",
        "ownerName": "USER CVV HML726",
        "cvv": "493",
        "expiryMonth": "05",
        "expiryYear": "2032"

      }
  },
  "merchant": {
    "orderNumber": "0000000000011",
    "mid": "{{merchantId}}"
  },
  "consumer": {
    "documentType": "CPF",
    "documentNumber": "24544784123"

  }
}'
```

2. Auto Captura via Token

```
curl --location '{{baseUrl}}/transaction-service/transaction' \
--header 'Ocp-Apim-Subscription-Key: {{subscriptionKeyTransaction}}’ \
--header 'Authorization: Bearer {{accessToken}}' \
--header 'Content-Type: application/json' \
--data '{
  "payment": {
    "captureType": "AC",
    "currency": "BRL",
    "amount": 0.2
  },
  "card": {
      "token": {
        "token": "b2b62a05-26b4-4ae9-835a-efcfa46bdacc",
        "ownerName": "USER CVV HML726",
```

```
        "cvv": "493",
        "expiryMonth": "05",
        "expiryYear": "2032"
      }
  },
  "merchant": {
    "orderNumber": "0000000000046",
    "mid": "{{merchantId}}"
  },
  "consumer": {
    "documentType": "CPF",
    "documentNumber": "24544784123"
  }
}'
```

3. Pré-Autorização via Vault

```
curl --location '{{baseUrl}}/transaction-service/transaction' \
--header 'Ocp-Apim-Subscription-Key: {{subscriptionKeyTransaction}}’ \
--header 'Authorization: Bearer {{accessToken}}' \
--header 'Content-Type: application/json' \
--data '{
  "payment": {
    "captureType": "PA",
    "currency": "BRL",
    "amount": 0.2
  },
  "card": {
      "vault": {
        "vaultId": "d4bfa0f8-575b-4ff9-97b4-357f526a2e94"
      }
  },
  "merchant": {
    "orderNumber": "0000000000046",
    "mid": "{{merchantId}}"
  },
  "consumer": {
    "documentType": "CPF",
    "documentNumber": "24544784123"
  }
}'
```

4. Auto Captura via Vault

```
curl --location '{{baseUrl}}/transaction-service/transaction' \
--header 'Ocp-Apim-Subscription-Key: {{subscriptionKeyTransaction}}' \
--header 'Authorization: Bearer {{accessToken}}' \
--header 'Content-Type: application/json' \
--data '{
  "payment": {
    "captureType": "AC",
    "currency": "BRL",
```

```
    "amount": 0.64
  },
  "card": {
      "vault": {
        "vaultId": "cb0ad202-147a-48b5-8b95-ef8f1b8c58a3"
      }
  },
  "merchant": {
    "orderNumber": "0000000000008",
    "mid": "{{merchantId}}"
  },
  "consumer": {
    "documentType": "CPF",
    "documentNumber": "24544784123"
  }
}'
```

Response

transactionId: ID da transação, que poderá ser utilizado posteriormente para Captura, Cancelamento ou Consulta da transação.

Exemplos de Response da Transação

1. Pré-Autorização

```
{
    "responseCode": "MPAL-TRANSACTION-0000",
    "responseDescription": "Sucess",
    "authorizationCode": "700181",
    "transactionId": "0100036456070315472070018 1ECOM00110000000001484902",
    "amount": 0.19,
    "status": "AWAITING_CAPTURE",
    "receivedAt": "2024-07-03T15:47:20-03:00"
}
```

2. Auto Captura

```
{
    "responseCode": "MPAL-TRANSACTION-0000",
    "responseDescription": "Sucess",
    "authorizationCode": "700176",
    "transactionId": "0200036451070315391970017 6ECOM00110000000001484902",
    "amount": 0.64,
    "status": "CAPTURED",
    "receivedAt": "2024-07-03T15:39:19-03:00"
}
```

2.2 Capturar (2ª Etapa da Pré-Autorização)

O serviço de captura é um endpoint usado para capturar o valor da transação e confirmar o pagamento (utilizado somente para casos de Pré-Autorização).

Fluxo de Captura da Pré-Autorização

Request

baseUrl: Segue exemplo abaixo.

Header

subscriptionKey: Segue exemplo abaixo.

Path

transactionId: ID da transação obtido no response da transação de Pré-Autorização.

Body

mid: Merchant ID do solicitante da captura.

amount: Valor que deverá ser confirmado (pode ser menor ou igual ao valor da Pré-Autorização).

Exemplo de Request de Captura da Pré-Autorização

```
curl --location '{{baseUrl}}/mpal/transaction-service/transaction/01000364560703154720700181ECOM001100000000001484902/capture' \
--header 'Ocp-Apim-Subscription-Key: {{subscriptionKeyTransaction}}’ \
--header 'Authorization: Bearer {{accessToken}}' \
--header 'Content-Type: application/json' \
--data '{
  "mid": "{{merchantId}}",
  "amount": 0.17
}'
```

Response

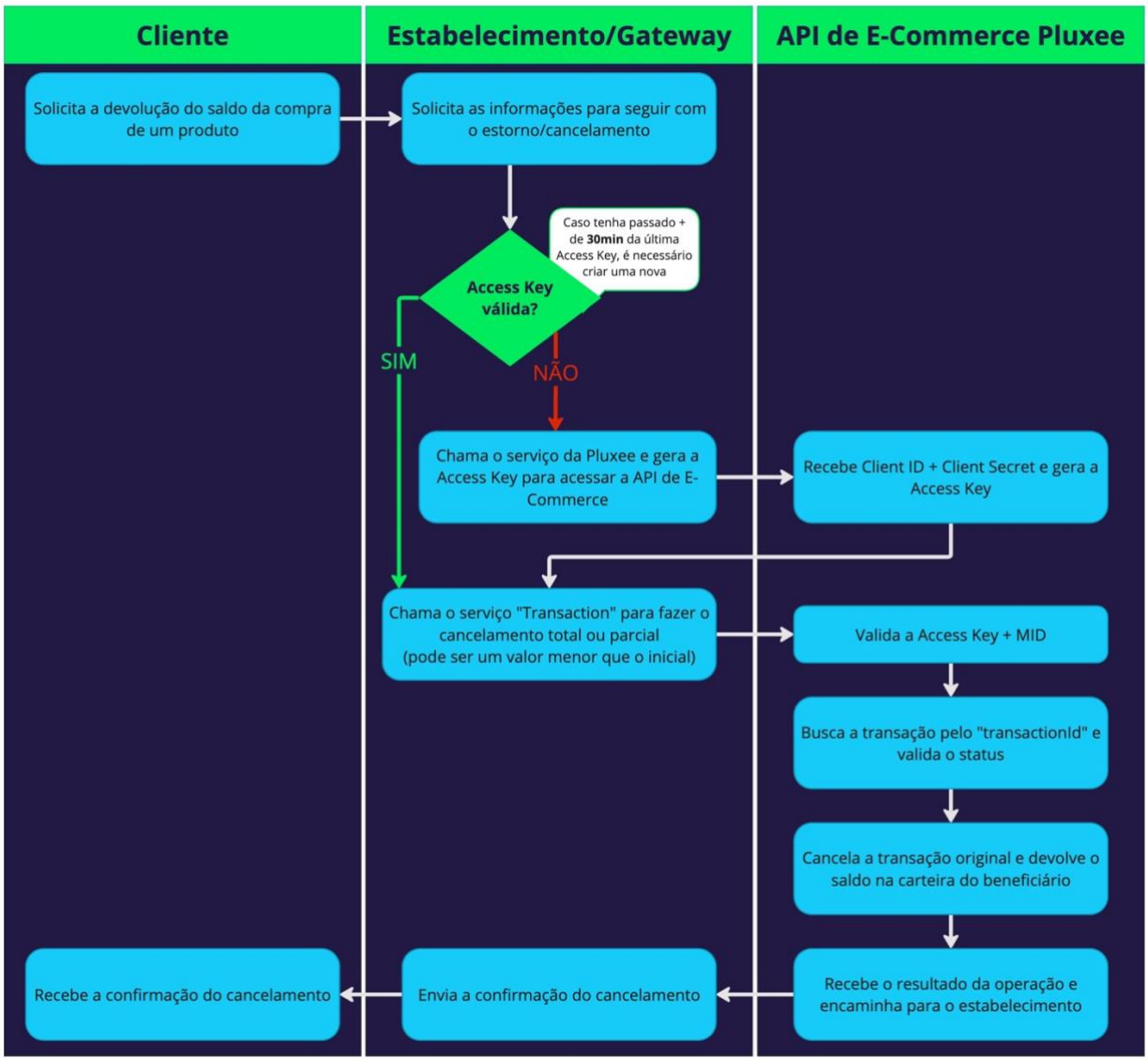
transactionId: ID da transação, que poderá ser utilizado posteriormente para o cancelamento da transação.

Exemplo de Response de Captura da Pré-Autorização


```
{
    "responseCode": "MPAL-TRANSACTION-0000",
    "responseDescription": "Sucess",
    "authorizationCode": "700181",
    "transactionId": "01000364560703154720700181ECOM001100000000001484902",
    "amount": 0.17,
    "status": "CAPTURED",
    "receivedAt": "2024-07-03T15:47:20-03:00"
}
```

2.3 Cancelar

O serviço de cancelamento é um endpoint usado para cancelar ou reverter uma transação.



Request

baseUrl: Segue exemplo abaixo.

Header

subscriptionKey: Segue exemplo abaixo.

Path

transactionId: ID da transação obtido no response da transação de Pré-Autorização ou de Auto-Captura.

Body

mid: Merchant Id do solicitante da captura.

amount: Valor que deverá ser confirmado (pode ser menor ou igual ao valor da Pré-Autorização ou Auto-Captura)

Exemplo de Request do Cancelamento

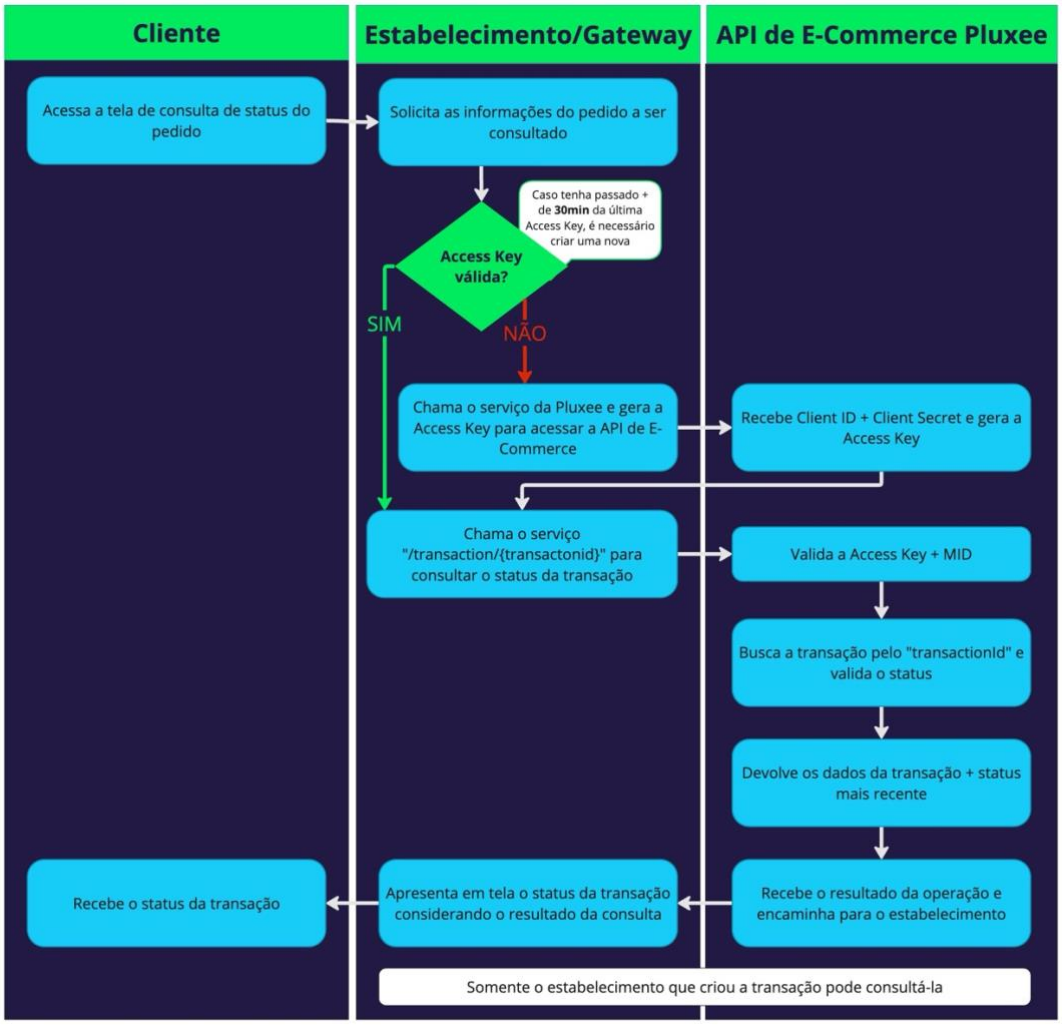
```
curl --location '{{baseUrl}}/transaction-
service/transaction/02000364510703153919700176ECOM00110000000001484902/cancel' \
--header 'Ocp-Apim-Subscription-Key: {{subscriptionKeyTransaction}}' \
--header 'Authorization: Bearer {{accessToken}}' \
--header 'Content-Type: application/json' \
--data '{
  "mid": "{{merchantId}}",
  "amount": 0.64
}'
```

Exemplo de Response do Cancelamento


```
{
    "responseCode": "MPAL-TRANSACTION-0000",
    "responseDescription": "APROVADO",
    "authorizationCode": "700176",
    "transactionId": "02000364510703153919700176ECOM00110000000001484902",
    "amount": 0.64,
    "status": "CANCELED",
    "receivedAt": "2024-07-03T15:41:49-03:00"
}
```


2.4 Consultar

O serviço de consulta é usado para consultar uma transação realizada anteriormente e suas respectivas informações.



Request

baseUrl: Segue exemplo abaixo.

Header

subscriptionKey: Segue exemplo abaixo.

Path

transactionId: ID da transação obtido no response da transação de pagamento, captura ou cancelamento.

Exemplo de Request da Consulta

```
curl --location '{{baseUrl}}/transaction-
service/transaction/01000364560703154720700181ECOM00110000000001484902' \
--header 'Ocp-Apim-Subscription-Key: {{subscriptionKeyTransaction}}' \
--header 'Authorization: Bearer {{accessToken}}'
```

Exemplo de Response da Consulta


```
{
    "responseCode": "MPAL-TRANSACTION-0000",
    "responseDescription": "Sucess",
    "authorizationCode": "700181",
    "transactionId": "01000364560703154720700181ECOM00110000000001484902",
    "authorizedAmount": 0.19,
    "capturedAmount": 0.17,
    "canceledAmount": null,
    "status": "CAPTURED",
    "receivedAt": "2024-07-03T15:47:20-03:00",
    "expireAt": "2024-07-10T15:47:20-03:00"
}
```


Códigos e Mensagens de Resposta

Código	Mensagem	JSON (Exemplo)
200	Process OK	{ "code": "MPAL-TRANSACTION-0000", "authorizationCode": "123456", "transactionId": "02000154031114131814000002225779000000000", "amount": 100.55, "status": "CAPTURED", "receivedAt": "2023-01-20T18:00:01Z" }
400	Bad Request (Solicitação Inválida)	{ "code": "MPAL-TRANSACTION-1400", "message": "Error response message, related to the validation or generic errors.", "details": [{ "amount": "not null" }], "releaseAt": "2023-01-01T18:00:01.000Z" }
401	Not Authorized (Não Autorizado)	{ "code": "MPAL-TRANSACTION-1401", "message": "Not Authorized", "details": [], "releaseAt": "2023-01-01T18:00:01.000Z" }

403	Access Denied
	(Acesso Negado)
		<pre>{
		  "code": "MPAL-TRANSACTION-1403",
		  "message": "Access Denied",
		  "details": [],
		  "releaseAt": "2023-01-01T18:00:01.000Z"
		}</pre>

422	Unprocessable Entity
	(Entidade Não Processável)
		<pre>{
		  "code": "MPAL-TRANSACTION-1051",
		  "message": "Insufficient Funds.",
		  "details": [],
		  "date": "2023-01-01T18:00:01Z"
		}</pre>

500	Transaction Rejected
	(Transação Rejeitada)
		<pre>{
		    "code": "MPAL-TRANSACTION-1012",
		    "message": "Transaction rejected",
		    "details": [],
		    "date": "2024-06-05T11:51:50.786211399-03:00"
		}</pre>

Código de Transação	Mensagem
MPAL-TRANSACTION-0000	Success
MPAL-TRANSACTION-1009	Try again
MPAL-TRANSACTION-1082	Invalid cvv
MPAL-TRANSACTION-1005	Merchant not found
MPAL-TRANSACTION-1129	Invalid security code indicator
MPAL-TRANSACTION-1012	Invalid transaction
MPAL-TRANSACTION-1001	Transaction not found
MPAL-TRANSACTION-1015	Invalid data
MPAL-TRANSACTION-1016	Invalid input
MPAL-TRANSACTION-1051	Not enough funds
MPAL-TRANSACTION-1017	Day limit exceeded
MPAL-TRANSACTION-1014	Invalid card
MPAL-TRANSACTION-1018	Card blocked
MPAL-TRANSACTION-1019	Account does not exist
MPAL-TRANSACTION-1054	Expired card
MPAL-TRANSACTION-1004	Invalid merchant
MPAL-TRANSACTION-1020	Card problem
MPAL-TRANSACTION-1013	Invalid amount
MPAL-TRANSACTION-1011	Transaction has already been cancelled
MPAL-TRANSACTION-1021	Invalid message
MPAL-TRANSACTION-1003	Invalid document number
MPAL-TRANSACTION-1024	Wrong expiration date
MPAL-TRANSACTION-1026	Transaction has already been captured

Fim do documento