Payment Methods API

Learn more about the API that powers a range of global payment methods.

The Payment Methods API allows you to accept a variety of payment methods through a single API. A **PaymentMethod** object contains the payment method details to create payments. With the Payment Methods API, you can combine a **PaymentMethod**:

- With a **PaymentIntent** to accept a payment
- With a **SetupIntent** and a **Customer** to save payment details for later

Supported payment methods

To determine which payment methods to use for specific locales, see the **guide to payment methods**.

The guide includes available payment methods for different regions, a detailed description of each payment method's characteristics, and the **geographic regions** where they are most relevant. You can enable any payment methods available to you in the **Dashboard**. Activation is generally instantaneous and does not require additional contracts.

Customer actions

Some payment methods require your customer to take additional steps to complete the payment. The PaymentIntent object's `next_action` parameter specifies the type of customer action.

Some common actions that customers need to perform are:

- Redirect to their bank's online service to authenticate and approve the payment.
- Verify ownership of their account by providing a one-time code that you post to the Stripe API (for example, microdeposits).
- Push funds (for example, in the case for bank transfers) through their bank's online service.

Not all payment methods require additional customer actions. For example, card payments (excluding 3D Secure) require no additional authentication beyond collecting card details.

> **Note**
> For payment methods that require customer action, listen to **webhooks** for notifications on whether a payment has succeeded or not.

Immediate or delayed notification of payment success

Some payment methods immediately return payment status when a transaction is attempted (for example, card payments) but other methods have a delay such as ACH debits. For those that immediately return payment status, the PaymentIntent status either changes to `succeeded` or `requires_payment_method`. A status of `succeeded` guarantees that you will receive the funds from your customers.

Payment methods with delayed notification can't guarantee payment during the delay. The status of the PaymentIntent object will be `processing` until the payment status is either successful or failed. It's common for businesses to hold an order in a *pending* state during this time, not fulfilling the order until the payment is successful.

> **Note**
> For payment methods with delayed notification, listen to **webhooks** for notifications on whether a payment has succeeded or not.

Single-use or reusable

You can reuse certain payment methods (for example, cards or bank debits) for additional payments without authorizing and collecting payment details again.

You should always set up reusable payment methods for future use to reduce the chance of future declines and payment friction (such as

Single-use payment methods (for example, some kinds of bank transfers) can't be attached to customers because they're consumed after a payment attempt.

Use webhooks to track payment status

Use **webhooks** for payment methods that either require customer action or when payment notification is delayed. Stripe sends the following events when the `PaymentIntent` status is updated:

EVENT	DESCRIPTION	NEXT STEPS
`payment_intent.processing`	The customer's payment was submitted to Stripe successfully. Only applicable to payment methods with **delayed notification**.	Wait for the initiated payment to succeed or fail.
`payment_intent.succeeded`	The payment succeeded.	Fulfill the purchased goods or services.
`payment_intent.payment_failed`	The payment failed.	Send an email or push notification to request another payment method.

You can also use the following options instead of building a webhook handler to listen to events:

- Manually track the status of payments in the Stripe Dashboard, if your business accepts a low volume of orders from payment methods with delayed notification. The Dashboard allows you to **view all your Stripe payments**, send email receipts, handle payouts, or retry failed payments.
- Use polling (for example, repeatedly retrieving a PaymentIntent so that you can check its status). Note that polling is significantly less reliable and may not work at scale. Stripe enforces rate limiting on API requests, so exercise caution if you use polling.
- Use a partner application to handle common business events, like **shipping** and **inventory management**.

The PaymentMethod object

A PaymentMethod contains reusable payment method details for creating payments (for example, card expiration date or billing address), it doesn't include transaction-specific information (for example, amount, currency). A PaymentMethod is attached to a PaymentIntent to represent the **states of a payment lifecycle**. Each PaymentMethod has a **type attribute** (for example, `"type": "sepa_debit"`) and an additional hash whose name matches the type and contains information specific to the PaymentMethod type (for example, `"sepa_debit": {}`). Example of a `sepa_debit` PaymentMethod object:

```
{
  "id": "pm_123456789",
  "object": "payment_method",
  "billing_details": {
    "address": {...},
    "email": "jenny@example.com",
    "name": "Jenny Rosen",
    "phone": "+335555555555"
  },
  "sepa_debit": {
    "bank_code": "37040044",
    "branch_code": "94832",
    "country": "FR",
    "fingerprint": "ygEJfUjzWMGyWnZg",
    "last4": "3000"
  },
  "type": "sepa_debit",
  (...)
}
```

> **Note**
> To safely handle sensitive payment information and automatically handle customer actions, Stripe recommends that you create payment methods using Stripe.js

See also

- Guide to Payment Methods
- Payment Methods API reference