

API Reference

The Stripe API is organized around **REST**. Our API has predictable resource-oriented URLs, accepts **form-encoded** request bodies, returns **JSON-encoded** responses, and uses standard HTTP response codes, authentication, and verbs.

You can use the Stripe API in test mode, which doesn't affect your live data or interact with the banking networks. The API key you use to **authenticate** the request determines whether the request is live mode or test mode.

The Stripe API differs for every account as we release new **versions** and tailor functionality. Log in to see docs customized to your version of the API, with your test key and data.

Authentication

The Stripe API uses **API keys** to authenticate requests. You can view and manage your API keys in [the Stripe Dashboard](#).

Test mode secret keys have the prefix `sk_test_` and live mode secret keys have the prefix `sk_live_`. Alternatively, you can use **restricted API keys** for granular permissions.

Your API keys carry many privileges, so be sure to keep them secure! Do not share your secret API keys in publicly accessible areas such as GitHub, client-side code, and so forth.

Authentication to the API is performed via **HTTP Basic Auth**. Provide your API key as the basic auth username value. You do not need to provide a password.

If you need to authenticate via bearer auth (e.g., for a cross-origin request), use `-H "Authorization: Bearer sk_test_4eC39HqLyjWDarjtT1zdp7dc"` instead of `-u sk_test_4eC39HqLyjWDarjtT1zdp7dc`.

All API requests must be made over **HTTPS**. Calls made over plain HTTP will fail. API requests without authentication will also fail.

Related video: [Authentication](#).

Connected Accounts

Clients can make requests as connected accounts using the special header `Stripe-Account` which should contain a Stripe account ID (usually starting with the prefix `acct_`).

See also [Making API calls for connected accounts](#).

Errors

Stripe uses conventional HTTP response codes to indicate the success or failure of an API request. In general: Codes in the `2xx` range indicate success. Codes in the `4xx` range indicate an error that failed given the information provided (e.g., a required parameter was omitted, a charge failed, etc.). Codes in the `5xx` range indicate an error with Stripe's servers (these are rare).

Some `4xx` errors that could be handled programmatically (e.g., a card is [declined](#)) include an [error code](#) that briefly explains the error reported.

Attributes

type string

The type of error returned. One of `api_error`, `card_error`, `idempotency_error`, or `invalid_request_error`

code string

For some errors that could be handled programmatically, a short string indicating the [error code](#) reported.

decline_code string

For card errors resulting from a card issuer decline, a short string indicating the [card issuer's reason for the decline](#) if they provide one.

message string

A human-readable message providing more details about the error. For card errors, these messages can be shown to your users.

param string

If the error is parameter-specific, the parameter related to the error. For example, you can use this to display a message near the correct form field.

payment_intent hash

The PaymentIntent object for errors returned on a request involving a PaymentIntent.

More attributes

✓ **charge** string

For card errors, the ID of the failed charge.

✓ **doc_url** string

A URL to more information about the [error code](#) reported.

✓ **payment_method** hash

The PaymentMethod object for errors returned on a request involving a PaymentMethod.

✓ **payment_method_type** string

If the error is specific to the type of payment method, the payment method type that had a problem. This field is only populated for invoice-related errors.

✓ **setup_intent** hash

The SetupIntent object for errors returned on a request involving a SetupIntent.

✓ **source** hash

The source object for errors returned on a request involving a source.

Handling errors

Our Client libraries raise exceptions for many reasons, such as a failed charge, invalid parameters, authentication errors, and network unavailability. We recommend writing code that gracefully handles all possible API exceptions.

Related guide: [Error Handling](#).

Expanding Responses

Many objects allow you to request additional information as an expanded response by using the `expand` request parameter. This parameter is available on all API requests, and applies to the response of that request only. Responses can be expanded in two ways.

In many cases, an object contains the ID of a related object in its response properties. For example, a `Charge` may have an associated `Customer` ID. Those objects can be expanded inline with the `expand` request parameter. ID fields that can be expanded into objects are noted in this documentation with the `expandable` label.

In some cases, such as the Issuing Card object's `number` and `cvc` fields, there are available fields that are not included in responses by default. You can request these fields as an expanded response by using the `expand` request parameter. Fields that can be included in an expanded response are noted in this documentation with the `expandable` label.

You can expand recursively by specifying nested fields after a dot (`.`). For example, requesting `invoice.subscription` on a charge will expand the `invoice` property into a full Invoice object, and will then expand the `subscription` property on that invoice into a full Subscription object.

You can use the `expand` param on any endpoint which returns expandable fields, including list, create, and update endpoints.

Expansions on list requests start with the `data` property. For example, you would expand `data.customers` on a request to list charges and associated customers. Many deep expansions on list requests can be slow.

Expansions have a maximum depth of four levels (so for example, when listing charges, `data.invoice.subscription.default_source` is the deepest allowed).

You can expand multiple objects at once by identifying multiple items in the `expand` array.

Related Guide: [Expanding responses](#)

Related video: [Expand](#).

Idempotent Requests

The API supports `idempotency` for safely retrying requests without accidentally performing the same operation twice. This is useful when an API call is disrupted in transit and you do not receive a response. For example, if a request to [create a charge](#) does not respond due to a network connection error, you can retry the request with the same idempotency key to guarantee that no more than one charge is created.

To perform an idempotent request, provide an additional `Idempotency-Key: <key>` header to the request.

Stripe's idempotency works by saving the resulting status code and body of the first request made for any given idempotency key, regardless of whether it succeeded or failed. Subsequent requests with the same key return the same result, including `500` errors.

An idempotency key is a unique value generated by the client which the server uses to recognize subsequent retries of the same request. How you create unique keys is up to you, but we suggest using V4 UUIDs, or another random string with enough entropy to avoid collisions. Idempotency keys can be up to 255 characters long.

Keys are eligible to be removed from the system automatically after they're at least 24 hours old, and a new request is generated if a key is reused after the original has been pruned. The idempotency layer compares incoming parameters to those of the original request and errors unless they're the same to prevent accidental misuse.

Results are only saved if an API endpoint started executing. If incoming parameters failed validation, or the request conflicted with another that was executing concurrently, no idempotent result is saved because no API endpoint began execution. It is safe to retry these requests.

All `POST` requests accept idempotency keys. Sending idempotency keys in `GET` and `DELETE` requests has no effect and should be avoided, as these requests are idempotent by definition.

Related video: [Idempotency and retries](#).

Metadata

Updateable Stripe objects—including [Account](#), [Charge](#), [Customer](#), [PaymentIntent](#), [Refund](#), [Subscription](#), and [Transfer](#)—have a `metadata` parameter. You can use this parameter to attach key-value data to these Stripe objects.

You can specify up to 50 keys, with key names up to 40 characters long and values up to 500 characters long.

Metadata is useful for storing additional, structured information on an object. As an example, you could store your user's full name and corresponding unique identifier from your system on a Stripe [Customer](#) object. Metadata is not used by Stripe—for example, not used to authorize or decline a charge—and won't be seen by your users unless you choose to show it to them.

Some of the objects listed above also support a `description` parameter. You can use the `description` parameter to annotate a charge—with, for example, a human-readable description like `2 shirts for test@example.com`. Unlike `metadata`, `description` is a single string, and your users may see it (e.g., in email receipts Stripe sends on your behalf).

Do not store any sensitive information (bank account numbers, card details, etc.) as metadata or in the `description` parameter.

Related video: [Metadata](#).

Sample metadata use cases

Link IDs

Attach your system's unique IDs to a Stripe object, for easy lookups. For example, add your order number to a charge, your user ID to a customer or recipient, or a unique receipt number to a transfer.

Refund papertrails

Store information about why a refund was created, and by whom.

Customer details

Annotate a customer by storing an internal ID for your later use.

Pagination

All top-level API resources have support for bulk fetches via "list" API methods. For instance, you can [list charges](#), [list customers](#), and [list invoices](#). These list API methods share a common structure, taking at least these three parameters: `limit`, `starting_after`, and `ending_before`.

Stripe's list API methods utilize cursor-based pagination via the `starting_after` and `ending_before` parameters. Both parameters take an existing object ID value (see below) and return objects in reverse chronological order. The `ending_before` parameter returns objects listed before the named object. The `starting_after` parameter returns objects listed after the named object. These parameters are mutually exclusive -- only one of `starting_after` or `ending_before` may be used.

Our client libraries offer [auto-pagination](#) helpers to easily traverse all pages of a list.

Related video: [Pagination and auto-pagination](#).

Parameters

`limit` optional, default is 10

A limit on the number of objects to be returned, between 1 and 100.

`starting_after` optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

ending_before optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

List Response Format

object string, value is "list"

A string describing the object type returned.

data array

An array containing the actual response elements, paginated by any request parameters.

has_more boolean

Whether or not there are more elements available after this set. If `false`, this set comprises the end of the list.

url string

The URL for accessing this list.

Search

Some top-level API resource have support for retrieval via "search" API methods. For example, you can [search charges](#), [search customers](#), and [search subscriptions](#).

Stripe's search API methods utilize cursor-based pagination via the `page` request parameter and `next_page` response parameter. For example, if you make a search request and receive `"next_page": "pagination_key"` in the response, your subsequent call can include `page=pagination_key` to fetch the next page of results.

Our client libraries offer [auto-pagination](#) helpers to easily traverse all pages of a search result.

Search request format

query REQUIRED

The search query string. See [search query language](#).

limit optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

page optional

A cursor for pagination across multiple pages of results. Don't include this parameter on the first call. Use the `next_page` value returned in a previous response to request subsequent results.

Search response format

object string, value is "search_result"

A string describing the object type returned.

url string

The URL for accessing this list.

has_more boolean

Whether or not there are more elements available after this set. If `false`, this set comprises the end of the list.

data array

An array containing the actual response elements, paginated by any request parameters.

next_page string

A cursor for use in pagination. If `has_more` is true, you can pass the value of `next_page` to a subsequent call to fetch the next page of results.

total_count optional positive integer or zero EXPANDABLE

The total number of objects that match the query, only accurate up to 10,000. This field is not included by default. To include it in the response, [expand](#) the `total_count` field.

Auto-pagination

Our libraries support auto-pagination. This feature easily handles fetching large lists of resources without having to manually paginate results and perform subsequent requests.

Request IDs

Each API request has an associated request identifier. You can find this value in the response headers, under `Request-Id`. You can also find request identifiers in the URLs of individual request logs in your [Dashboard](#). **If you need to contact us about a specific request, providing the request identifier will ensure the fastest possible resolution.**

Versioning

When **backwards-incompatible** changes are made to the API, a new, dated version is released. The current version is **2020-08-27**. Read our [API upgrades guide](#) to see our [API changelog](#) and to learn more about backwards compatibility.

All requests use your account API settings, unless you override the API version. The [changelog](#) lists every available version. *Note that by default webhook events are structured according to your account API version, unless you set an API version during endpoint creation.*

To set the API version on a specific request, send a `Stripe-Version` header.

You can visit [your Dashboard](#) to upgrade your API version. As a precaution, use API versioning to test a new API version before committing to an upgrade.

Related video: [Versioning](#).

Balance

This is an object representing your Stripe balance. You can retrieve it to see the balance currently on your Stripe account.

You can also retrieve the balance history, which contains a list of [transactions](#) that contributed to the balance (charges, payouts, and so forth).

The available and pending amounts for each currency are broken down further by payment source types.

The balance object

Attributes

available array of hashes

Funds that are available to be transferred or paid out, whether automatically by Stripe or explicitly via the [Transfers API](#) or [Payouts API](#). The available balance for each currency and payment type can be found in the `source_types` property.

pending array of hashes

Funds that are not yet available in the balance, due to the 7-day rolling pay cycle. The pending balance for each currency, and for each payment type, can be found in the `source_types` property.

More attributes

✓ **object** string, value is "balance"

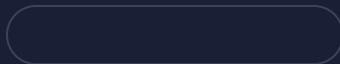
String representing the object's type. Objects of the same type share the same value.

✓ **connect_reserved** array of hashes CONNECT ONLY

Funds held due to negative balances on connected Custom accounts. The connect reserve balance for each currency and payment type can be found in the `source_types` property.

✓ **instant_available** array of hashes

Funds that can be paid out using Instant Payouts.



✓ **issuing** hash

Funds that can be spent on your [Issued Cards](#).



✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

Retrieve balance

Retrieves the current account balance, based on the authentication that was used to make the request. For a sample request, see [Accounting for negative balances](#).

Parameters

No parameters.

Returns

Returns a balance object for the account that was authenticated in the request.

Balance Transactions

Balance transactions represent funds moving through your Stripe account. They're created for every type of transaction that comes into or flows out of your Stripe account balance.

Related guide: [Balance Transaction Types](#).

The balance transaction object

Attributes

id string

Unique identifier for the object.

amount integer

Gross amount of the transaction, in cents.

currency currency

Three-letter ISO currency code, in lowercase. Must be a [supported currency](#).

description string

An arbitrary string attached to the object. Often useful for displaying to users.

fee integer

Fees (in cents) paid for this transaction.

fee_details array of hashes

Detailed breakdown of fees (in cents) paid for this transaction.

net integer

Net amount of the transaction, in cents.

source string [EXPANDABLE](#)

The Stripe object to which this transaction is related.

status string

If the transaction's net funds are available in the Stripe balance yet. Either [available](#) or [pending](#).

type string

Transaction type: [adjustment](#), [advance](#), [advance_funding](#), [anticipation_repayment](#), [application_fee](#), [application_fee_refund](#), [charge](#), [connect_collection_transfer](#), [contribution](#), [issuing_authorization_hold](#), [issuing_authorization_release](#), [issuing_dispute](#), [issuing_transaction](#), [payment](#), [payment_failure_refund](#), [payment_refund](#), [payout](#), [payout_cancel](#), [payout_failure](#), [refund](#), [refund_failure](#), [reserve_transaction](#), [reserved_funds](#), [stripe_fee](#), [stripe_fx_fee](#), [tax_fee](#), [topup](#), [topup_reversal](#), [transfer](#), [transfer_cancel](#), [transfer_failure](#), or [transfer_refund](#). [Learn more](#) about balance transaction types and what they represent. If you are looking to classify transactions for accounting purposes, you might want to consider [reporting_category](#) instead.

More attributes

✓ **object** string, value is "balance_transaction"

String representing the object's type. Objects of the same type share the same value.

✓ **available_on** timestamp

The date the transaction's net funds will become available in the Stripe balance.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **exchange_rate** decimal

The exchange rate used, if applicable, for this transaction. Specifically, if money was converted from currency A to currency B, then the `amount` in currency A, times `exchange_rate`, would be the `amount` in currency B. For example, suppose you charged a customer 10.00 EUR. Then the PaymentIntent's `amount` would be `1000` and `currency` would be `eur`. Suppose this was converted into 12.34 USD in your Stripe account. Then the BalanceTransaction's `amount` would be `1234`, `currency` would be `usd`, and `exchange_rate` would be `1.234`.

✓ **reporting_category** string

[Learn more](#) about how reporting categories can help you understand balance transactions from an accounting perspective.

Retrieve a balance transaction

Retrieves the balance transaction with the given ID.

Note that this endpoint previously used the path `/v1/balance/history/:id`.

Parameters

No parameters.

Returns

Returns a balance transaction if a valid balance transaction ID was provided. Returns [an error](#) otherwise.

List all balance transactions

Returns a list of transactions that have contributed to the Stripe account balance (e.g., charges, transfers, and so forth). The transactions are returned in sorted order, with the most recent transactions appearing first.

Note that this endpoint was previously called “Balance history” and used the path `/v1/balance/history`.

Parameters

payout optional

For automatic Stripe payouts only, only returns transactions that were paid out on the specified payout ID.

type optional

Only returns transactions of the given type. One of: `adjustment`, `advance`, `advance_funding`, `anticipation_repayment`, `application_fee`, `application_fee_refund`, `charge`, `connect_collection_transfer`, `contribution`, `issuing_authorization_hold`, `issuing_authorization_release`, `issuing_dispute`, `issuing_transaction`, `payment`, `payment_failure_refund`, `payment_refund`, `payout`, `payout_cancel`, `payout_failure`, `refund`, `refund_failure`, `reserve_transaction`, `reserved_funds`, `stripe_fee`, `stripe_fx_fee`, `tax_fee`, `topup`, `topup_reversal`, `transfer`, `transfer_cancel`, `transfer_failure`, or `transfer_refund`.

More parameters

created optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

currency optional

Only return transactions in a certain currency. Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **source** optional

Only returns the original transaction.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` transactions, starting after transaction `starting_after`. Each entry in the array is a separate transaction history object. If no more transactions are available, the resulting array will be empty.

Charges

To charge a credit or a debit card, you create a `Charge` object. You can retrieve and refund individual charges as well as list all charges. Charges are identified by a unique, random ID.

Related guide: [Accept a payment with the Charges API](#).

The charge object

Attributes

id string

Unique identifier for the object.

amount positive integer or zero

Amount intended to be collected by this payment. A positive integer representing how much to charge in the [smallest currency unit](#) (e.g., 100 cents to charge \$1.00 or 100 to charge ¥100, a zero-decimal currency). The minimum amount is \$0.50 US or [equivalent in charge currency](#). The amount value supports up to eight digits (e.g., a value of 99999999 for a USD charge of \$999,999.99).

balance_transaction string [EXPANDABLE](#)

ID of the balance transaction that describes the impact of this charge on your account balance (not including refunds or disputes).

billing_details hash

Billing information associated with the payment method at the time of the transaction.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

customer string [EXPANDABLE](#)

ID of the customer this charge is for if one exists.

description string

An arbitrary string attached to the object. Often useful for displaying to users.

disputed boolean

Whether the charge has been disputed.

invoice string [EXPANDABLE](#)

ID of the invoice this charge is for if one exists.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

payment_intent string [EXPANDABLE](#)

ID of the PaymentIntent associated with this charge, if one exists.

payment_method_details hash

Details about the payment method at the time of the transaction.



receipt_email string

This is the email address that the receipt for this charge was sent to.

refunded boolean

Whether the charge has been fully refunded. If the charge is only partially refunded, this attribute will still be false.

shipping hash

Shipping information for the charge.

statement_descriptor string

For card charges, use `statement_descriptor_suffix` instead. Otherwise, you can use this value as the complete description of a charge on your customers' statements. Must contain at least one letter, maximum 22 characters.

statement_descriptor_suffix string

Provides information about the charge that customers see on their statements.

Concatenated with the prefix (shortened descriptor) or statement descriptor that's set on the account to form the complete statement descriptor. Maximum 22 characters for the concatenated descriptor.

status enum

The status of the payment is either `succeeded`, `pending`, or `failed`.

Possible enum values

`succeeded``pending``failed`

More attributes

object string, value is "charge"

String representing the object's type. Objects of the same type share the same value.

amount_captured positive integer or zero

Amount in cents captured (can be less than the amount attribute on the charge if a partial capture was made).

amount_refunded positive integer or zero

Amount in cents refunded (can be less than the amount attribute on the charge if a partial refund was issued).

application string EXPANDABLE "APPLICATION" CONNECT ONLY

ID of the Connect application that created the charge.

✓ **application_fee** string EXPANDABLE CONNECT ONLY

The application fee (if any) for the charge. [See the Connect documentation](#) for details.

✓ **application_fee_amount** integer CONNECT ONLY

The amount of the application fee (if any) requested for the charge. [See the Connect documentation](#) for details.

✓ **calculated_statement_descriptor** string

The full statement descriptor that is passed to card networks, and that is displayed on your customers' credit card and bank statements. Allows you to see what the statement descriptor looks like after the static and dynamic portions are combined.

✓ **captured** boolean

If the charge was created without capturing, this Boolean represents whether it is still uncaptured or has since been captured.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **failure_balance_transaction** string EXPANDABLE

ID of the balance transaction that describes the reversal of the balance on your account due to payment failure.

✓ **failure_code** string

Error code explaining reason for charge failure if available (see [the errors section](#) for a list of codes).

✓ **failure_message** string

Message to user further explaining reason for charge failure if available.

✓ **fraud_details** hash

Information on fraud assessments for the charge.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **on_behalf_of** string EXPANDABLE CONNECT ONLY

The account (if any) the charge was made on behalf of without triggering an automatic transfer. See the [Connect documentation](#) for details.

✓ **order** string EXPANDABLE

ID of the order this charge is for if one exists.

✓ **outcome** hash

Details about whether the payment was accepted, and why. See [understanding declines](#) for details.

✓ **paid** boolean

`true` if the charge succeeded, or was successfully authorized for later capture.

✓ **payment_method** string

ID of the payment method used in this charge.

✓ **receipt_number** string

This is the transaction number that appears on email receipts sent for this charge. This attribute will be `null` until a receipt has been sent.

✓ **receipt_url** string

This is the URL to view the receipt for this charge. The receipt is kept up-to-date to the latest state of the charge, including any refunds. If the charge is for an Invoice, the receipt will be stylized as an Invoice receipt.

✓ **refunds** list

A list of refunds that have been applied to the charge.

✓ **review** string EXPANDABLE

ID of the review associated with this charge if one exists.

✓ **source_transfer** string EXPANDABLE CONNECT ONLY

The transfer ID which created this charge. Only present if the charge came from another Stripe account. See the [Connect documentation](#) for details.

✓ **transfer** string EXPANDABLE CONNECT ONLY

ID of the transfer to the `destination` account (only applicable if the charge was created using the `destination` parameter).

✓ **transfer_data** hash CONNECT ONLY

An optional dictionary including the account to automatically transfer to as part of a destination charge. See the [Connect documentation](#) for details.

✓ **transfer_group** string CONNECT ONLY

A string that identifies this transaction as part of a group. See the [Connect documentation](#) for details.

Create a charge

To charge a credit card or other payment source, you create a `Charge` object. If your API key is in test mode, the supplied payment source (e.g., card) won't actually be charged, although everything else will occur as if in live mode. (Stripe assumes that the charge would have completed successfully).

Parameters

amount REQUIRED

Amount intended to be collected by this payment. A positive integer representing how much to charge in the [smallest currency unit](#) (e.g., 100 cents to charge \$1.00 or 100 to charge ¥100, a zero-decimal currency). The minimum amount is \$0.50 US or [equivalent in charge currency](#). The amount value supports up to eight digits (e.g., a value of 99999999 for a USD charge of \$999,999.99).

currency REQUIRED

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

customer optional

The ID of an existing customer that will be charged in this request.

description optional

An arbitrary string which you can attach to a `Charge` object. It is displayed when in the web interface alongside the charge. Note that if you use Stripe to send automatic email receipts to your customers, your receipt emails will include the `description` of the charge(s) that they are describing.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

receipt_email optional

The email address to which this charge's **receipt** will be sent. The receipt will not be sent until the charge is paid, and no receipts will be sent for test mode charges. If this charge is for a **Customer**, the email address specified here will override the customer's email address. If `receipt_email` is specified for a charge in live mode, a receipt will be sent regardless of your **email settings**.

shipping optional dictionary

Shipping information for the charge. Helps prevent fraud on charges for physical goods.

source optional

A payment source to be charged. This can be the ID of a **card** (i.e., credit or debit card), a **bank account**, a **source**, a **token**, or a **connected account**. For certain sources—namely, **cards**, **bank accounts**, and attached **sources**—you must also pass the ID of the associated customer.

statement_descriptor optional

For card charges, use `statement_descriptor_suffix` instead. Otherwise, you can use this value as the complete description of a charge on your customers' statements. Must contain at least one letter, maximum 22 characters.

statement_descriptor_suffix optional

Provides information about the charge that customers see on their statements.

Concatenated with the prefix (shortened descriptor) or statement descriptor that's set on the account to form the complete statement descriptor. Maximum 22 characters for the concatenated descriptor.

More parameters

✓ **application_fee_amount** optional CONNECT ONLY

A fee in cents that will be applied to the charge and transferred to the application owner's Stripe account. The request must be made with an OAuth key or the `Stripe-Account` header in order to take an application fee. For more information, see the [application fees documentation](#).

✓ **capture** optional

Whether to immediately capture the charge. Defaults to `true`. When `false`, the charge issues an authorization (or pre-authorization), and will need to be [captured](#) later. Uncaptured charges expire after a set number of days (7 by default). For more information, see the [authorizing charges and settling later](#) documentation.

✓ **on_behalf_of** optional CONNECT ONLY

The Stripe account ID for which these funds are intended. Automatically set if you use the `destination` parameter. For details, see [Creating Separate Charges and Transfers](#).

✓ **transfer_data** optional dictionary CONNECT ONLY

An optional dictionary including the account to automatically transfer to as part of a destination charge. See the [Connect documentation](#) for details.



✓ **transfer_group** optional CONNECT ONLY

A string that identifies this transaction as part of a group. For details, see [Grouping transactions](#).

Returns

Returns the charge object if the charge succeeded. This call will return [an error](#) if something goes wrong. A common source of error is an invalid or expired card, or a valid card with insufficient available balance.

Verification responses

If the `cvc` parameter is provided, Stripe will attempt to check the correctness of the CVC, and will return this check's result. Similarly, if `address_line1` or `address_zip` are provided, Stripe will try to check the validity of those parameters.

Some card issuers do not support checking one or more of these parameters, in which case Stripe will return an `unavailable` result.

Also note that, depending on the card issuer, charges can succeed even when passed incorrect CVC and address information.

CVC <small><code>payment_method_details.card.checks.cvc_check</code></small>	
<code>pass</code>	The CVC provided is correct.
<code>fail</code>	The CVC provided is incorrect.
<code>unavailable</code>	The customer's card issuer did not check the CVC provided.
<code>unchecked</code>	The CVC was provided but hasn't been checked yet. Checks are typically performed when attaching a card to a <code>Customer</code> object, or when creating a charge. For more details, see Check if a card is valid without a charge .

ADDRESS LINE `payment_method_details.card.checks.address_line1_check`

pass	The first address line provided is correct.
fail	The first address line provided is incorrect.
unavailable	The customer's card issuer did not check the first address line provided.
unchecked	The first address line was provided but hasn't been checked yet. Checks are typically performed when attaching a card to a <code>Customer</code> object, or when creating a charge. For more details, see Check if a card is valid without a charge .

ADDRESS ZIP `payment_method_details.card.checks.address_zip_check`

pass	The ZIP code provided is correct.
fail	The ZIP code provided is incorrect.
unavailable	The customer's card issuer did not check the ZIP code.
unchecked	The ZIP code was provided but hasn't been checked yet. Checks are typically performed when attaching a card to a <code>Customer</code> object, or when creating a charge. For more details, see Check if a card is valid without a charge .

Retrieve a charge

Retrieves the details of a charge that has previously been created. Supply the unique charge ID that was returned from your previous request, and Stripe will return the corresponding charge information. The same information is returned when creating or refunding the charge.

Parameters

No parameters.

Returns

Returns a charge if a valid identifier was provided, and returns an error otherwise.

Update a charge

Updates the specified charge by setting the values of the parameters passed. Any parameters not provided will be left unchanged.

Parameters

customer optional

The ID of an existing customer that will be associated with this request. This field may only be updated if there is no existing associated customer with this charge.

description optional

An arbitrary string which you can attach to a charge object. It is displayed when in the web interface alongside the charge. Note that if you use Stripe to send automatic email receipts to your customers, your receipt emails will include the `description` of the charge(s) that they are describing.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

receipt_email optional

This is the email address that the receipt for this charge will be sent to. If this field is updated, then a new email receipt will be sent to the updated address.

shipping optional dictionary

Shipping information for the charge. Helps prevent fraud on charges for physical goods.

More parameters

[Collapse all](#)

✗ **fraud_details** optional dictionary

A set of key-value pairs you can attach to a charge giving information about its riskiness. If you believe a charge is fraudulent, include a `user_report` key with a value of `fraudulent`. If you believe a charge is safe, include a `user_report` key with a value of `safe`. Stripe will use the information you send to improve our fraud detection algorithms.

✓ **transfer_group** optional CONNECT ONLY

A string that identifies this transaction as part of a group. `transfer_group` may only be provided if it has not been set. See the [Connect documentation](#) for details.

Returns

Returns the charge object if the update succeeded. This call will return [an error](#) if update parameters are invalid.

Capture a charge

Capture the payment of an existing, uncaptured, charge. This is the second half of the two-step payment flow, where first you [created a charge](#) with the capture option set to false.

Uncaptured payments expire a set number of days after they are created ([7 by default](#)). If they are not captured by that point in time, they will be marked as refunded and will no longer be capturable.

Parameters

amount optional

The amount to capture, which must be less than or equal to the original amount. Any additional amount will be automatically refunded.

receipt_email optional

The email address to send this charge's receipt to. This will override the previously-specified email address for this charge, if one was set. Receipts will not be sent in test mode.

statement_descriptor optional

For card charges, use `statement_descriptor_suffix` instead. Otherwise, you can use this value as the complete description of a charge on your customers' statements. Must contain at least one letter, maximum 22 characters.

statement_descriptor_suffix optional

Provides information about the charge that customers see on their statements. Concatenated with the prefix (shortened descriptor) or statement descriptor that's set on the account to form the complete statement descriptor. Maximum 22 characters for the concatenated descriptor.

More parameters

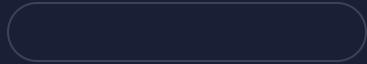
[Collapse all](#)

✓ **application_fee_amount** optional CONNECT ONLY

An application fee amount to add on to this charge, which must be less than or equal to the original amount.

✓ **transfer_data** optional dictionary CONNECT ONLY

An optional dictionary including the account to automatically transfer to as part of a destination charge. See the [Connect documentation](#) for details.



✓ **transfer_group** optional CONNECT ONLY

A string that identifies this transaction as part of a group. `transfer_group` may only be provided if it has not been set. See the [Connect documentation](#) for details.

Returns

Returns the charge object, with an updated captured property (set to true). Capturing a charge will always succeed, unless the charge is already refunded, expired, captured, or an invalid capture amount is specified, in which case this method will return an error.

List all charges

Returns a list of charges you've previously created. The charges are returned in sorted order, with the most recent charges appearing first.

Parameters

customer optional

Only return charges for the customer specified by this customer ID.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **payment_intent** optional

Only return charges that were created by the PaymentIntent specified by this PaymentIntent ID.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

✓ **transfer_group** optional CONNECT ONLY

Only return charges for this transfer group.

Returns

A dictionary with a `data` property that contains an array of up to `limit` charges, starting after charge `starting_after`. Each entry in the array is a separate charge object. If no more charges are available, the resulting array will be empty. If you provide a non-existent customer ID, this call returns an error.

Search charges

Search for charges you've previously created using Stripe's [Search Query Language](#). Don't use search in read-after-write flows where strict consistency is necessary. Under normal operating conditions, data is searchable in less than a minute. Occasionally, propagation of new or updated data can be up to an hour behind during outages. Search functionality is not available to merchants in India.

Parameters

query REQUIRED

The search query string. See [search query language](#) and the list of supported [query fields for charges](#).

limit optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

page optional

A cursor for pagination across multiple pages of results. Don't include this parameter on the first call. Use the next_page value returned in a previous response to request subsequent results.

Returns

A dictionary with a `data` property that contains an array of up to `limit` charges. If no objects match the query, the resulting array will be empty. See the related guide on [expanding properties in lists](#).

Customers

This object represents a customer of your business. It lets you create recurring charges and track payments that belong to the same customer.

Related guide: [Save a card during payment](#).

The customer object

Attributes

id string

Unique identifier for the object.

address hash

The customer's address.

description string

An arbitrary string attached to the object. Often useful for displaying to users.

email string

The customer's email address.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

name string

The customer's full name or business name.

phone string

The customer's phone number.

shipping hash

Mailing and shipping address for the customer. Appears on invoices emailed to this customer.

More attributes[Collapse all](#)▼ **object** string, value is "customer"

String representing the object's type. Objects of the same type share the same value.

▼ **balance** integer

Current balance, if any, being stored on the customer. If negative, the customer has credit to apply to their next invoice. If positive, the customer has an amount owed that will be added to their next invoice. The balance does not refer to any unpaid invoices; it solely takes into account amounts that have yet to be successfully applied to any invoice. This balance is only taken into account as invoices are finalized.

▼ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

▼ **currency** string

Three-letter [ISO code for the currency](#) the customer can be charged in for recurring billing purposes.

✓ **default_source** string EXPANDABLE

ID of the default payment source for the customer.

If you are using payment methods created via the PaymentMethods API, see the [invoice_settings.default_payment_method](#) field instead.

✓ **delinquent** boolean

When the customer's latest invoice is billed by charging automatically, `delinquent` is `true` if the invoice's latest charge failed. When the customer's latest invoice is billed by sending an invoice, `delinquent` is `true` if the invoice isn't paid by its due date.

If an invoice is marked uncollectible by `dunning`, `delinquent` doesn't get reset to `false`.

✓ **discount** hash, [discount object](#)

Describes the current discount active on the customer, if there is one.

✓ **invoice_prefix** string

The prefix for the customer used to generate unique invoice numbers.

✓ **invoice_settings** hash

The customer's default invoice settings.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **next_invoice_sequence** integer

The suffix of the customer's next invoice number, e.g., 0001.

✓ **preferred_locales** array containing strings

The customer's preferred locales (languages), ordered by preference.

✓ **sources** list EXPANDABLE

The customer's payment sources, if any. This field is not included by default. To include it in the response, [expand](#) the `sources` field.

✓ **subscriptions** list EXPANDABLE

The customer's current subscriptions, if any. This field is not included by default. To include it in the response, [expand](#) the `subscriptions` field.

✓ **tax** hash EXPANDABLE

Tax details for the customer. This field is not included by default. To include it in the response, [expand](#) the `tax` field.

✓ **tax_exempt** string

Describes the customer's tax exemption status. One of `none`, `exempt`, or `reverse`. When set to `reverse`, invoice and receipt PDFs include the text "**Reverse charge**".

✓ **tax_ids** list EXPANDABLE

The customer's tax IDs. This field is not included by default. To include it in the response, [expand](#) the `tax_ids` field.

✓ **test_clock** string EXPANDABLE

ID of the test clock this customer belongs to.

Create a customer

Parameters

address optional dictionary

The customer's address.

description optional

An arbitrary string that you can attach to a customer object. It is displayed alongside the customer in the dashboard.

email optional

Customer's email address. It's displayed alongside the customer in your dashboard and can be useful for searching and tracking. This may be up to *512 characters*.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

name optional

The customer's full name or business name.

payment_method optional

The ID of the PaymentMethod to attach to the customer.

phone optional

The customer's phone number.

shipping optional dictionary

The customer's shipping information. Appears on invoices emailed to this customer.

More parameters[Collapse all](#)` **balance** optional

An integer amount in cents that represents the customer's current balance, which affect the customer's future invoices. A negative amount represents a credit that decreases the amount due on an invoice; a positive amount increases the amount due on an invoice.

` **cash_balance** optional dictionary PREVIEW FEATURE

Balance information and default balance settings for this customer.

` **coupon** optional

If you provide a coupon code, the customer will have a discount applied on all recurring charges. Charges you create through the API will not have the discount.

` **invoice_prefix** optional

The prefix for the customer used to generate unique invoice numbers. Must be 3–12 uppercase letters or numbers.

` **invoice_settings** optional dictionary

Default invoice settings for this customer.

` **next_invoice_sequence** optional

The sequence to be used on the customer's next invoice. Defaults to 1.

` **preferred_locales** optional

Customer's preferred languages, ordered by preference.

✓ **promotion_code** optional

The API ID of a promotion code to apply to the customer. The customer will have a discount applied on all recurring payments. Charges you create through the API will not have the discount.

✓ **source** optional

When using payment sources created via the Token or Sources APIs, passing `source` will create a new source object, make it the new customer default source, and delete the old customer default if one exists. If you want to add additional sources instead of replacing the existing default, use the [card creation API](#). Whenever you attach a card to a customer, Stripe will automatically validate the card.

✓ **tax** optional dictionary

Tax details about the customer.

✓ **tax_exempt** optional

The customer's tax exemption. One of `none`, `exempt`, or `reverse`.

✓ **tax_id_data** optional array of hashes

The customer's tax IDs.

✓ **test_clock** optional

ID of the test clock to attach to the customer.

Returns

Returns the customer object if the update succeeded. Returns [an error](#) if create parameters are invalid (e.g. specifying an invalid coupon or an invalid source).

Retrieve a customer

Retrieves a Customer object.

Parameters

Returns

Returns the Customer object for a valid identifier. If it's for a deleted Customer, a subset of the customer's information is returned, including a `deleted` property that's set to true.

Update a customer

Updates the specified customer by setting the values of the parameters passed. Any parameters not provided will be left unchanged. For example, if you pass the **source** parameter, that becomes the customer's active source (e.g., a card) to be used for all charges in the future. When you update a customer to a new valid card source by passing the **source** parameter: for each of the customer's current subscriptions, if the subscription bills automatically and is in the `past_due` state, then the latest open invoice for the subscription with automatic collection enabled will be retried. This retry will not count as an automatic retry, and will not affect the next regularly scheduled payment for the invoice. Changing the **default_source** for a customer will not trigger this behavior.

This request accepts mostly the same arguments as the customer creation call.

Parameters

address optional dictionary

The customer's address.



description optional

An arbitrary string that you can attach to a customer object. It is displayed alongside the customer in the dashboard.

email optional

Customer's email address. It's displayed alongside the customer in your dashboard and can be useful for searching and tracking. This may be up to *512 characters*.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

name optional

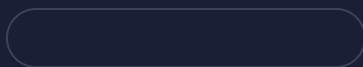
The customer's full name or business name.

phone optional

The customer's phone number.

shipping optional dictionary

The customer's shipping information. Appears on invoices emailed to this customer.



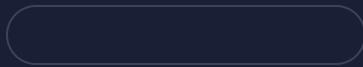
More parameters

balance optional

An integer amount in cents that represents the customer's current balance, which affect the customer's future invoices. A negative amount represents a credit that decreases the amount due on an invoice; a positive amount increases the amount due on an invoice.

cash_balance optional dictionary PREVIEW FEATURE

Balance information and default balance settings for this customer.

**coupon** optional

If you provide a coupon code, the customer will have a discount applied on all recurring charges. Charges you create through the API will not have the discount.

default_source optional

If you are using payment methods created via the PaymentMethods API, see the [invoice_settings.default_payment_method](#) parameter.

Provide the ID of a payment source already attached to this customer to make it this customer's default payment source.

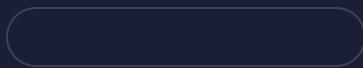
If you want to add a new payment source and make it the default, see the [source](#) property.

invoice_prefix optional

The prefix for the customer used to generate unique invoice numbers. Must be 3–12 uppercase letters or numbers.

invoice_settings optional dictionary

Default invoice settings for this customer.



next_invoice_sequence optional

The sequence to be used on the customer's next invoice. Defaults to 1.

✓ **preferred_locales** optional

Customer's preferred languages, ordered by preference.

✓ **promotion_code** optional

The API ID of a promotion code to apply to the customer. The customer will have a discount applied on all recurring payments. Charges you create through the API will not have the discount.

✓ **source** optional

When using payment sources created via the Token or Sources APIs, passing `source` will create a new source object, make it the new customer default source, and delete the old customer default if one exists. If you want to add additional sources instead of replacing the existing default, use the [card creation API](#). Whenever you attach a card to a customer, Stripe will automatically validate the card.

✓ **tax** optional dictionary

Tax details about the customer.

✓ **tax_exempt** optional

The customer's tax exemption. One of `none`, `exempt`, or `reverse`.

Returns

Returns the customer object if the update succeeded. Returns [an error](#) if update parameters are invalid (e.g. specifying an invalid coupon or an invalid source).

Delete a customer

Permanently deletes a customer. It cannot be undone. Also immediately cancels any active subscriptions on the customer.

Parameters

No parameters.

Returns

Returns an object with a deleted parameter on success. If the customer ID does not exist, this call returns [an error](#).

Unlike other objects, deleted customers can still be retrieved through the API, in order to be able to track the history of customers while still removing their credit card details and preventing any further operations to be performed (such as adding a new subscription).

List all customers

Returns a list of your customers. The customers are returned sorted by creation date, with the most recent customers appearing first.

Parameters

email optional

A case-sensitive filter on the list based on the customer's `email` field. The value must be a string.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

✓ `test_clock` optional

Provides a list of customers that are associated with the specified test clock. The response will not include customers with test clocks if this parameter is not set.

Returns

A dictionary with a `data` property that contains an array of up to `limit` customers, starting after customer `starting_after`. Passing an optional `email` will result in filtering to customers with only that exact email address. Each entry in the array is a separate customer object. If no more customers are available, the resulting array will be empty. This request should never return an error.

Search customers

Search for customers you've previously created using Stripe's [Search Query Language](#). Don't use search in read-after-write flows where strict consistency is necessary. Under normal operating conditions, data is searchable in less than a minute. Occasionally, propagation of new or updated data can be up to an hour behind during outages. Search functionality is not available to merchants in India.

Parameters

`query` REQUIRED

The search query string. See [search query language](#) and the list of supported [query fields for customers](#).

`limit` optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

`page` optional

A cursor for pagination across multiple pages of results. Don't include this parameter on the first call. Use the `next_page` value returned in a previous response to request subsequent results.

Returns

A dictionary with a `data` property that contains an array of up to `limit` customers. If no objects match the query, the resulting array will be empty. See the related guide on [expanding properties in lists](#).

Disputes

A dispute occurs when a customer questions your charge with their card issuer. When this happens, you're given the opportunity to respond to the dispute with evidence that shows that the charge is legitimate. You can find more information about the dispute process in our [Disputes and Fraud](#) documentation.

Related guide: [Disputes and Fraud](#).

The dispute object

Attributes

id string

Unique identifier for the object.

amount integer

Disputed amount. Usually the amount of the charge, but can differ (usually because of currency fluctuation or because only part of the order is disputed).

charge string EXPANDABLE

ID of the charge that was disputed.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

evidence hash

Evidence provided to respond to a dispute. Updating any field in the hash will submit all fields in the hash for review.

metadata hash

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

payment_intent string EXPANDABLE

ID of the PaymentIntent that was disputed.

reason string

Reason given by cardholder for dispute. Possible values are `bank_cannot_process`, `check_returned`, `credit_not_processed`, `customer_initiated`, `debit_not_authorized`, `duplicate`, `fraudulent`, `general`, `incorrect_account_details`, `insufficient_funds`, `product_not_received`, `product_unacceptable`, `subscription_canceled`, or `unrecognized`. Read more about [dispute reasons](#).

status string

Current status of dispute. Possible values are `warning_needs_response`, `warning_under_review`, `warning_closed`, `needs_response`, `under_review`, `charge_refunded`, `won`, or `lost`.

More attributes

[Collapse all](#)

✓ **object** string, value is "dispute"

String representing the object's type. Objects of the same type share the same value.

✓ **balance_transactions** array, contains: `balance_transaction` object

List of zero, one, or two balance transactions that show funds withdrawn and reinstated to your Stripe account as a result of this dispute.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **evidence_details** hash

Information about the evidence submission.

✓ **is_charge_refundable** boolean

If true, it is still possible to refund the disputed payment. Once the payment has been fully refunded, no further funds will be withdrawn from your Stripe account as a result of this dispute.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

The dispute evidence object

Attributes

access_activity_log string

Any server or activity logs showing proof that the customer accessed or downloaded the purchased digital product. This information should include IP addresses, corresponding timestamps, and any detailed recorded activity.

billing_address string

The billing address provided by the customer.

cancellation_policy string EXPANDABLE

(ID of a [file upload](#)) Your subscription cancellation policy, as shown to the customer.

cancellation_policy_disclosure string

An explanation of how and when the customer was shown your refund policy prior to purchase.

cancellation_rebuttal string

A justification for why the customer's subscription was not canceled.

customer_communication string EXPANDABLE

(ID of a [file upload](#)) Any communication with the customer that you feel is relevant to your case. Examples include emails proving that the customer received the product or service, or demonstrating their use of or satisfaction with the product or service.

customer_email_address string

The email address of the customer.

customer_name string

The name of the customer.

customer_purchase_ip string

The IP address that the customer used when making the purchase.

customer_signature string EXPANDABLE

(ID of a [file upload](#)) A relevant document or contract showing the customer's signature.

duplicate_charge_documentation string EXPANDABLE

(ID of a [file upload](#)) Documentation for the prior charge that can uniquely identify the charge, such as a receipt, shipping label, work order, etc. This document should be paired with a similar document from the disputed payment that proves the two payments are separate.

duplicate_charge_explanation string

An explanation of the difference between the disputed charge versus the prior charge that appears to be a duplicate.

duplicate_charge_id string

The Stripe ID for the prior charge which appears to be a duplicate of the disputed charge.

product_description string

A description of the product or service that was sold.

receipt string EXPANDABLE

(ID of a [file upload](#)) Any receipt or message sent to the customer notifying them of the charge.

refund_policy string EXPANDABLE

(ID of a [file upload](#)) Your refund policy, as shown to the customer.

refund_policy_disclosure string

Documentation demonstrating that the customer was shown your refund policy prior to purchase.

refund_refusal_explanation string

A justification for why the customer is not entitled to a refund.

service_date string

The date on which the customer received or began receiving the purchased service, in a clear human-readable format.

service_documentation string EXPANDABLE

(ID of a [file upload](#)) Documentation showing proof that a service was provided to the customer. This could include a copy of a signed contract, work order, or other form of written agreement.

shipping_address string

The address to which a physical product was shipped. You should try to include as complete address information as possible.

shipping_carrier string

The delivery service that shipped a physical product, such as Fedex, UPS, USPS, etc. If multiple carriers were used for this purchase, please separate them with commas.

shipping_date string

The date on which a physical product began its route to the shipping address, in a clear human-readable format.

shipping_documentation string EXPANDABLE

(ID of a [file upload](#)) Documentation showing proof that a product was shipped to the customer at the same address the customer provided to you. This could include a copy of the shipment receipt, shipping label, etc. It should show the customer's full shipping address, if possible.

shipping_tracking_number string

The tracking number for a physical product, obtained from the delivery service. If multiple tracking numbers were generated for this purchase, please separate them with commas.

uncategorized_file string EXPANDABLE

(ID of a [file upload](#)) Any additional evidence or statements.

uncategorized_text string

Any additional evidence or statements.

Retrieve a dispute

Retrieves the dispute with the given ID.

Parameters

No parameters.

Returns

Returns a dispute if a valid dispute ID was provided. Returns [an error](#) otherwise.

Update a dispute

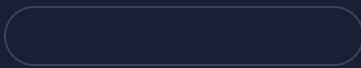
When you get a dispute, contacting your customer is always the best first step. If that doesn't work, you can submit evidence to help us resolve the dispute in your favor. You can do this in your [dashboard](#), but if you prefer, you can use the API to submit evidence programmatically.

Depending on your dispute type, different evidence fields will give you a better chance of winning your dispute. To figure out which evidence fields to provide, see our [guide to dispute types](#).

Parameters

evidence optional dictionary

Evidence to upload, to respond to a dispute. Updating any field in the hash will submit all fields in the hash for review. The combined character count of all fields is limited to 150,000.



metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

submit optional

Whether to immediately submit evidence to the bank. If `false`, evidence is staged on the dispute. Staged evidence is visible in the API and Dashboard, and can be submitted to the bank by making another request with this attribute set to `true` (the default).

Returns

Returns the dispute object.

Close a dispute

Closing the dispute for a charge indicates that you do not have any evidence to submit and are essentially dismissing the dispute, acknowledging it as lost.

The status of the dispute will change from `needs_response` to `lost`. *Closing a dispute is irreversible.*

Parameters

No parameters.

Returns

Returns the dispute object.

List all disputes

Returns a list of your disputes.

Parameters

charge optional

Only return disputes associated to the charge specified by this charge ID.

payment_intent optional

Only return disputes associated to the PaymentIntent specified by this PaymentIntent ID.

More parameters

[Collapse all](#)

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

starting_after optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` disputes, starting after dispute `starting_after`. Each entry in the array is a separate dispute object. If no more disputes are available, the resulting array will be empty. This request should never return an error.

Events

Events are our way of letting you know when something interesting happens in your account. When an interesting event occurs, we create a new `Event` object. For example, when a charge succeeds, we create a `charge.succeeded` event; and when an invoice payment attempt fails, we create an

`invoice.payment_failed` event. Note that many API requests may cause multiple events to be created. For example, if you create a new subscription for a customer, you will receive both a `customer.subscription.created` event and a `charge.succeeded` event.

Events occur when the state of another API resource changes. The state of that resource at the time of the change is embedded in the event's data field. For example, a `charge.succeeded` event will contain a charge, and an `invoice.payment_failed` event will contain an invoice.

As with other API resources, you can use endpoints to retrieve an [individual event](#) or a [list of events](#) from the API. We also have a separate [webhooks](#) system for sending the `Event` objects directly to an endpoint on your server. Webhooks are managed in your [account settings](#), and our [Using Webhooks](#) guide will help you get set up.

When using [Connect](#), you can also receive notifications of events that occur in connected accounts. For these events, there will be an additional `account`

attribute in the received `Event` object.

NOTE: Right now, access to events through the [Retrieve Event API](#) is guaranteed only for 30 days.

The event object

Attributes

id string

Unique identifier for the object.

api_version string

The Stripe API version used to render `data`. *Note: This property is populated only for events on or after October 31, 2014.*

data hash

Object containing data associated with the event.

request hash

Information on the API request that instigated the event.

type string

Description of the event (e.g., `invoice.created` or `charge.refunded`).

More attributes

[Collapse all](#)

✓ **object** string, value is "event"

String representing the object's type. Objects of the same type share the same value.

✓ **account** string CONNECT ONLY

The connected account that originated the event.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **pending_webhooks** positive integer or zero

Number of webhooks that have yet to be successfully delivered (i.e., to return a 20x response) to the URLs you've specified.

Retrieve an event

Retrieves the details of an event. Supply the unique identifier of the event, which you might have received in a webhook.

Parameters

No parameters.

Returns

Returns an event object if a valid identifier was provided. All events share a common structure, detailed to the right. The only property that will differ is the `data` property.

In each case, the `data` dictionary will have an attribute called `object` and its value will be the same as retrieving the same object directly from the API. For example, a `customer.created` event will have the same information as retrieving the relevant customer would.

In cases where the attributes of an object have changed, `data` will also contain a dictionary containing the changes.

List all events

List events, going back up to 30 days. Each event data is rendered according to Stripe API version at its creation time, specified in `event object api_version` attribute (not according to your current Stripe API version or `Stripe-Version` header).

Parameters

types optional

An array of up to 20 strings containing specific event names. The list will be filtered to include only events with a matching event property. You may pass either `type` or `types`, but not both.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **delivery_success** optional

Filter events by whether all webhooks were successfully delivered. If false, events which are still pending or have failed all delivery attempts to a webhook endpoint will be returned.

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

✓ **type** optional

A string containing a specific event name, or group of events using * as a wildcard. The list will be filtered to include only events with a matching event property.

Returns

A dictionary with a `data` property that contains an array of up to `limit` events, starting after event `starting_after`. Each entry in the array is a separate event object. If no more events are available, the resulting array will be empty. This request should never return an error.

Types of events

This is a list of all the types of events we currently send. We may add more at any time, so in developing and maintaining your code, you should not assume that only these types exist.

You'll notice that these events follow a pattern: `resource.event`. Our goal is to design a consistent system that makes things easier to anticipate and code against. **NOTE:** Events that occur on subresources like `customer.subscription` do not trigger the parent's `update` event.

Events marked as `SELECTION REQUIRED` are only created when a `webhook` has been configured to listen for that type of event specifically. A webhook set to listen to all events will **not** receive an event requiring explicit selection.

Event

`account.updated` `data.object` is an `account`

Occurs whenever an account status or property has changed.

`account.application.authorized` `data.object` is an "application"

Occurs whenever a user authorizes an application. Sent to the related application only.



`account.application.deauthorized` `data.object` is an "application"

Occurs whenever a user deauthorizes an application. Sent to the related application only.



`account.external_account.created`

`data.object` is an external account (e.g., `card` or `bank account`)

Occurs whenever an external account is created.

`account.external_account.deleted`

`data.object` is an external account (e.g., `card` or `bank account`)

Occurs whenever an external account is deleted.

`account.external_account.updated`

`data.object` is an external account (e.g., `card` or `bank account`)

Occurs whenever an external account is updated.

application_fee.created `[data.object]` is an application fee

Occurs whenever an application fee is created on a charge.

application_fee.refunded `[data.object]` is an application fee

Occurs whenever an application fee is refunded, whether from refunding a charge or from [refunding the application fee directly](#). This includes partial refunds.

application_fee.refund.updated `[data.object]` is a [fee refund](#)

Occurs whenever an application fee refund is updated.

balance.available `[data.object]` is a [balance](#)

Occurs whenever your Stripe balance has been updated (e.g., when a charge is available to be paid out). By default, Stripe automatically transfers funds in your balance to your bank account on a daily basis.

billing_portal.configuration.created `[data.object]` is a [portal configuration](#)

Occurs whenever a portal configuration is created.

billing_portal.configuration.updated `[data.object]` is a [portal configuration](#)

Occurs whenever a portal configuration is updated.

capability.updated `[data.object]` is a [capability](#)

Occurs whenever a capability has new requirements or a new status.

cash_balance.funds_available `[data.object]` is a [cash balance](#)

Occurs whenever there is a positive remaining cash balance after Stripe automatically reconciles new funds into the cash balance. If you enabled manual reconciliation, this webhook will fire whenever there are new funds into the cash balance.

charge.captured `[data.object]` is a [charge](#)

Occurs whenever a previously uncaptured charge is captured.

charge.expired `[data.object]` is a [charge](#)

Occurs whenever an uncaptured charge expires.

charge.failed `[data.object]` is a [charge](#)

Occurs whenever a failed charge attempt occurs.

charge.pending `[data.object]` is a [charge](#)

Occurs whenever a pending charge is created.

charge.refunded `[data.object]` is a [charge](#)

Occurs whenever a charge is refunded, including partial refunds.

charge.succeeded `data.object` is a [charge](#)

Occurs whenever a charge is successful.

charge.updated `data.object` is a [charge](#)

Occurs whenever a charge description or metadata is updated.

charge.dispute.closed `data.object` is a [dispute](#)

Occurs when a dispute is closed and the dispute status changes to [lost](#), [warning_closed](#), or [won](#).

charge.dispute.created `data.object` is a [dispute](#)

Occurs whenever a customer disputes a charge with their bank.

charge.dispute.funds_reinstated `data.object` is a [dispute](#)

Occurs when funds are reinstated to your account after a dispute is closed. This includes [partially refunded payments](#).

charge.dispute.funds_withdrawn `data.object` is a [dispute](#)

Occurs when funds are removed from your account due to a dispute.

charge.dispute.updated `data.object` is a [dispute](#)

Occurs when the dispute is updated (usually with evidence).

charge.refund.updated `data.object` is a [refund](#)

Occurs whenever a refund is updated, on selected payment methods.

checkout.session.async_payment_failed `data.object` is a [checkout session](#)

Occurs when a payment intent using a delayed payment method fails.

checkout.session.async_payment_succeeded `data.object` is a [checkout session](#)

Occurs when a payment intent using a delayed payment method finally succeeds.

checkout.session.completed `data.object` is a [checkout session](#)

Occurs when a Checkout Session has been successfully completed.

checkout.session.expired `data.object` is a [checkout session](#)

Occurs when a Checkout Session is expired.

coupon.created `data.object` is a [coupon](#)

Occurs whenever a coupon is created.

coupon.deleted `data.object` is a [coupon](#)

Occurs whenever a coupon is deleted.

coupon.updated `data.object` is a [coupon](#)

Occurs whenever a coupon is updated.

credit_note.created `data.object` is a [credit note](#)

Occurs whenever a credit note is created.

credit_note.updated `data.object` is a [credit note](#)

Occurs whenever a credit note is updated.

credit_note.voided `data.object` is a [credit note](#)

Occurs whenever a credit note is voided.

customer.created `data.object` is a [customer](#)

Occurs whenever a new customer is created.

customer.deleted `data.object` is a [customer](#)

Occurs whenever a customer is deleted.

customer.updated `data.object` is a [customer](#)

Occurs whenever any property of a customer changes.

customer.discount.created `data.object` is a [discount](#)

Occurs whenever a coupon is attached to a customer.

customer.discount.deleted `data.object` is a [discount](#)

Occurs whenever a coupon is removed from a customer.

customer.discount.updated `data.object` is a [discount](#)

Occurs whenever a customer is switched from one coupon to another.

customer.source.created `data.object` is a source (e.g., [card](#))

Occurs whenever a new source is created for a customer.

customer.source.deleted `data.object` is a source (e.g., [card](#))

Occurs whenever a source is removed from a customer.

customer.source.expiring `data.object` is a source (e.g., [card](#))

Occurs whenever a card or source will expire at the end of the month.

customer.source.updated `data.object` is a source (e.g., [card](#))

Occurs whenever a source's details are changed.

customer.subscription.created `data.object` is a [subscription](#)

Occurs whenever a customer is signed up for a new plan.

customer.subscription.deleted `data.object` is a [subscription](#)

Occurs whenever a customer's subscription ends.

customer.subscription.pending_update_applied `data.object` is a [subscription](#)

Occurs whenever a customer's subscription's pending update is applied, and the subscription is updated.

customer.subscription.pending_update_expired `data.object` is a [subscription](#)

Occurs whenever a customer's subscription's pending update expires before the related invoice is paid.

customer.subscription.trial_will_end `data.object` is a [subscription](#)

Occurs three days before a subscription's trial period is scheduled to end, or when a trial is ended immediately (using `trial_end=now`).

customer.subscription.updated `data.object` is a [subscription](#)

Occurs whenever a subscription changes (e.g., switching from one plan to another, or changing the status from trial to active).

customer.tax_id.created `data.object` is a [tax id](#)

Occurs whenever a tax ID is created for a customer.

customer.tax_id.deleted `data.object` is a [tax id](#)

Occurs whenever a tax ID is deleted from a customer.

customer.tax_id.updated `data.object` is a [tax id](#)

Occurs whenever a customer's tax ID is updated.

file.created `data.object` is a [file](#)

Occurs whenever a new Stripe-generated file is available for your account.

identity.verification_session.canceled `data.object` is a [verification session](#)

Occurs whenever a VerificationSession is canceled

identity.verification_session.created `data.object` is a [verification session](#)

Occurs whenever a VerificationSession is created

identity.verification_session.processing `data.object` is a [verification session](#)

Occurs whenever a VerificationSession transitions to processing

identity.verification_session.redacted SELECTION REQUIRED

`data.object` is a [verification session](#)

Occurs whenever a VerificationSession is redacted. You must create a webhook endpoint which explicitly subscribes to this event type to access it. Webhook endpoints which subscribe to all events will not include this event type.

identity.verification_session.requires_input `data.object` is a [verification session](#)

Occurs whenever a VerificationSession transitions to require user input

identity.verification_session.verified `data.object` is a [verification session](#)

Occurs whenever a VerificationSession transitions to verified

invoice.created `data.object` is an [invoice](#)

Occurs whenever a new invoice is created. To learn how webhooks can be used with this event, and how they can affect it, see [Using Webhooks with Subscriptions](#).

invoice.deleted `data.object` is an [invoice](#)

Occurs whenever a draft invoice is deleted.

invoice.finalization_failed `data.object` is an [invoice](#)

Occurs whenever a draft invoice cannot be finalized. See the invoice's [last finalization error](#) for details.

invoice.finalized `data.object` is an [invoice](#)

Occurs whenever a draft invoice is finalized and updated to be an open invoice.

invoice.marked_uncollectible `data.object` is an [invoice](#)

Occurs whenever an invoice is marked uncollectible.

invoice.paid `data.object` is an [invoice](#)

Occurs whenever an invoice payment attempt succeeds or an invoice is marked as paid out-of-band.

invoice.payment_action_required `data.object` is an [invoice](#)

Occurs whenever an invoice payment attempt requires further user action to complete.

invoice.payment_failed `data.object` is an [invoice](#)

Occurs whenever an invoice payment attempt fails, due either to a declined payment or to the lack of a stored payment method.

invoice.payment_succeeded `data.object` is an [invoice](#)

Occurs whenever an invoice payment attempt succeeds.

invoice.sent `data.object` is an [invoice](#)

Occurs whenever an invoice email is sent out.

invoice.upcoming `[data.object]` is an [invoice](#)

Occurs X number of days before a subscription is scheduled to create an invoice that is automatically charged—where X is determined by your [subscriptions settings](#). Note: The received [Invoice](#) object will not have an invoice ID.

invoice.updated `[data.object]` is an [invoice](#)

Occurs whenever an invoice changes (e.g., the invoice amount).

invoice.voided `[data.object]` is an [invoice](#)

Occurs whenever an invoice is voided.

invoiceitem.created `[data.object]` is an [invoiceitem](#)

Occurs whenever an invoice item is created.

invoiceitem.deleted `[data.object]` is an [invoiceitem](#)

Occurs whenever an invoice item is deleted.

invoiceitem.updated `[data.object]` is an [invoiceitem](#)

Occurs whenever an invoice item is updated.

issuing_authorization.created `[data.object]` is an [issuing authorization](#)

Occurs whenever an authorization is created.

issuing_authorization.request SELECTION REQUIRED `[data.object]` is an [issuing authorization](#)

Represents a synchronous request for authorization, see [Using your integration to handle authorization requests](#). You must create a webhook endpoint which explicitly subscribes to this event type to access it. Webhook endpoints which subscribe to all events will not include this event type.

issuing_authorization.updated `[data.object]` is an [issuing authorization](#)

Occurs whenever an authorization is updated.

issuing_card.created `[data.object]` is an [issuing card](#)

Occurs whenever a card is created.

issuing_card.updated `[data.object]` is an [issuing card](#)

Occurs whenever a card is updated.

issuing_cardholder.created `[data.object]` is an [issuing cardholder](#)

Occurs whenever a cardholder is created.

issuing_cardholder.updated `[data.object]` is an [issuing cardholder](#)

Occurs whenever a cardholder is updated.

issuing_dispute.closed `data.object` is an issuing dispute

Occurs whenever a dispute is won, lost or expired.

issuing_dispute.created `data.object` is an issuing dispute

Occurs whenever a dispute is created.

issuing_dispute.funds_reinstated `data.object` is an issuing dispute

Occurs whenever funds are reinstated to your account for an Issuing dispute.

issuing_dispute.submitted `data.object` is an issuing dispute

Occurs whenever a dispute is submitted.

issuing_dispute.updated `data.object` is an issuing dispute

Occurs whenever a dispute is updated.

issuing_transaction.created `data.object` is an issuing transaction

Occurs whenever an issuing transaction is created.

issuing_transaction.updated `data.object` is an issuing transaction

Occurs whenever an issuing transaction is updated.

mandate.updated `data.object` is a mandate

Occurs whenever a Mandate is updated.

order.created `data.object` is an order

Occurs whenever an order is created.

order.payment_failed `data.object` is an order

Occurs whenever an order payment attempt fails.

order.payment_succeeded `data.object` is an order

Occurs whenever an order payment attempt succeeds.

order.updated `data.object` is an order

Occurs whenever an order is updated.

order_return.created `data.object` is an order return

Occurs whenever an order return is created.

payment_intent.amount_capturable_updated `data.object` is a payment intent

Occurs when a PaymentIntent has funds to be captured. Check the `amount_capturable` property on the PaymentIntent to determine the amount that can be captured. You may capture the PaymentIntent with an `amount_to_capture` value up to the specified amount.

[Learn more about capturing PaymentIntents.](#)

payment_intent.canceled `data.object` is a payment intent

Occurs when a PaymentIntent is canceled.

payment_intent.created `data.object` is a payment intent

Occurs when a new PaymentIntent is created.

payment_intent.partially_funded `data.object` is a payment intent

Occurs when funds are applied to a customer_balance PaymentIntent and the 'amount_remaining' changes.

payment_intent.payment_failed `data.object` is a payment intent

Occurs when a PaymentIntent has failed the attempt to create a payment method or a payment.

payment_intent.processing `data.object` is a payment intent

Occurs when a PaymentIntent has started processing.

payment_intent.requires_action `data.object` is a payment intent

Occurs when a PaymentIntent transitions to requires_action state

payment_intent.succeeded `data.object` is a payment intent

Occurs when a PaymentIntent has successfully completed payment.

payment_link.created `data.object` is a payment link

Occurs when a payment link is created.

payment_link.updated `data.object` is a payment link

Occurs when a payment link is updated.

payment_method.attached `data.object` is a payment method

Occurs whenever a new payment method is attached to a customer.

payment_method.automaticaly_updated `data.object` is a payment method

Occurs whenever a payment method's details are automatically updated by the network.

payment_method.detached `data.object` is a payment method

Occurs whenever a payment method is detached from a customer.

payment_method.updated `data.object` is a [payment method](#)

Occurs whenever a payment method is updated via the [PaymentMethod update API](#).

payout.canceled `data.object` is a [payout](#)

Occurs whenever a payout is canceled.

payout.created `data.object` is a [payout](#)

Occurs whenever a payout is created.

payout.failed `data.object` is a [payout](#)

Occurs whenever a payout attempt fails.

payout.paid `data.object` is a [payout](#)

Occurs whenever a payout is *expected* to be available in the destination account. If the payout fails, a `payout.failed` notification is also sent, at a later time.

payout.updated `data.object` is a [payout](#)

Occurs whenever a payout is updated.

person.created `data.object` is a [person](#)

Occurs whenever a person associated with an account is created.

person.deleted `data.object` is a [person](#)

Occurs whenever a person associated with an account is deleted.

person.updated `data.object` is a [person](#)

Occurs whenever a person associated with an account is updated.

plan.created `data.object` is a [plan](#)

Occurs whenever a plan is created.

plan.deleted `data.object` is a [plan](#)

Occurs whenever a plan is deleted.

plan.updated `data.object` is a [plan](#)

Occurs whenever a plan is updated.

price.created `data.object` is a [price](#)

Occurs whenever a price is created.

price.deleted `data.object` is a [price](#)

Occurs whenever a price is deleted.

price.updated `[data.object]` is a [price](#)

Occurs whenever a price is updated.

product.created `[data.object]` is a [product](#)

Occurs whenever a product is created.

product.deleted `[data.object]` is a [product](#)

Occurs whenever a product is deleted.

product.updated `[data.object]` is a [product](#)

Occurs whenever a product is updated.

promotion_code.created `[data.object]` is a [promotion code](#)

Occurs whenever a promotion code is created.

promotion_code.updated `[data.object]` is a [promotion code](#)

Occurs whenever a promotion code is updated.

quote.accepted `[data.object]` is a [quote](#)

Occurs whenever a quote is accepted.

quote.canceled `[data.object]` is a [quote](#)

Occurs whenever a quote is canceled.

quote.created `[data.object]` is a [quote](#)

Occurs whenever a quote is created.

quote.finalized `[data.object]` is a [quote](#)

Occurs whenever a quote is finalized.

radar.early_fraud_warning.created `[data.object]` is an [early fraud warning](#)

Occurs whenever an early fraud warning is created.

radar.early_fraud_warning.updated `[data.object]` is an [early fraud warning](#)

Occurs whenever an early fraud warning is updated.

recipient.created `[data.object]` is a [recipient](#)

Occurs whenever a recipient is created.

recipient.deleted `[data.object]` is a [recipient](#)

Occurs whenever a recipient is deleted.

recipient.updated `[data.object]` is a [recipient](#)

Occurs whenever a recipient is updated.

reporting.report_run.failed `data.object` is a [report run](#)

Occurs whenever a requested `ReportRun` failed to complete.

reporting.report_run.succeeded `data.object` is a [report run](#)

Occurs whenever a requested `ReportRun` completed successfully.

reporting.report_type.updated SELECTION REQUIRED `data.object` is a [report type](#)

Occurs whenever a `ReportType` is updated (typically to indicate that a new day's data has come available). You must create a webhook endpoint which explicitly subscribes to this event type to access it. Webhook endpoints which subscribe to all events will not include this event type.

review.closed `data.object` is a [review](#)

Occurs whenever a review is closed. The review's `reason` field indicates why: `approved`, `disputed`, `refunded`, or `refunded_as_fraud`.

review.opened `data.object` is a [review](#)

Occurs whenever a review is opened.

setup_intent.canceled `data.object` is a [setup intent](#)

Occurs when a SetupIntent is canceled.

setup_intent.created `data.object` is a [setup intent](#)

Occurs when a new SetupIntent is created.

setup_intent.requires_action `data.object` is a [setup intent](#)

Occurs when a SetupIntent is in requires_action state.

setup_intent.setup_failed `data.object` is a [setup intent](#)

Occurs when a SetupIntent has failed the attempt to setup a payment method.

setup_intent.succeeded `data.object` is a [setup intent](#)

Occurs when an SetupIntent has successfully setup a payment method.

sigma.scheduled_query_run.created `data.object` is a [scheduled query run](#)

Occurs whenever a Sigma scheduled query run finishes.

sku.created `data.object` is a [sku](#)

Occurs whenever a SKU is created.

sku.deleted `data.object` is a [sku](#)

Occurs whenever a SKU is deleted.

sku.updated `[data.object]` is a [sku](#)

Occurs whenever a SKU is updated.

source.canceled `[data.object]` is a source (e.g., [card](#))

Occurs whenever a source is canceled.

source.chargeable `[data.object]` is a source (e.g., [card](#))

Occurs whenever a source transitions to chargeable.

source.failed `[data.object]` is a source (e.g., [card](#))

Occurs whenever a source fails.

source.mandate_notification `[data.object]` is a source (e.g., [card](#))

Occurs whenever a source mandate notification method is set to manual.

source.refund_attributes_required `[data.object]` is a source (e.g., [card](#))

Occurs whenever the refund attributes are required on a receiver source to process a refund or a mispayment.

source.transaction.created `[data.object]` is a [source transaction](#)

Occurs whenever a source transaction is created.

source.transaction.updated `[data.object]` is a [source transaction](#)

Occurs whenever a source transaction is updated.

subscription_schedule.aborted `[data.object]` is a [subscription schedule](#)

Occurs whenever a subscription schedule is canceled due to the underlying subscription being canceled because of delinquency.

subscription_schedule.canceled `[data.object]` is a [subscription schedule](#)

Occurs whenever a subscription schedule is canceled.

subscription_schedule.completed `[data.object]` is a [subscription schedule](#)

Occurs whenever a new subscription schedule is completed.

subscription_schedule.created `[data.object]` is a [subscription schedule](#)

Occurs whenever a new subscription schedule is created.

subscription_schedule.expiring `[data.object]` is a [subscription schedule](#)

Occurs 7 days before a subscription schedule will expire.

subscription_schedule.released `data.object` is a [subscription schedule](#)

Occurs whenever a new subscription schedule is released.

subscription_schedule.updated `data.object` is a [subscription schedule](#)

Occurs whenever a subscription schedule is updated.

tax_rate.created `data.object` is a [tax rate](#)

Occurs whenever a new tax rate is created.

tax_rate.updated `data.object` is a [tax rate](#)

Occurs whenever a tax rate is updated.

terminal.reader.action_failed `data.object` is a [reader](#)

Occurs whenever an action sent to a Terminal reader failed.

terminal.reader.action_succeeded `data.object` is a [reader](#)

Occurs whenever an action sent to a Terminal reader was successful.

test_helpers.test_clock.advancing `data.object` is a [test clock](#)

Occurs whenever a test clock starts advancing.

test_helpers.test_clock.created `data.object` is a [test clock](#)

Occurs whenever a test clock is created.

test_helpers.test_clock.deleted `data.object` is a [test clock](#)

Occurs whenever a test clock is deleted.

test_helpers.test_clock.internal_failure `data.object` is a [test clock](#)

Occurs whenever a test clock fails to advance its frozen time.

test_helpers.test_clock.ready `data.object` is a [test clock](#)

Occurs whenever a test clock transitions to a ready status.

topup.canceled `data.object` is a [topup](#)

Occurs whenever a top-up is canceled.

topup.created `data.object` is a [topup](#)

Occurs whenever a top-up is created.

topup.failed `data.object` is a [topup](#)

Occurs whenever a top-up fails.

topup.reversed `data.object` is a [topup](#)

Occurs whenever a top-up is reversed.

topup.succeeded `[data.object]` is a [topup](#)

Occurs whenever a top-up succeeds.

transfer.created `[data.object]` is a [transfer](#)

Occurs whenever a transfer is created.

transfer.failed `[data.object]` is a [transfer](#)

Occurs whenever a transfer failed.

transfer.paid `[data.object]` is a [transfer](#)

Occurs after a transfer is paid. For Instant Payouts, the event will typically be sent within 30 minutes.

transfer.reversed `[data.object]` is a [transfer](#)

Occurs whenever a transfer is reversed, including partial reversals.

transfer.updated `[data.object]` is a [transfer](#)

Occurs whenever a transfer's description or metadata is updated.

Files

This is an object representing a file hosted on Stripe's servers. The file may have been uploaded by yourself using the [create file](#) request (for example, when uploading dispute evidence) or it may have been created by Stripe (for example, the results of a [Sigma scheduled query](#)).

Related guide: [File Upload Guide](#).

The file object

Attributes

id string

Unique identifier for the object.

purpose enum

The [purpose](#) of the uploaded file.

Possible enum values

account_requirement

Additional documentation requirements that can be requested for an account.

additional_verification

Additional verification for custom accounts.

business_icon

A business icon.

business_logo

A business logo.

customer_signature

Customer signature image.

dispute_evidence

Evidence to submit with a dispute response.

document_provider_identity_document

Show 7 more

type string

The type of the file returned (e.g., `csv`, `pdf`, `jpg`, or `png`).

More attributes

[Collapse all](#)

object string, value is "file"

String representing the object's type. Objects of the same type share the same value.

created timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

expires_at timestamp

The time at which the file expires and is no longer available in epoch seconds.

filename string

A filename for the file, suitable for saving to a filesystem.

links list

A list of [file links](#) that point at this file.

✓ **size** integer

The size in bytes of the file object.

✓ **title** string

A user friendly title for the document.

✓ **url** string

The URL from which the file can be downloaded using your live secret API key.

Create a file

To upload a file to Stripe, you'll need to send a request of type `multipart/form-data`. The request should contain the file you would like to upload, as well as the parameters for creating a file.

All of Stripe's officially supported Client libraries should have support for sending `multipart/form-data`.

Parameters

file REQUIRED

A file to upload. The file should follow the specifications of RFC 2388 (which defines file transfers for the `multipart/form-data` protocol).

purpose REQUIRED

The **purpose** of the uploaded file.

Possible enum values

`account_requirement`

Additional documentation requirements that can be requested for an account.

`additional_verification`

Additional verification for custom accounts.

`business_icon`

A business icon.

`business_logo`

A business logo.

`customer_signature`

Customer signature image.

`dispute_evidence`

Evidence to submit with a dispute response.

`identity_document`

A document to verify the identity of an account owner during account provisioning.

`pci_document`

A self-assessment PCI questionnaire.

`tax_document_user_upload`

A user-uploaded tax document.

More parameters

✓ `file_link_data` optional dictionary

Optional parameters to automatically create a [file link](#) for the newly created file.

Returns

Returns the file object.

Retrieve a file

Retrieves the details of an existing file object. Supply the unique file ID from a file, and Stripe will return the corresponding file object. To access file contents, see the [File Upload Guide](#).

Parameters

No parameters.

Returns

Returns a file object if a valid identifier was provided, and returns [an error](#) otherwise.

List all files

Returns a list of the files that your account has access to. The files are returned sorted by creation date, with the most recently created files appearing first.

Parameters

purpose optional

The file purpose to filter queries by. If none is provided, files will not be filtered by purpose.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` files, starting after file `starting_after`. Each entry in the array is a separate file object. If no more files are

available, the resulting array will be empty. This request should never return an error.

File Links

To share the contents of a `File` object with non-Stripe users, you can create a `FileLink`. `FileLink`s contain a URL that can be used to retrieve the contents of the file without authentication.

The file link object

Attributes

`id` string

Unique identifier for the object.

`expires_at` timestamp

Time at which the link expires.

`file` string EXPANDABLE

The file object this link points to.

`metadata` hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

`url` string

The publicly accessible URL to download the file.

More attributes

[Collapse all](#)

✓ `object` string, value is "file_link"

String representing the object's type. Objects of the same type share the same value.

✓ `created` timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ `expired` boolean

Whether this link is already expired.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

Create a file link

Creates a new file link object.

Parameters

file REQUIRED

The ID of the file. The file's `purpose` must be one of the following: `business_icon`, `business_logo`, `customer_signature`, `dispute_evidence`, `finance_report_run`, `identity_document_downloadable`, `pci_document`, `selfie`, `sigma_scheduled_query`, or `tax_document_user_upload`.

expires_at optional

A future timestamp after which the link will no longer be usable.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the file link object if successful, and returns [an error](#) otherwise.

Retrieve a file link

Retrieves the file link with the given ID.

Parameters

No parameters.

Returns

Returns a file link object if a valid identifier was provided, and returns [an error](#) otherwise.

Update a file link

Updates an existing file link object. Expired links can no longer be updated.

Parameters

expires_at optional

A future timestamp after which the link will no longer be usable, or `now` to expire the link immediately.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the file link object if successful, and returns [an error](#) otherwise.

List all file links

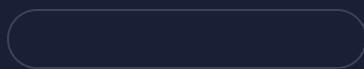
Returns a list of file links.

Parameters

[Collapse all](#)

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **expired** optional

Filter links by their expiration status. By default, all links are returned.

✓ **file** optional

Only return links for the given file.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` file links, starting after file link `starting_after`. Each entry in the array is a separate file link object. If no more file links are available, the resulting array will be empty. This request should never return an error.

Mandates

A Mandate is a record of the permission a customer has given you to debit their payment method.

The Mandates object

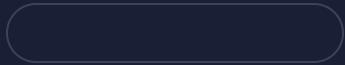
Attributes

id string

Unique identifier for the object.

customer_acceptance hash

Details about the customer's acceptance of the mandate.



payment_method string EXPANDABLE

ID of the payment method associated with this mandate.

payment_method_details hash

Additional mandate information specific to the payment method type.



status enum

The status of the mandate, which indicates whether it can be used to initiate a payment.

Possible enum values

active

The mandate can be used to initiate a payment.

inactive

The mandate was rejected, revoked, or previously used, and may not be used to initiate future payments.

pending

The mandate is newly created and is not yet active or inactive.

type enum

The type of the mandate.

Possible enum values

multi_use

Represents permission given for multiple payments.

single_use

More attributes

[Collapse all](#)

✓ **object** string, value is "mandate"

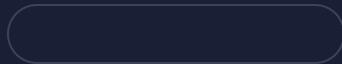
String representing the object's type. Objects of the same type share the same value.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **multi_use** hash

If this is a `multi_use` mandate, this hash contains details about the mandate.



✓ **single_use** hash

If this is a `single_use` mandate, this hash contains details about the mandate.



Retrieve a Mandate

Retrieves a Mandate object.

Parameters

No parameters.

Returns

Returns a Mandate object.

PaymentIntents

A PaymentIntent guides you through the process of collecting a payment from your customer. We recommend that you create exactly one PaymentIntent for each order or customer session in your system. You can reference the PaymentIntent later to see the history of payment attempts for a particular session.

A PaymentIntent transitions through [multiple statuses](#) throughout its lifetime as it interfaces with Stripe.js to perform authentication flows and ultimately creates at most one successful charge.

Related guide: [Payment Intents API](#).

The PaymentIntent object

Attributes

id string RETRIEVABLE WITH PUBLISHABLE KEY

Unique identifier for the object.

amount integer RETRIEVABLE WITH PUBLISHABLE KEY

Amount intended to be collected by this PaymentIntent. A positive integer representing how much to charge in the [smallest currency unit](#) (e.g., 100 cents to charge \$1.00 or 100 to charge ¥100, a zero-decimal currency). The minimum amount is \$0.50 US or [equivalent in charge currency](#). The amount value supports up to eight digits (e.g., a value of 99999999 for a USD charge of \$999.999.99).

automatic_payment_methods hash RETRIEVABLE WITH PUBLISHABLE KEY

Settings to configure compatible payment methods from the [Stripe Dashboard](#)

charges list

Charges that were created by this PaymentIntent, if any.

client_secret string RETRIEVABLE WITH PUBLISHABLE KEY

The client secret of this PaymentIntent. Used for client-side retrieval using a publishable key.

The client secret can be used to complete a payment from your frontend. It should not be stored, logged, or exposed to anyone other than the customer. Make sure that you have TLS enabled on any page that includes the client secret.

Refer to our docs to [accept a payment](#) and learn about how `client_secret` should be handled.

currency currency RETRIEVABLE WITH PUBLISHABLE KEY

Three-letter ISO currency code, in lowercase. Must be a [supported currency](#).

customer string EXPANDABLE

ID of the Customer this PaymentIntent belongs to, if one exists.

Payment methods attached to other Customers cannot be used with this PaymentIntent.

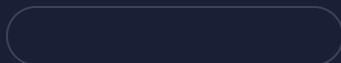
If present in combination with `setup_future_usage`, this PaymentIntent's payment method will be attached to the Customer after the PaymentIntent has been confirmed and any required actions from the user are complete.

description string RETRIEVABLE WITH PUBLISHABLE KEY

An arbitrary string attached to the object. Often useful for displaying to users.

last_payment_error hash RETRIEVABLE WITH PUBLISHABLE KEY

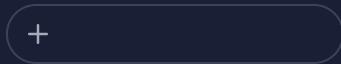
The payment error encountered in the previous PaymentIntent confirmation. It will be cleared if the PaymentIntent is later updated for any reason.

**metadata** hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. For more information, see the [documentation](#).

next_action hash RETRIEVABLE WITH PUBLISHABLE KEY

If present, this property tells you what actions you need to take in order for your customer to fulfill a payment using the provided source.

**payment_method** string EXPANDABLE RETRIEVABLE WITH PUBLISHABLE KEY

ID of the payment method used in this PaymentIntent.

payment_method_types array containing strings RETRIEVABLE WITH PUBLISHABLE KEY

The list of payment method types (e.g. card) that this PaymentIntent is allowed to use.

receipt_email string RETRIEVABLE WITH PUBLISHABLE KEY

Email address that the receipt for the resulting payment will be sent to. If `receipt_email` is specified for a payment in live mode, a receipt will be sent regardless of your [email settings](#).

setup_future_usage enum RETRIEVABLE WITH PUBLISHABLE KEY

Indicates that you intend to make future payments with this PaymentIntent's payment method.

Providing this parameter will [attach the payment method](#) to the PaymentIntent's Customer, if present, after the PaymentIntent is confirmed and any required actions from the user are complete. If no Customer was provided, the payment method can still be [attached](#) to a Customer after the transaction completes.

When processing card payments, Stripe also uses `setup_future_usage` to dynamically optimize your payment flow and comply with regional legislation and network rules, such as [SCA](#).

Possible enum values

`on_session`

Use `on_session` if you intend to only reuse the payment method when your customer is present in your checkout flow.

`off_session`

Use `off_session` if your customer may or may not be present in your checkout flow.

shipping hash RETRIEVABLE WITH PUBLISHABLE KEY

Shipping information for this PaymentIntent.

+

statement_descriptor string

For non-card charges, you can use this value as the complete description that appears on your customers' statements. Must contain at least one letter, maximum 22 characters.

statement_descriptor_suffix string

Provides information about a card payment that customers see on their statements. Concatenated with the prefix (shortened descriptor) or statement descriptor that's set on the account to form the complete statement descriptor. Maximum 22 characters for the concatenated descriptor.

status string RETRIEVABLE WITH PUBLISHABLE KEY

Status of this PaymentIntent, one of `requires_payment_method`, `requires_confirmation`, `requires_action`, `processing`, `requires_capture`, `canceled`, or `succeeded`. Read more about each PaymentIntent [status](#).

More attributes

[Collapse all](#)

✓ **object** string, value is "payment_intent" RETRIEVABLE WITH PUBLISHABLE KEY

String representing the object's type. Objects of the same type share the same value.

✓ **amount_capturable** integer

Amount that can be captured from this PaymentIntent.

✓ **amount_details** hash

Details about items included in the amount

+

✓ **amount_received** integer

Amount that was collected by this PaymentIntent.

✓ **application** string EXPANDABLE "APPLICATION" CONNECT ONLY

ID of the Connect application that created the PaymentIntent.

✓ **application_fee_amount** integer CONNECT ONLY

The amount of the application fee (if any) that will be requested to be applied to the payment and transferred to the application owner's Stripe account. The amount of the application fee collected will be capped at the total payment amount. For more information, see the PaymentIntents [use case for connected accounts](#).

✓ **canceled_at** timestamp RETRIEVABLE WITH PUBLISHABLE KEY

Populated when `status` is `canceled`, this is the time at which the PaymentIntent was canceled. Measured in seconds since the Unix epoch.

✓ **cancellation_reason** string RETRIEVABLE WITH PUBLISHABLE KEY

Reason for cancellation of this PaymentIntent, either user-provided (`duplicate`, `fraudulent`, `requested_by_customer`, or `abandoned`) or generated by Stripe internally (`failed_invoice`, `void_invoice`, or `automatic`).

✓ **capture_method** enum RETRIEVABLE WITH PUBLISHABLE KEY

Controls when the funds will be captured from the customer's account.

Possible enum values

`automatic`

(Default) Stripe automatically captures funds when the customer authorizes the payment.

`manual`

Place a hold on the funds when the customer authorizes the payment, but [don't capture the funds until later](#). (Not all payment methods support this.)

✓ **confirmation_method** enum RETRIEVABLE WITH PUBLISHABLE KEY

Possible enum values

`automatic`

(Default) PaymentIntent can be confirmed using a publishable key. After `next_action`s are handled, no additional confirmation is required to complete the payment.

manual

All payment attempts must be made using a secret key. The PaymentIntent returns to the `requires_confirmation` state after handling `next_action`s, and requires your server to initiate each payment attempt with an explicit confirmation.

✓ **created** timestamp RETRIEVABLE WITH PUBLISHABLE KEY

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **invoice** string EXPANDABLE

ID of the invoice that created this PaymentIntent, if it exists.

✓ **livemode** boolean RETRIEVABLE WITH PUBLISHABLE KEY

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **on_behalf_of** string EXPANDABLE CONNECT ONLY

The account (if any) for which the funds of the PaymentIntent are intended. See the PaymentIntents [use case for connected accounts](#) for details.

✓ **payment_method_options** hash

Payment-method-specific configuration for this PaymentIntent.

✓ **processing** hash RETRIEVABLE WITH PUBLISHABLE KEY

If present, this property tells you about the processing state of the payment.

✓ **review** string EXPANDABLE

ID of the review associated with this PaymentIntent, if any.

✓ **transfer_data** hash CONNECT ONLY

The data with which to automatically create a Transfer when the payment is finalized. See the PaymentIntents [use case for connected accounts](#) for details.

✓ **transfer_group** string CONNECT ONLY

A string that identifies the resulting payment as part of a group. See the PaymentIntents [use case for connected accounts](#) for details.

Create a PaymentIntent

Creates a PaymentIntent object.

After the PaymentIntent is created, attach a payment method and [confirm](#) to continue the payment. You can read more about the different payment flows available via the Payment Intents API [here](#).

When `confirm=true` is used during creation, it is equivalent to creating and confirming the PaymentIntent in the same call. You may use any parameters available in the [confirm API](#) when `confirm=true` is supplied.

Parameters

amount REQUIRED

Amount intended to be collected by this PaymentIntent. A positive integer representing how much to charge in the [smallest currency unit](#) (e.g., 100 cents to charge \$1.00 or 100 to charge ¥100, a zero-decimal currency). The minimum amount is \$0.50 US or [equivalent in charge currency](#). The amount value supports up to eight digits (e.g., a value of 99999999 for a USD charge of \$999,999.99).

currency REQUIRED

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

automatic_payment_methods optional dictionary

When enabled, this PaymentIntent will accept payment methods that you have enabled in the Dashboard and are compatible with this PaymentIntent's other parameters.



confirm optional

Set to `true` to attempt to [confirm](#) this PaymentIntent immediately. This parameter defaults to `false`. When creating and confirming a PaymentIntent at the same time, parameters available in the [confirm API](#) may also be provided.

customer optional

ID of the Customer this PaymentIntent belongs to, if one exists.

Payment methods attached to other Customers cannot be used with this PaymentIntent.

If present in combination with [setup_future_usage](#), this PaymentIntent's payment method will be attached to the Customer after the PaymentIntent has been confirmed and any required actions from the user are complete.

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

off_session optional ONLY WHEN CONFIRM=TRUE

Set to `true` to indicate that the customer is not in your checkout flow during this payment attempt, and therefore is unable to authenticate. This parameter is intended for scenarios where you collect card details and **charge them later**. This parameter can only be used with `confirm=true`.

payment_method optional

ID of the payment method (a PaymentMethod, Card, or **compatible Source** object) to attach to this PaymentIntent.

If this parameter is omitted with `confirm=true`, `customer.default_source` will be attached as this PaymentIntent's payment instrument to improve the migration experience for users of the Charges API. We recommend that you explicitly provide the `payment_method` going forward.

payment_method_types optional

The list of **payment method types** that this PaymentIntent is allowed to use. If this is not provided, defaults to `["card"]`. Valid payment method types include: `acss_debit`, `afterpay_clearpay`, `alipay`, `au_becs_debit`, `bacs_debit`, `bancontact`, `boleto`, `card`, `card_present`, `eps`, `fpx`, `giropay`, `grabpay`, `ideal`, `klarna`, `konbini`, `oxxo`, `p24`, `paynow`, `sepa_debit`, `sofort`, `us_bank_account`, and `wechat_pay`.

receipt_email optional

Email address that the receipt for the resulting payment will be sent to. If `receipt_email` is specified for a payment in live mode, a receipt will be sent regardless of your **email settings**.

setup_future_usage optional enum

Indicates that you intend to make future payments with this PaymentIntent's payment method.

Providing this parameter will **attach the payment method** to the PaymentIntent's Customer, if present, after the PaymentIntent is confirmed and any required actions from the user are complete. If no Customer was provided, the payment method can still be **attached** to a Customer after the transaction completes.

When processing card payments, Stripe also uses `setup_future_usage` to dynamically optimize your payment flow and comply with regional legislation and network rules, such as **SCA**.

Possible enum values

on_session

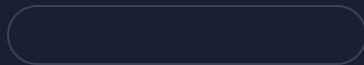
Use `on_session` if you intend to only reuse the payment method when your customer is present in your checkout flow.

off_session

Use `off_session` if your customer may or may not be present in your checkout flow.

shipping optional dictionary

Shipping information for this PaymentIntent.



statement_descriptor optional

For non-card charges, you can use this value as the complete description that appears on your customers' statements. Must contain at least one letter, maximum 22 characters.

statement_descriptor_suffix optional

Provides information about a card payment that customers see on their statements.

Concatenated with the prefix (shortened descriptor) or statement descriptor that's set on the account to form the complete statement descriptor. Maximum 22 characters for the concatenated descriptor.

More parameters

[Collapse all](#)

application_fee_amount optional CONNECT ONLY

The amount of the application fee (if any) that will be requested to be applied to the payment and transferred to the application owner's Stripe account. The amount of the application fee collected will be capped at the total payment amount. For more information, see the PaymentIntents [use case for connected accounts](#).

capture_method optional enum

Controls when the funds will be captured from the customer's account.

Possible enum values

automatic

(Default) Stripe automatically captures funds when the customer authorizes the payment.

manual

Place a hold on the funds when the customer authorizes the payment, but [don't capture the funds until later](#). (Not all payment methods support this.)

✓ **confirmation_method** optional enum

Possible enum values

`automatic`

(Default) PaymentIntent can be confirmed using a publishable key. After `next_action`s are handled, no additional confirmation is required to complete the payment.

`manual`

All payment attempts must be made using a secret key. The PaymentIntent returns to the `requires_confirmation` state after handling `next_action`s, and requires your server to initiate each payment attempt with an explicit confirmation.

✓ **error_on_requires_action** optional ONLY WHEN CONFIRM=TRUE

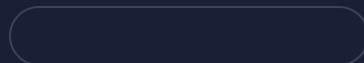
Set to `true` to fail the payment attempt if the PaymentIntent transitions into `requires_action`. This parameter is intended for simpler integrations that do not handle customer actions, like [saving cards without authentication](#). This parameter can only be used with `confirm=true`.

✓ **mandate** optional ONLY WHEN CONFIRM=TRUE

ID of the mandate to be used for this payment. This parameter can only be used with `confirm=true`.

✓ **mandate_data** optional dictionary ONLY WHEN CONFIRM=TRUE

This hash contains details about the Mandate to create. This parameter can only be used with `confirm=true`.

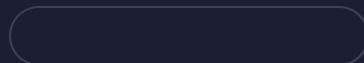


✓ **on_behalf_of** optional CONNECT ONLY

The Stripe account ID for which these funds are intended. For details, see the PaymentIntents [use case for connected accounts](#).

✓ **payment_method_data** optional dictionary

If provided, this hash will be used to create a PaymentMethod. The new PaymentMethod will appear in the `payment_method` property on the PaymentIntent.



✓ **payment_method_options** optional dictionary

Payment-method-specific configuration for this PaymentIntent.

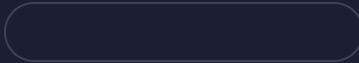


✓ **return_url** optional ONLY WHEN CONFIRM=TRUE

The URL to redirect your customer back to after they authenticate or cancel their payment on the payment method's app or site. If you'd prefer to redirect to a mobile application, you can alternatively supply an application URI scheme. This parameter can only be used with `confirm=true`.

✓ **transfer_data** optional dictionary CONNECT ONLY

The parameters used to automatically create a Transfer when the payment succeeds. For more information, see the PaymentIntents [use case for connected accounts](#).



✓ **transfer_group** optional CONNECT ONLY

A string that identifies the resulting payment as part of a group. See the PaymentIntents [use case for connected accounts](#) for details.

✓ **use_stripe_sdk** optional

Set to `true` only when using manual confirmation and the iOS or Android SDKs to handle additional authentication steps.

Returns

Returns a PaymentIntent object.

Retrieve a PaymentIntent

Retrieves the details of a PaymentIntent that has previously been created.

Client-side retrieval using a publishable key is allowed when the `client_secret` is provided in the query string.

When retrieved with a publishable key, only a subset of properties will be returned. Please refer to the [payment intent](#) object reference for more details.

Parameters

client_secret REQUIRED IF USING PUBLISHABLE KEY

The client secret of the PaymentIntent. Required if a publishable key is used to retrieve the source.

Returns

Returns a PaymentIntent if a valid identifier was provided.

Update a PaymentIntent

Updates properties on a PaymentIntent object without confirming.

Depending on which properties you update, you may need to confirm the PaymentIntent again. For example, updating the `payment_method` will always require you to confirm the PaymentIntent again. If you prefer to update and confirm at the same time, we recommend updating properties via the [confirm API](#) instead.

Parameters

amount optional

Amount intended to be collected by this PaymentIntent. A positive integer representing how much to charge in the [smallest currency unit](#) (e.g., 100 cents to charge \$1.00 or 100 to charge ¥100, a zero-decimal currency). The minimum amount is \$0.50 US or [equivalent in charge currency](#). The amount value supports up to eight digits (e.g., a value of 99999999 for a USD charge of \$999,999.99).

currency optional

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

customer optional

ID of the Customer this PaymentIntent belongs to, if one exists.

Payment methods attached to other Customers cannot be used with this PaymentIntent.

If present in combination with [setup_future_usage](#), this PaymentIntent's payment method will be attached to the Customer after the PaymentIntent has been confirmed and any required actions from the user are complete.

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset

by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

payment_method optional

ID of the payment method (a PaymentMethod, Card, or [compatible Source](#) object) to attach to this PaymentIntent.

payment_method_types optional

The list of payment method types (e.g. card) that this PaymentIntent is allowed to use.

receipt_email optional

Email address that the receipt for the resulting payment will be sent to. If `receipt_email` is specified for a payment in live mode, a receipt will be sent regardless of your [email settings](#).

setup_future_usage optional enum

Indicates that you intend to make future payments with this PaymentIntent's payment method.

Providing this parameter will [attach the payment method](#) to the PaymentIntent's Customer, if present, after the PaymentIntent is confirmed and any required actions from the user are complete. If no Customer was provided, the payment method can still be [attached](#) to a Customer after the transaction completes.

When processing card payments, Stripe also uses `setup_future_usage` to dynamically optimize your payment flow and comply with regional legislation and network rules, such as [SCA](#).

If `setup_future_usage` is already set and you are performing a request using a publishable key, you may only update the value from `on_session` to `off_session`.

Possible enum values

`on_session`

Use `on_session` if you intend to only reuse the payment method when your customer is present in your checkout flow.

`off_session`

Use `off_session` if your customer may or may not be present in your checkout flow.

shipping optional dictionary

Shipping information for this PaymentIntent.

statement_descriptor optional

For non-card charges, you can use this value as the complete description that appears on your customers' statements. Must contain at least one letter, maximum 22 characters.

statement_descriptor_suffix optional

Provides information about a card payment that customers see on their statements. Concatenated with the prefix (shortened descriptor) or statement descriptor that's set on the account to form the complete statement descriptor. Maximum 22 characters for the concatenated descriptor.

More parameters

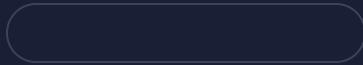
[Collapse all](#)

✓ **application_fee_amount** optional CONNECT ONLY

The amount of the application fee (if any) that will be requested to be applied to the payment and transferred to the application owner's Stripe account. The amount of the application fee collected will be capped at the total payment amount. For more information, see the PaymentIntents [use case for connected accounts](#).

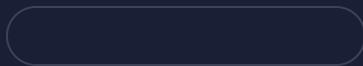
✓ **payment_method_data** optional dictionary

If provided, this hash will be used to create a PaymentMethod. The new PaymentMethod will appear in the **payment_method** property on the PaymentIntent.



✓ **payment_method_options** optional dictionary

Payment-method-specific configuration for this PaymentIntent.



✓ **transfer_data** optional dictionary CONNECT ONLY

The parameters used to automatically create a Transfer when the payment succeeds. For more information, see the PaymentIntents [use case for connected accounts](#).

✓ **transfer_group** optional CONNECT ONLY

A string that identifies the resulting payment as part of a group. `transfer_group` may only be provided if it has not been set. See the PaymentIntents [use case for connected accounts](#) for details.

Returns

Returns a PaymentIntent object.

Confirm a PaymentIntent

Confirm that your customer intends to pay with current or provided payment method. Upon confirmation, the PaymentIntent will attempt to initiate a payment.

If the selected payment method requires additional authentication steps, the PaymentIntent will transition to the `requires_action` status and suggest additional actions via `next_action`. If payment fails, the PaymentIntent will transition to the `requires_payment_method` status. If payment succeeds, the PaymentIntent will transition to the `succeeded` status (or `requires_capture`, if `capture_method` is set to `manual`).

If the `confirmation_method` is `automatic`, payment may be attempted using our [client SDKs](#) and the PaymentIntent's `client_secret`. After `next_action`s are handled by the client, no additional confirmation is required to complete the payment.

If the `confirmation_method` is `manual`, all payment attempts must be initiated using a secret key. If any actions are required for the payment, the PaymentIntent will return to the `requires_confirmation` state after those actions are completed. Your server needs to then explicitly re-confirm the PaymentIntent to initiate the next payment attempt. Read the [expanded documentation](#) to learn more about manual confirmation.

Parameters

`payment_method` optional

ID of the payment method (a `PaymentMethod`, `Card`, or [compatible Source](#) object) to attach to this PaymentIntent.

`receipt_email` optional

Email address that the receipt for the resulting payment will be sent to. If `receipt_email` is specified for a payment in live mode, a receipt will be sent regardless of your [email settings](#).

`setup_future_usage` optional enum

Indicates that you intend to make future payments with this PaymentIntent's payment method.

Providing this parameter will [attach the payment method](#) to the PaymentIntent's Customer, if present, after the PaymentIntent is confirmed and any required actions from the user are complete. If no Customer was provided, the payment method can still be [attached](#) to a Customer after the transaction completes.

When processing card payments, Stripe also uses `setup_future_usage` to dynamically optimize your payment flow and comply with regional legislation and network rules, such as [SCA](#).

If `setup_future_usage` is already set and you are performing a request using a publishable key, you may only update the value from `on_session` to `off_session`.

Possible enum values

`on_session`

Use `on_session` if you intend to only reuse the payment method when your customer is present in your checkout flow.

`off_session`

Use `off_session` if your customer may or may not be present in your checkout flow.

shipping optional dictionary

Shipping information for this PaymentIntent.

More parameters

✓ **error_on_requires_action** optional

Set to `true` to fail the payment attempt if the PaymentIntent transitions into `requires_action`. This parameter is intended for simpler integrations that do not handle customer actions, like [saving cards without authentication](#).

✓ **mandate** optional SECRET KEY ONLY

ID of the mandate to be used for this payment.

mandate_data optional dictionary

This hash contains details about the Mandate to create

off_session optional SECRET KEY ONLY

Set to `true` to indicate that the customer is not in your checkout flow during this payment attempt, and therefore is unable to authenticate. This parameter is intended for scenarios where you collect card details and [charge them later](#).

payment_method_data optional dictionary

If provided, this hash will be used to create a PaymentMethod. The new PaymentMethod will appear in the `payment_method` property on the PaymentIntent.

payment_method_options optional dictionary SECRET KEY ONLY

Payment-method-specific configuration for this PaymentIntent.

✓ **payment_method_types** optional SECRET KEY ONLY

The list of payment method types (e.g. card) that this PaymentIntent is allowed to use.

✓ **return_url** optional

The URL to redirect your customer back to after they authenticate or cancel their payment on the payment method's app or site. If you'd prefer to redirect to a mobile application, you can alternatively supply an application URI scheme. This parameter is only used for cards and other redirect-based payment methods.

✓ **use_stripe_sdk** optional

Set to `true` only when using manual confirmation and the iOS or Android SDKs to handle additional authentication steps.

Returns

Returns the resulting PaymentIntent after all possible transitions are applied.

Capture a PaymentIntent

Capture the funds of an existing uncaptured PaymentIntent when its status is `requires_capture`.

Uncaptured PaymentIntents will be canceled a set number of days after they are created (7 by default).

Learn more about [separate authorization and capture](#).

Parameters

amount_to_capture optional

The amount to capture from the PaymentIntent, which must be less than or equal to the original amount. Any additional amount will be automatically refunded. Defaults to the full `amount_capturable` if not provided.

More parameters

✓ **application_fee_amount** optional CONNECT ONLY

The amount of the application fee (if any) that will be requested to be applied to the payment and transferred to the application owner's Stripe account. The amount of the application fee

collected will be capped at the total payment amount. For more information, see the PaymentIntents [use case for connected accounts](#).

✓ **statement_descriptor** optional

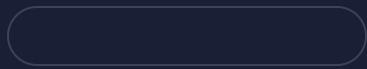
For non-card charges, you can use this value as the complete description that appears on your customers' statements. Must contain at least one letter, maximum 22 characters.

✓ **statement_descriptor_suffix** optional

Provides information about a card payment that customers see on their statements. Concatenated with the prefix (shortened descriptor) or statement descriptor that's set on the account to form the complete statement descriptor. Maximum 22 characters for the concatenated descriptor.

✓ **transfer_data** optional dictionary CONNECT ONLY

The parameters used to automatically create a Transfer when the payment is captured. For more information, see the PaymentIntents [use case for connected accounts](#).



Returns

Returns a PaymentIntent object with `status="succeeded"` if the PaymentIntent was capturable. Returns an error if the PaymentIntent was not capturable or an invalid amount to capture was provided.

Cancel a PaymentIntent

A PaymentIntent object can be canceled when it is in one of these statuses:

`requires_payment_method`, `requires_capture`, `requires_confirmation`,
`requires_action`, or `processing`.

Once canceled, no additional charges will be made by the PaymentIntent and any operations on the PaymentIntent will fail with an error. For PaymentIntents with `status='requires_capture'`, the remaining `amount_capturable` will automatically be refunded.

You cannot cancel the PaymentIntent for a Checkout Session. [Expire the Checkout Session](#) instead

Parameters

cancellation_reason optional

Reason for canceling this PaymentIntent. Possible values are `duplicate`, `fraudulent`, `requested_by_customer`, or `abandoned`

Returns

Returns a PaymentIntent object if the cancellation succeeded. Returns an error if the PaymentIntent has already been canceled or is not in a cancelable state.

List all PaymentIntents

Returns a list of PaymentIntents.

Parameters

customer optional

Only return PaymentIntents for the customer specified by this customer ID.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` `PaymentIntents`, starting after `PaymentIntent` `starting_after`. Each entry in the array is a separate `PaymentIntent` object. If no more `PaymentIntents` are available, the resulting array will be empty. This request should never return an error.

Increment an authorization Terminal only

Perform an incremental authorization on an eligible `PaymentIntent`. To be eligible, the `PaymentIntent`'s status must be `requires_capture` and `incremental_authorization_supported` must be `true`.

Incremental authorizations attempt to increase the authorized amount on your customer's card to the new, higher `amount` provided. As with the initial authorization, incremental authorizations may be declined. A single `PaymentIntent` can call this endpoint multiple times to further increase the authorized amount.

If the incremental authorization succeeds, the `PaymentIntent` object is returned with the updated `amount`. If the incremental authorization fails, a `card_declined` error is returned, and no fields on the `PaymentIntent` or `Charge` are updated. The `PaymentIntent` object remains captureable for the previously authorized amount.

Each `PaymentIntent` can have a maximum of 10 incremental authorization attempts, including declines. Once captured, a `PaymentIntent` can no longer be incremented.

Learn more about [incremental authorizations](#).

Parameters

amount REQUIRED

The updated total amount you intend to collect from the cardholder. This amount must be greater than the currently authorized amount.

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset

by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

[Collapse all](#)

✓ `application_fee_amount` optional CONNECT ONLY

The amount of the application fee (if any) that will be requested to be applied to the payment and transferred to the application owner's Stripe account. The amount of the application fee collected will be capped at the total payment amount. For more information, see the [PaymentIntents use case for connected accounts](#).

`transfer_data` optional dictionary CONNECT ONLY

The parameters used to automatically create a Transfer when the payment is captured. For more information, see the [PaymentIntents use case for connected accounts](#).

Returns

Returns a `PaymentIntent` object with the updated amount if the incremental authorization succeeded. Returns an error if the incremental authorization failed or the `PaymentIntent` isn't eligible for incremental authorizations.

Search PaymentIntents

Search for `PaymentIntents` you've previously created using Stripe's [Search Query Language](#). Don't use search in read-after-write flows where strict consistency is necessary. Under normal operating conditions, data is searchable in less than a minute. Occasionally, propagation of new or updated data can be up to an hour behind during outages. Search functionality is not available to merchants in India.

Parameters

`query` REQUIRED

The search query string. See [search query language](#) and the list of supported [query fields for payment intents](#).

`limit` optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

page optional

A cursor for pagination across multiple pages of results. Don't include this parameter on the first call. Use the next_page value returned in a previous response to request subsequent results.

Returns

A dictionary with a `data` property that contains an array of up to `limit` PaymentIntents. If no objects match the query, the resulting array will be empty. See the related guide on [expanding properties in lists](#).

Verify microdeposits on a PaymentIntent

Verifies microdeposits on a PaymentIntent object.

Parameters

client_secret REQUIRED IF USING PUBLISHABLE KEY

The client secret of the PaymentIntent.

amounts optional

Two positive integers, in cents, equal to the values of the microdeposits sent to the bank account.

descriptor_code optional

A six-character code starting with SM present in the microdeposit sent to the bank account.

Returns

Returns a PaymentIntent object.

Reconcile a customer_balance PaymentIntent

Manually reconcile the remaining amount for a customer_balance PaymentIntent.

This can be used when the cash balance for a customer in manual reconciliation mode received funds.

Parameters

amount optional

Amount intended to be applied to this PaymentIntent from the customer's cash balance.

A positive integer representing how much to charge in the **smallest currency unit** (e.g., 100 cents to charge \$1.00 or 100 to charge ¥100, a zero-decimal currency).

The maximum amount is the amount of the PaymentIntent.

When omitted, the amount defaults to the remaining amount requested on the PaymentIntent.

currency optional

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

Returns

Returns a PaymentIntent object.

SetupIntents

A SetupIntent guides you through the process of setting up and saving a customer's payment credentials for future payments. For example, you could use a SetupIntent to set up and save your customer's card without immediately collecting a payment. Later, you can use [PaymentIntents](#) to drive the payment flow.

Create a SetupIntent as soon as you're ready to collect your customer's payment credentials. Do not maintain long-lived, unconfirmed SetupIntents as they may no longer be valid. The SetupIntent then transitions through multiple [statuses](#) as it guides you through the setup process.

Successful SetupIntents result in payment credentials that are optimized for future payments. For example, cardholders in [certain regions](#) may need to be run through [Strong Customer Authentication](#) at the time of payment method collection in order to streamline later [off-session payments](#). If the SetupIntent is used with a [Customer](#), upon success, it will automatically attach the resulting

payment method to that Customer. We recommend using `SetupIntents` or `setup_future_usage` on `PaymentIntents` to save payment methods in order to prevent saving invalid or unoptimized payment methods.

By using `SetupIntents`, you ensure that your customers experience the minimum set of required friction, even as regulations change over time.

Related guide: [Setup Intents API](#).

The `SetupIntent` object

Attributes

id string RETRIEVABLE WITH PUBLISHABLE KEY

Unique identifier for the object.

client_secret string RETRIEVABLE WITH PUBLISHABLE KEY

The client secret of this `SetupIntent`. Used for client-side retrieval using a publishable key.

The client secret can be used to complete payment setup from your frontend. It should not be stored, logged, or exposed to anyone other than the customer. Make sure that you have TLS enabled on any page that includes the client secret.

customer string EXPANDABLE

ID of the Customer this `SetupIntent` belongs to, if one exists.

If present, the `SetupIntent`'s payment method will be attached to the Customer on successful setup. Payment methods attached to other Customers cannot be used with this `SetupIntent`.

description string RETRIEVABLE WITH PUBLISHABLE KEY

An arbitrary string attached to the object. Often useful for displaying to users.

last_setup_error hash RETRIEVABLE WITH PUBLISHABLE KEY

The error encountered in the previous `SetupIntent` confirmation.



metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

next_action hash RETRIEVABLE WITH PUBLISHABLE KEY

If present, this property tells you what actions you need to take in order for your customer to continue payment setup.

payment_method string EXPANDABLE RETRIEVABLE WITH PUBLISHABLE KEY

ID of the payment method used with this SetupIntent.

payment_method_types array containing strings RETRIEVABLE WITH PUBLISHABLE KEY

The list of payment method types (e.g. card) that this SetupIntent is allowed to set up.

status string RETRIEVABLE WITH PUBLISHABLE KEY

Status of this SetupIntent, one of `requires_payment_method`, `requires_confirmation`, `requires_action`, `processing`, `canceled`, or `succeeded`.

usage string RETRIEVABLE WITH PUBLISHABLE KEY

Indicates how the payment method is intended to be used in the future.

Use `on_session` if you intend to only reuse the payment method when the customer is in your checkout flow. Use `off_session` if your customer may or may not be in your checkout flow. If not provided, this value defaults to `off_session`.

More attributes

✓ **object** string, value is "setup_intent" RETRIEVABLE WITH PUBLISHABLE KEY

String representing the object's type. Objects of the same type share the same value.

✓ **application** string EXPANDABLE "APPLICATION" CONNECT ONLY

ID of the Connect application that created the SetupIntent.

✓ **cancellation_reason** string RETRIEVABLE WITH PUBLISHABLE KEY

Reason for cancellation of this SetupIntent, one of `abandoned`, `requested_by_customer`, or `duplicate`.

✓ **created** timestamp RETRIEVABLE WITH PUBLISHABLE KEY

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **latest_attempt** string EXPANDABLE

The most recent SetupAttempt for this SetupIntent.

✓ **livemode** boolean RETRIEVABLE WITH PUBLISHABLE KEY

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **mandate** string [EXPANDABLE](#)

ID of the multi use Mandate generated by the SetupIntent.

✓ **on_behalf_of** string [EXPANDABLE](#) CONNECT ONLY

The account (if any) for which the setup is intended.

✓ **payment_method_options** hash

Payment-method-specific configuration for this SetupIntent.



✓ **single_use_mandate** string [EXPANDABLE](#)

ID of the single_use Mandate generated by the SetupIntent.

Create a SetupIntent

Creates a SetupIntent object.

After the SetupIntent is created, attach a payment method and [confirm](#) to collect any required permissions to charge the payment method later.

Parameters

confirm optional

Set to `true` to attempt to confirm this SetupIntent immediately. This parameter defaults to `false`. If the payment method attached is a card, a `return_url` may be provided in case additional authentication is required.

customer optional

ID of the Customer this SetupIntent belongs to, if one exists.

If present, the SetupIntent's payment method will be attached to the Customer on successful setup. Payment methods attached to other Customers cannot be used with this SetupIntent.

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

payment_method optional

ID of the payment method (a PaymentMethod, Card, or saved Source object) to attach to this SetupIntent.

payment_method_types optional

The list of **payment method types** that this SetupIntent is allowed to set up. If this is not provided, defaults to `["card"]`. Valid payment method types include: `acss_debit`, `au_becs_debit`, `bacs_debit`, `bancontact`, `boleto`, `card`, `card_present`, `ideal`, `sepa_debit`, `sofort`, and `us_bank_account`.

usage optional enum

Indicates how the payment method is intended to be used in the future. If not provided, this value defaults to `off_session`.

Possible enum values

`on_session`

Use `on_session` if you intend to only reuse the payment method when the customer is in your checkout flow.

`off_session`

Use `off_session` if your customer may or may not be in your checkout flow.

More parameters

✓ **mandate_data** optional dictionary ONLY WHEN CONFIRM=TRUE

This hash contains details about the Mandate to create. This parameter can only be used with `confirm=true`.

✓ **on_behalf_of** optional CONNECT ONLY

The Stripe account ID for which this SetupIntent is created.

✓ **payment_method_options** optional dictionary

Payment-method-specific configuration for this SetupIntent.

✓ **return_url** optional ONLY WHEN CONFIRM=TRUE

The URL to redirect your customer back to after they authenticate or cancel their payment on the payment method's app or site. If you'd prefer to redirect to a mobile application, you can alternatively supply an application URI scheme. This parameter can only be used with `confirm=true`.

✓ **single_use** optional dictionary

If this hash is populated, this SetupIntent will generate a single_use Mandate on success.

Returns

Returns a SetupIntent object.

Retrieve a SetupIntent

Retrieves the details of a SetupIntent that has previously been created.

Client-side retrieval using a publishable key is allowed when the `client_secret` is provided in the query string.

When retrieved with a publishable key, only a subset of properties will be returned. Please refer to the [SetupIntent](#) object reference for more details.

Parameters

client_secret REQUIRED IF USING PUBLISHABLE KEY

The client secret of the SetupIntent. Required if a publishable key is used to retrieve the SetupIntent.

Returns

Returns a SetupIntent if a valid identifier was provided.

Update a SetupIntent

Updates a SetupIntent object.

Parameters

customer optional

ID of the Customer this SetupIntent belongs to, if one exists.

If present, the SetupIntent's payment method will be attached to the Customer on successful setup. Payment methods attached to other Customers cannot be used with this SetupIntent.

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

payment_method optional

ID of the payment method (a PaymentMethod, Card, or saved Source object) to attach to this SetupIntent.

payment_method_types optional

The list of payment method types (e.g. card) that this SetupIntent is allowed to set up. If this is not provided, defaults to ["card"].

More parameters

✓ **payment_method_options** optional dictionary

Payment-method-specific configuration for this SetupIntent.

Returns

Returns a SetupIntent object.

Confirm a SetupIntent

Confirm that your customer intends to set up the current or provided payment method. For example, you would confirm a SetupIntent when a customer hits the "Save" button on a payment method management page on your website.

If the selected payment method does not require any additional steps from the customer, the SetupIntent will transition to the `succeeded` status.

Otherwise, it will transition to the `requires_action` status and suggest additional actions via `next_action`. If setup fails, the SetupIntent will transition to the `requires_payment_method` status.

Parameters

payment_method optional

ID of the payment method (a PaymentMethod, Card, or saved Source object) to attach to this SetupIntent.

More parameters

✓ **mandate_data** optional dictionary

This hash contains details about the Mandate to create

✓ **payment_method_options** optional dictionary SECRET KEY ONLY

Payment-method-specific configuration for this SetupIntent.

✓ **return_url** optional

The URL to redirect your customer back to after they authenticate on the payment method's app or site. If you'd prefer to redirect to a mobile application, you can alternatively supply an application URI scheme. This parameter is only used for cards and other redirect-based payment methods.

Returns

Returns the resulting SetupIntent after all possible transitions are applied.

Cancel a SetupIntent

A SetupIntent object can be canceled when it is in one of these statuses:

`requires_payment_method`, `requires_confirmation`, or `requires_action`.

Once canceled, setup is abandoned and any operations on the SetupIntent will fail with an error.

Parameters

cancellation_reason optional

Reason for canceling this SetupIntent. Possible values are `abandoned`, `requested_by_customer`, or `duplicate`

Returns

Returns a SetupIntent object if the cancellation succeeded. Returns an error if the SetupIntent has already been canceled or is not in a cancelable state.

List all SetupIntents

Returns a list of SetupIntents.

Parameters

customer optional

Only return SetupIntents for the customer specified by this customer ID.

payment_method optional

Only return SetupIntents associated with the specified payment method.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`,

your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` `SetupIntents`, starting after `SetupIntent` `starting_after`. Each entry in the array is a separate `SetupIntent` object. If no more `SetupIntents` are available, the resulting array will be empty. This request should never return an error.

Verify microdeposits on a `SetupIntent`

Verifies microdeposits on a `SetupIntent` object.

Parameters

client_secret REQUIRED IF USING PUBLISHABLE KEY

The client secret of the `SetupIntent`.

amounts optional

Two positive integers, in `cents`, equal to the values of the microdeposits sent to the bank account.

descriptor_code optional

A six-character code starting with SM present in the microdeposit sent to the bank account.

Returns

Returns a SetupIntent object.

SetupAttempts

A SetupAttempt describes one attempted confirmation of a SetupIntent, whether that confirmation was successful or unsuccessful. You can use SetupAttempts to inspect details of a specific attempt at setting up a payment method using a SetupIntent.

The SetupAttempt object

Attributes

id string

Unique identifier for the object.

object string, value is "setup_attempt"

String representing the object's type. Objects of the same type share the same value.

application string EXPANDABLE "APPLICATION"

The value of **application** on the SetupIntent at the time of this confirmation.

created timestamp RETRIEVABLE WITH PUBLISHABLE KEY

Time at which the object was created. Measured in seconds since the Unix epoch.

customer string EXPANDABLE

The value of **customer** on the SetupIntent at the time of this confirmation.

livemode boolean RETRIEVABLE WITH PUBLISHABLE KEY

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

on_behalf_of string EXPANDABLE

The value of **on_behalf_of** on the SetupIntent at the time of this confirmation.

payment_method string EXPANDABLE RETRIEVABLE WITH PUBLISHABLE KEY

ID of the payment method used with this SetupAttempt.

payment_method_details hash

Details about the payment method at the time of SetupIntent confirmation.

setup_error hash

The error encountered during this attempt to confirm the SetupIntent, if any.

setup_intent string EXPANDABLE

ID of the SetupIntent that this attempt belongs to.

status string

Status of this SetupAttempt, one of `requires_confirmation`, `requires_action`, `processing`, `succeeded`, `failed`, or `abandoned`.

usage string

The value of `usage` on the SetupIntent at the time of this confirmation, one of `off_session` or `on_session`.

List all SetupAttempts

Returns a list of SetupAttempts associated with a provided SetupIntent.

Parameters

setup_intent REQUIRED

Only return SetupAttempts created by the SetupIntent specified by this ID.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` `SetupAttempts` which were created by the specified `SetupIntent`, starting after `SetupAttempts starting_after`. Each entry in the array is a separate `SetupAttempts` object. If no more `SetupAttempts` are available, the resulting array will be empty. This request should never return an error.

Payouts

A `Payout` object is created when you receive funds from Stripe, or when you initiate a payout to either a bank account or debit card of a [connected Stripe account](#). You can retrieve individual payouts, as well as list all payouts. Payouts are made on [varying schedules](#), depending on your country and industry.

Related guide: [Receiving Payouts](#).

The payout object

Attributes

id string

Unique identifier for the object.

amount integer

Amount (in cents) to be transferred to your bank account or debit card.

arrival_date timestamp

Date the payout is expected to arrive in the bank. This factors in delays like weekends or bank holidays.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

description string

An arbitrary string attached to the object. Often useful for displaying to users.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

statement_descriptor string

Extra information about a payout to be displayed on the user's bank statement.

status string

Current status of the payout: `paid`, `pending`, `in_transit`, `canceled` or `failed`. A payout is `pending` until it is submitted to the bank, when it becomes `in_transit`. The status then changes to `paid` if the transaction goes through, or to `failed` or `canceled` (within 5 business days). Some failed payouts may initially show as `paid` but then change to `failed`.

More attributes

object string, value is "payout"

String representing the object's type. Objects of the same type share the same value.

automatic boolean

Returns `true` if the payout was created by an [automated payout schedule](#), and `false` if it was [requested manually](#).

balance_transaction string [EXPANDABLE](#)

ID of the balance transaction that describes the impact of this payout on your account balance.

created timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

destination string EXPANDABLE CARD OR BANK ACCOUNT

ID of the bank account or card the payout was sent to.

failure_balance_transaction string EXPANDABLE

If the payout failed or was canceled, this will be the ID of the balance transaction that reversed the initial balance transaction, and puts the funds from the failed payout back in your balance.

failure_code string

Error code explaining reason for payout failure if available. See [Types of payout failures](#) for a list of failure codes.

failure_message string

Message to user further explaining reason for payout failure if available.

livemode boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

method string

The method used to send this payout, which can be `standard` or `instant`. `instant` is only supported for payouts to debit cards. (See [Instant payouts for marketplaces](#) for more information.)

original_payout string EXPANDABLE

If the payout reverses another, this is the ID of the original payout.

reversed_by string EXPANDABLE

If the payout was reversed, this is the ID of the payout that reverses this payout.

source_type string

The source balance this payout came from. One of `card`, `fpx`, or `bank_account`.

type enum

Can be `bank_account` or `card`.

Possible enum values

`card`

`bank_account`

Create a payout

To send funds to your own bank account, you create a new payout object. Your [Stripe balance](#) must be able to cover the payout amount, or you'll receive an "Insufficient Funds" error.

If your API key is in test mode, money won't actually be sent, though everything else will occur as if in live mode.

If you are creating a manual payout on a Stripe account that uses multiple payment source types, you'll need to specify the source type balance that the payout should draw from. The [balance object](#) details available and pending amounts by source type.

Parameters

amount REQUIRED

A positive integer in cents representing how much to payout.

currency REQUIRED

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

statement_descriptor optional

A string to be displayed on the recipient's bank or card statement. This may be at most 22 characters. Attempting to use a [statement_descriptor](#) longer than 22 characters will return an error. Note: Most banks will truncate this information and/or display it inconsistently. Some may not display it at all.

More parameters

[Collapse all](#)

destination optional

The ID of a bank account or a card to send the payout to. If no destination is supplied, the default external account for the specified currency will be used.

✓ **method** optional

The method used to send this payout, which can be `standard` or `instant`. `instant` is only supported for payouts to debit cards. (See [Instant payouts for marketplaces](#) for more information.)

✓ **source_type** optional

The balance type of your Stripe balance to draw this payout from. Balances for different payment sources are kept separately. You can find the amounts with the balances API. One of `bank_account`, `card`, or `fpx`.

Returns

Returns a payout object if there were no initial errors with the payout creation (invalid routing number, insufficient funds, etc). The status of the payout object will be initially marked as `pending`.

Retrieve a payout

Retrieves the details of an existing payout. Supply the unique payout ID from either a payout creation request or the payout list, and Stripe will return the corresponding payout information.

Parameters

No parameters.

Returns

Returns a payout object if a valid identifier was provided, and returns [an error](#) otherwise.

Update a payout

Updates the specified payout by setting the values of the parameters passed. Any parameters not provided will be left unchanged. This request accepts only the metadata as arguments.

Parameters

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the payout object if the update succeeded. This call returns [an error](#) if update parameters are invalid.

List all payouts

Returns a list of existing payouts sent to third-party bank accounts or that Stripe has sent you. The payouts are returned in sorted order, with the most recently created payouts appearing first.

Parameters

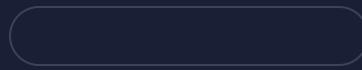
status optional

Only return payouts that have the given status: `pending`, `paid`, `failed`, or `canceled`.

More parameters

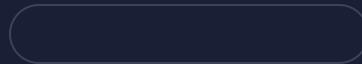
✓ **arrival_date** optional dictionary

A filter on the list based on the object `arrival_date` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **destination** optional

The ID of an external account - only return payouts sent to this external account.

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` payouts, starting after payout `starting_after`. Each entry in the array is a separate payout object. If no more payouts are available, the resulting array will be empty.

Cancel a payout

A previously created payout can be canceled if it has not yet been paid out. Funds will be refunded to your available balance. You may not cancel automatic Stripe payouts.

Parameters

No parameters.

Returns

Returns the payout object if the cancellation succeeded. Returns an error if the payout has already been canceled or cannot be canceled.

Reverse a payout

Reverses a payout by debiting the destination bank account. Only payouts for connected accounts to US bank accounts may be reversed at this time. If the payout is in the `[pending]` status, `/v1/payouts/:id/cancel` should be used instead.

By requesting a reversal via `/v1/payouts/:id/reverse`, you confirm that the authorized signatory of the selected bank account has authorized the debit on the bank account and that no other authorization is required.

Parameters

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the reversing payout object if the reversal was successful. Returns an error if the payout has already been reversed or cannot be reversed.

Types of payout failures

Payouts can fail for a variety of reasons. The reason a given payout failed is available in a Payout object's `failure_code` attribute.

Below is a list of all the types of failure codes we currently send. We may add more at any time, so in developing and maintaining your code, you should not assume that only these types exist. For further details on troubleshooting payouts, see [Receiving Payouts](#).

Failure codes

account_closed

The bank account has been closed.

account_frozen

The bank account has been frozen.

bank_account_restricted

The bank account has restrictions on either the type, or the number, of payouts allowed. This normally indicates that the bank account is a savings or other non-checking account.

bank_ownership_changed

The destination bank account is no longer valid because its branch has changed ownership.

could_not_process

The bank could not process this payout.

debit_notAuthorized

Debit transactions are not approved on the bank account. (Stripe requires bank accounts to be set up for both credit and debit payouts.)

declined

The bank has declined this transfer. Please contact the bank before retrying.

insufficient_funds

Your Stripe account has insufficient funds to cover the payout.

invalid_account_number

The routing number seems correct, but the account number is invalid.

incorrect_account_holder_name

Your bank notified us that the bank account holder name on file is incorrect.

incorrect_account_holder_address

Your bank notified us that the bank account holder address on file is incorrect.

incorrect_account_holder_tax_id

Your bank notified us that the bank account holder tax ID on file is incorrect.

invalid_currency

The bank was unable to process this payout because of its currency. This is probably because the bank account cannot accept payments in that currency.

no_account

The bank account details on file are probably incorrect. No bank account could be located with those details.

unsupported_card

The bank no longer supports payouts to this card.

Refunds

`Refund` objects allow you to refund a charge that has previously been created but not yet refunded. Funds will be refunded to the credit or debit card that was originally charged.

Related guide: [Refunds](#).

The refund object

Attributes

id string

Unique identifier for the object.

amount integer

Amount, in cents.

charge string EXPANDABLE

ID of the charge that was refunded.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

description string

An arbitrary string attached to the object. Often useful for displaying to users. (Available on non-card refunds only)

metadata hash

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

payment_intent string EXPANDABLE

ID of the PaymentIntent that was refunded.

reason enum

Reason for the refund, either user-provided (`duplicate`, `fraudulent`, or `requested_by_customer`) or generated by Stripe internally (`expired_uncaptured_charge`).

Possible enum values

`duplicate`

`fraudulent`

`requested_by_customer`

`expired_uncaptured_charge`

status string

Status of the refund. For credit card refunds, this can be `pending`, `succeeded`, or `failed`. For other types of refunds, it can be `pending`, `succeeded`, `failed`, or `canceled`. Refer to our [refunds](#) documentation for more details.

More attributes

✓ **object** string, value is "refund"

String representing the object's type. Objects of the same type share the same value.

✓ **balance_transaction** string EXPANDABLE

Balance transaction that describes the impact on your account balance.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **failure_balance_transaction** string EXPANDABLE

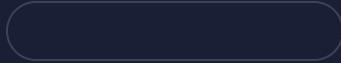
If the refund failed, this balance transaction describes the adjustment made on your account balance that reverses the initial balance transaction.

✓ **failure_reason** string

If the refund failed, the reason for refund failure if known. Possible values are `lost_or_stolen_card`, `expired_or_canceled_card`, or `unknown`.

✓ **next_action** hash

If the refund has a status of `requires_action`, this property will describe what the refund needs in order to continue processing.



✓ **receipt_number** string

This is the transaction number that appears on email receipts sent for this refund.

✓ **source_transfer_reversal** string EXPANDABLE CONNECT ONLY

The transfer reversal that is associated with the refund. Only present if the charge came from another Stripe account. See the Connect documentation for details.

✓ **transfer_reversal** string EXPANDABLE CONNECT ONLY

If the accompanying transfer was reversed, the transfer reversal object. Only applicable if the charge was created using the `destination` parameter.

Create a refund

When you create a new refund, you must specify a Charge or a PaymentIntent object on which to create it.

Creating a new refund will refund a charge that has previously been created but not yet refunded. Funds will be refunded to the credit or debit card that was originally charged.

You can optionally refund only part of a charge. You can do so multiple times, until the entire charge has been refunded.

Once entirely refunded, a charge can't be refunded again. This method will return an error when called on an already-refunded charge, or when trying to refund more money than is left on a charge.

Parameters

charge optional

The identifier of the charge to refund.

amount optional, default is entire charge

A positive integer in `cents` representing how much of this charge to refund. Can refund only up to the remaining, unrefunded amount of the charge.

metadata optional dictionary, default is {}

A set of key-value pairs that you can attach to a `Refund` object. This can be useful for storing additional information about the refund in a structured format. You can unset individual keys if you POST an empty value for that key. You can clear all keys if you POST an empty value for `metadata`

payment_intent optional

ID of the PaymentIntent to refund.

reason optional, default is null

String indicating the reason for the refund. If set, possible values are `duplicate`, `fraudulent`, and `requested_by_customer`. If you believe the charge to be fraudulent, specifying `fraudulent` as the reason will add the associated card and email to your `block lists`, and will also help us improve our fraud detection algorithms.

refund_application_fee optional, default is false CONNECT ONLY

Boolean indicating whether the application fee should be refunded when refunding this charge. If a full charge refund is given, the full application fee will be refunded. Otherwise, the application fee will be refunded in an amount proportional to the amount of the charge refunded.

An application fee can be refunded only by the application that created the charge.

reverse_transfer optional, default is false CONNECT ONLY

Boolean indicating whether the transfer should be reversed when refunding this charge. The transfer will be reversed proportionally to the amount being refunded (either the entire or partial amount).

A transfer can be reversed only by the application that created the charge.

Returns

Returns the `Refund` object if the refund succeeded. Returns an error if the Charge/PaymentIntent has already been refunded, or if an invalid identifier was provided.

Retrieve a refund

Retrieves the details of an existing refund.

Parameters

No parameters.

Returns

Returns a refund if a valid ID was provided. Returns [an error](#) otherwise.

Update a refund

Updates the specified refund by setting the values of the parameters passed. Any parameters not provided will be left unchanged.

This request only accepts `metadata` as an argument.

Parameters

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the refund object if the update succeeded. This call will return [an error](#) if update parameters are invalid.

Cancel a refund

Cancels a refund with a status of `requires_action`.

Refunds in other states cannot be canceled, and only refunds for payment methods that require customer action will enter the `requires_action` state.

Parameters

No parameters.

Returns

Returns the refund object if the cancelation succeeded. This call will return [an error](#) if the refund is unable to be canceled.

List all refunds

Returns a list of all refunds you've previously created. The refunds are returned in sorted order, with the most recent refunds appearing first. For convenience, the 10 most recent refunds are always available by default on the charge object.

Parameters

charge optional

Only return refunds for the charge specified by this charge ID.

payment_intent optional

Only return refunds for the PaymentIntent specified by this ID.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

starting_after optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` refunds, starting after refund `starting_after`. Each entry in the array is a separate refund object. If no more refunds are available, the resulting array will be empty. If you provide a non-existent charge ID, this call returns [an error](#).

Tokens

Tokenization is the process Stripe uses to collect sensitive card or bank account details, or personally identifiable information (PII), directly from your customers in a secure manner. A token representing this information is returned to your server to use. You should use our [recommended payments integrations](#) to perform this process client-side. This ensures that no sensitive card data touches your server, and allows your integration to operate in a PCI-compliant way.

If you cannot use client-side tokenization, you can also create tokens using the API with either your publishable or secret API key. Keep in mind that if your integration uses this method, you are responsible for any PCI compliance that may be required, and you must keep your secret API key safe. Unlike with client-side tokenization, your customer's information is not sent directly to Stripe, so we cannot determine how it is handled or stored.

Tokens cannot be stored or used more than once. To store card or bank account information for later use, you can create [Customer](#) objects or [Custom accounts](#). Note that [Radar](#), our integrated solution for automatic fraud protection, performs best with integrations that use client-side tokenization.

Related guide: [Accept a payment](#)

The token object

Attributes

id string

Unique identifier for the object.

More attributes

✓ **object** string, value is "token"

String representing the object's type. Objects of the same type share the same value.

✓ **bank_account** hash

Hash describing the bank account.

✓ **card** hash

Hash describing the card used to make the charge.

✓ **client_ip** string

IP address of the client that generated the token.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **type** string

Type of the token: `account`, `bank_account`, `card`, or `pii`.

✓ **used** boolean

Whether this token has already been used (tokens can be used only once).

Create a card token

Creates a single-use token that represents a credit card's details. This token can be used in place of a credit card dictionary with any API method. These tokens can be used only once: by [creating a new Charge object](#), or by attaching them to a [Customer object](#).

In most cases, you should use our recommended [payments integrations](#) instead of using the API.

Parameters

card optional dictionary

The card this token will represent. If you also pass in a customer, the card must be the ID of a card belonging to the customer. Otherwise, if you do not pass in a customer, this is a dictionary containing a user's credit card details, with the options described below.

More parameters

✓ **customer** optional CONNECT ONLY

The customer (owned by the application's account) for which to create a token. Also, this can be used only with an [OAuth access token](#) or [Stripe-Account header](#). For more details, see [Cloning Saved Payment Methods](#).

Returns

Returns the created card token if successful. Otherwise, this call raises [an error](#).

Create a bank account token

Creates a single-use token that represents a bank account's details. This token can be used with any API method in place of a bank account dictionary. This token can be used only once, by attaching it to a [Custom account](#).

Parameters

bank_account optional

The bank account this token will represent.

✓ customer optional CONNECT ONLY

The customer (owned by the application's account) for which to create a token. This can be used only with an [OAuth access token](#) or [Stripe-Account header](#). For more details, see [Cloning Saved Payment Methods](#).

Returns

Returns the created bank account token if successful. Otherwise, this call returns [an error](#).

Create a PII token

Creates a single-use token that represents the details of personally identifiable information (PII). This token can be used in place of an [id_number](#) in Account or Person Update API methods. A PII token can be used only once.

Parameters**pii** REQUIRED

The PII this token will represent.

Returns

Returns the created PII token if successful. Otherwise, this call returns [an error](#).

Create an account token

Creates a single-use token that wraps a user's legal entity information. Use this when creating or updating a Connect account. See [the account tokens documentation](#) to learn more.

In live mode, account tokens can only be created with your application's publishable key. In test mode, account tokens can be created with your secret key or publishable key.

Parameters

account REQUIRED

Information for the account this token will represent.

Returns

Returns the created account token if successful. Otherwise, this call returns [an error](#).

Create a person token

Creates a single-use token that represents the details for a person. Use this when creating or updating persons associated with a Connect account. See [the documentation](#) to learn more.

Person tokens may be created only in live mode, with your application's publishable key. Your application's secret key may be used to create person tokens only in test mode.

Parameters

person REQUIRED

Information for the person this token will represent.

Returns

Returns the created person token if successful. Otherwise, this call returns [an error](#).

Create a CVC update token

Creates a single-use token that represents an updated CVC value to be used for [CVC re-collection](#). This token can be used when [confirming a card payment](#) using a saved card on a

`PaymentIntent` with `confirmation_method: manual`.

For most cases, use our [JavaScript library](#) instead of using the API. For a `PaymentIntent` with `confirmation_method: automatic`, use our recommended [payments integration](#) without tokenizing the CVC value.

Parameters

cvc_update REQUIRED

The updated CVC value this token will represent.

Returns

Returns the created CVC update token if successful. Otherwise, this call raises [an error](#).

Retrieve a token

Retrieves the token with the given ID.

Parameters

No parameters.

Returns

Returns a token if a valid ID was provided. Returns [an error](#) otherwise.

PaymentMethods

PaymentMethod objects represent your customer's payment instruments. You can use them with [PaymentIntents](#) to collect payments or save them to Customer objects to store instrument details for future payments.

The PaymentMethod object

Attributes

id string

Unique identifier for the object.

billing_details hash

Billing information associated with the PaymentMethod that may be used or required by particular types of payment methods.

customer string EXPANDABLE

The ID of the Customer to which this PaymentMethod is saved. This will not be set when the PaymentMethod has not been saved to a Customer.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

type enum

The type of the PaymentMethod. An additional hash is included on the PaymentMethod with a name matching this value. It contains additional information specific to the PaymentMethod type.

Possible enum values

`acss_debit`

[Pre-authorized debit payments](#) are used to debit Canadian bank accounts through the Automated Clearing Settlement System (ACSS).

`afterpay_clearpay`

[Afterpay / Clearpay](#) is a buy now, pay later payment method used in Australia, Canada, France, New Zealand, Spain, the UK, and the US.

`alipay`

[Alipay](#) is a digital wallet payment method used in China.

`au_becs_debit`

[BECS Direct Debit](#) is used to debit Australian bank accounts through the Bulk Electronic Clearing System (BECS).

bacs_debit

Bacs Direct Debit is used to debit UK bank accounts.

bancontact

Bancontact is a bank redirect payment method used in Belgium.

boleto

Boleto is a voucher-based payment method used in Brazil.

Show 18 more

More attributes

✓ **object** string, value is "payment_method"

String representing the object's type. Objects of the same type share the same value.

✓ **acss_debit** hash

If this is an **acss_debit** PaymentMethod, this hash contains details about the ACSS Debit payment method.

✓ **afterpay_clearpay** hash

If this is an **AfterpayClearpay** PaymentMethod, this hash contains details about the AfterpayClearpay payment method.

✓ **alipay** hash

If this is an **Alipay** PaymentMethod, this hash contains details about the Alipay payment method.

✓ **au_becs_debit** hash

If this is an **au_becs_debit** PaymentMethod, this hash contains details about the bank account.

✓ **bacs_debit** hash

If this is a **bacs_debit** PaymentMethod, this hash contains details about the Bacs Direct Debit bank account.

✓ **bancontact** hash

If this is a `bancontact` PaymentMethod, this hash contains details about the Bancontact payment method.

✓ **boleto** hash

If this is a `boleto` PaymentMethod, this hash contains details about the Boleto payment method.

✓ **card** hash

If this is a `card` PaymentMethod, this hash contains the user's card details.

✓ **card_present** hash

If this is a `card_present` PaymentMethod, this hash contains details about the Card Present payment method.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **customer_balance** hash PREVIEW FEATURE

If this is a `customer_balance` PaymentMethod, this hash contains details about the CustomerBalance payment method.

✓ **eps** hash

If this is an `eps` PaymentMethod, this hash contains details about the EPS payment method.

✓ **fpx** hash

If this is an `fpx` PaymentMethod, this hash contains details about the FPX payment method.

✓ **giropay** hash

If this is a `giropay` PaymentMethod, this hash contains details about the Giropay payment method.

✓ **grabpay** hash

If this is a `grabpay` PaymentMethod, this hash contains details about the GrabPay payment method.

✓ **ideal** hash

If this is an `ideal` PaymentMethod, this hash contains details about the iDEAL payment method.

✓ **interac_present** hash PREVIEW FEATURE

If this is an `interac_present` PaymentMethod, this hash contains details about the Interac Present payment method.

✓ **klarna** hash

If this is a `klarna` PaymentMethod, this hash contains details about the Klarna payment method.

✓ **konbini** hash

If this is a `konbini` PaymentMethod, this hash contains details about the Konbini payment method.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **oxxo** hash

If this is an `oxxo` PaymentMethod, this hash contains details about the OXXO payment method.

✓ **p24** hash

If this is a `p24` PaymentMethod, this hash contains details about the P24 payment method.

✓ **paynow** hash

If this is a `paynow` PaymentMethod, this hash contains details about the PayNow payment method.

✓ **sepa_debit** hash

If this is a `sepa_debit` PaymentMethod, this hash contains details about the SEPA debit bank account.

✓ **sofort** hash

If this is a `sofort` PaymentMethod, this hash contains details about the SOFORT payment method.

✓ **us_bank_account** hash

If this is an `us_bank_account` PaymentMethod, this hash contains details about the US bank account payment method.

✓ **wechat_pay** hash

If this is an `wechat_pay` PaymentMethod, this hash contains details about the wechat_pay payment method.

Create a PaymentMethod

Creates a PaymentMethod object. Read the [Stripe.js reference](#) to learn how to create PaymentMethods via Stripe.js.

Instead of creating a PaymentMethod directly, we recommend using the [PaymentIntents API](#) to accept a payment immediately or the [SetupIntent API](#) to collect payment method details ahead of a future payment.

Parameters

type REQUIRED

The type of the PaymentMethod. An additional hash is included on the PaymentMethod with a name matching this value. It contains additional information specific to the PaymentMethod type. Required unless `payment_method` is specified (see the [Cloning PaymentMethods guide](#)).

Possible enum values

`acss_debit`

[Pre-authorized debit payments](#) are used to debit Canadian bank accounts through the Automated Clearing Settlement System (ACSS).

`afterpay_clearpay`

[Afterpay / Clearpay](#) is a buy now, pay later payment method used in Australia, Canada, France, New Zealand, Spain, the UK, and the US.

`alipay`

[Alipay](#) is a digital wallet payment method used in China.

`au_becs_debit`

[BECS Direct Debit](#) is used to debit Australian bank accounts through the Bulk Electronic Clearing System (BECS).

`bacs_debit`

[Bacs Direct Debit](#) is used to debit UK bank accounts.

`bancontact`

[Bancontact](#) is a bank redirect payment method used in Belgium.

`boleto`

[Boleto](#) is a voucher-based payment method used in Brazil.

`billing_details` optional dictionary

Billing information associated with the PaymentMethod that may be used or required by particular types of payment methods.

`metadata` optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset

by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ `acss_debit` optional dictionary

If this is an `acss_debit` PaymentMethod, this hash contains details about the ACSS Debit payment method.

✓ `afterpay_clearpay` optional dictionary

If this is an `AfterpayClearpay` PaymentMethod, this hash contains details about the AfterpayClearpay payment method.

✓ `alipay` optional dictionary

If this is an `Alipay` PaymentMethod, this hash contains details about the Alipay payment method.

✓ `au_becs_debit` optional dictionary

If this is an `au_becs_debit` PaymentMethod, this hash contains details about the bank account.

✓ `bacs_debit` optional dictionary

If this is a `bacs_debit` PaymentMethod, this hash contains details about the Bacs Direct Debit bank account.

✓ `bancontact` optional dictionary

If this is a `bancontact` PaymentMethod, this hash contains details about the Bancontact payment method.

✓ `boleto` optional dictionary

If this is a `boleto` PaymentMethod, this hash contains details about the Boleto payment method.

✓ **card** optional dictionary

If this is a `card` PaymentMethod, this hash contains the user's card details. For backwards compatibility, you can alternatively provide a Stripe token (e.g., for Apple Pay, Amex Express Checkout, or legacy Checkout) into the card hash with format `card: {token: "tok_visa"}`. When providing a card number, you must meet the requirements for [PCI compliance](#). We strongly recommend using Stripe.js instead of interacting with this API directly.

✓ **customer_balance** optional dictionary PREVIEW FEATURE

If this is a `customer_balance` PaymentMethod, this hash contains details about the CustomerBalance payment method.

✓ **eps** optional dictionary

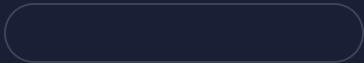
If this is an `eps` PaymentMethod, this hash contains details about the EPS payment method.

✓ **fpx** optional dictionary

If this is an `fpx` PaymentMethod, this hash contains details about the FPX payment method.

✓ **giropay** optional dictionary

If this is a `giropay` PaymentMethod, this hash contains details about the Giropay payment method.



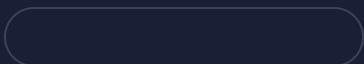
✓ **grabpay** optional dictionary

If this is a `grabpay` PaymentMethod, this hash contains details about the GrabPay payment method.



✓ **ideal** optional dictionary

If this is an `ideal` PaymentMethod, this hash contains details about the iDEAL payment method.



✓ **interac_present** optional dictionary PREVIEW FEATURE

If this is an `interac_present` PaymentMethod, this hash contains details about the Interac Present payment method.

✓ **klarna** optional dictionary

If this is a `klarna` PaymentMethod, this hash contains details about the Klarna payment method.

✓ **konbini** optional dictionary

If this is a `konbini` PaymentMethod, this hash contains details about the Konbini payment method.

✓ **oxxo** optional dictionary

If this is an `oxxo` PaymentMethod, this hash contains details about the OXXO payment method.

✓ **p24** optional dictionary

If this is a `p24` PaymentMethod, this hash contains details about the P24 payment method.

✓ **paynow** optional dictionary

If this is a `paynow` PaymentMethod, this hash contains details about the PayNow payment method.

✓ **sepa_debit** optional dictionary

If this is a `sepa_debit` PaymentMethod, this hash contains details about the SEPA debit bank account.

✓ **sofort** optional dictionary

If this is a `sofort` PaymentMethod, this hash contains details about the SOFORT payment method.

✓ **us_bank_account** optional dictionary

If this is an `us_bank_account` PaymentMethod, this hash contains details about the US bank account payment method.

∨ **wechat_pay** optional dictionary

If this is an `wechat_pay` PaymentMethod, this hash contains details about the wechat_pay payment method.

Returns

Returns a PaymentMethod object.

Retrieve a PaymentMethod

Retrieves a PaymentMethod object.

Parameters

No parameters.

Returns

Returns a PaymentMethod object.

Update a PaymentMethod

Updates a PaymentMethod object. A PaymentMethod must be attached a customer to be updated.

Parameters

billing_details optional dictionary

Billing information associated with the PaymentMethod that may be used or required by particular types of payment methods.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ **card** optional dictionary

If this is a `card` PaymentMethod, this hash contains the user's card details.

✓ **us_bank_account** optional dictionary

If this is an `us_bank_account` PaymentMethod, this hash contains details about the US bank account payment method.

Returns

Returns a PaymentMethod object.

List PaymentMethods

Returns a list of PaymentMethods. For listing a customer's payment methods, you should use [List a Customer's PaymentMethods](#)

Parameters

type REQUIRED

A required filter on the list, based on the object `type` field.

Possible enum values

`acss_debit`

`afterpay_clearpay`

`alipay`

`au_becs_debit`

`bacs_debit`

`bancontact`

`boleto`

customer optional

The ID of the customer whose PaymentMethods will be retrieved. If not provided, the response list will be empty.

More parameters

[Collapse all](#)

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` PaymentMethods of type `type`, starting after PaymentMethods `starting_after`. Each entry in the array is a

separate PaymentMethod object. If no more PaymentMethods are available, the resulting array will be empty. This request should never return an error.

List a Customer's PaymentMethods

Returns a list of PaymentMethods for a given Customer

Parameters

type REQUIRED

A required filter on the list, based on the object `type` field.

Possible enum values
<code>acss_debit</code>
<code>afterpay_clearpay</code>
<code>alipay</code>
<code>au_becs_debit</code>
<code>bacs_debit</code>
<code>bancontact</code>
<code>boleto</code>
Show 16 more

More parameters

[Collapse all](#)

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` `PaymentMethods` of type `type`, starting after `PaymentMethods starting_after`. Each entry in the array is a separate `PaymentMethod` object. If no more `PaymentMethods` are available, the resulting array will be empty. This request should never return an error.

Attach a PaymentMethod to a Customer

Attaches a `PaymentMethod` object to a `Customer`.

To attach a new `PaymentMethod` to a `customer` for future payments, we recommend you use a `SetupIntent` or a `PaymentIntent` with `setup_future_usage`. These approaches will perform any necessary steps to ensure that the `PaymentMethod` can be used in a future payment. Using the `/v1/payment_methods/:id/attach` endpoint does not ensure that future payments can be made with the attached `PaymentMethod`. See [Optimizing cards for future payments](#) for more information about setting up future payments.

To use this `PaymentMethod` as the default for invoice or subscription payments, set `invoice_settings.default_payment_method`, on the `Customer` to the `PaymentMethod`'s ID.

Parameters

`customer` REQUIRED

The ID of the `customer` to which to attach the `PaymentMethod`.

Returns

Returns a `PaymentMethod` object.

Detach a PaymentMethod from a Customer

Detaches a PaymentMethod object from a Customer. After a PaymentMethod is detached, it can no longer be used for a payment or re-attached to a Customer.

Parameters

No parameters.

Returns

Returns a PaymentMethod object.

Bank Accounts

These bank accounts are payment methods on `Customer` objects.

On the other hand [External Accounts](#) are transfer destinations on `Account` objects for [Custom accounts](#). They can be bank accounts or debit cards as well, and are documented in the links above.

Related guide: [Bank Debits and Transfers](#).

The bank account object

Attributes

`id` string

Unique identifier for the object.

`account_holder_name` string

The name of the person or business that owns the bank account.

`account_holder_type` string

The type of entity that holds the account. This can be either `individual` or `company`.

bank_name string

Name of the bank associated with the routing number (e.g., `WELLS FARGO`).

country string

Two-letter ISO code representing the country the bank account is located in.

currency currency

Three-letter [ISO code for the currency](#) paid out to the bank account.

customer string [EXPANDABLE](#)

The ID of the customer that the bank account is associated with.

fingerprint string

Uniquely identifies this particular bank account. You can use this attribute to check whether two bank accounts are the same.

last4 string

The last four digits of the bank account number.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

routing_number string

The routing transit number for the bank account.

More attributes[Collapse all](#)**object** string, value is "bank_account"

String representing the object's type. Objects of the same type share the same value.

account string [EXPANDABLE](#)

The ID of the account that the bank account is associated with.

account_type string

The bank account type. This can only be `checking` or `savings` in most countries. In Japan, this can only be `futsu` or `toza`.

available_payout_methods array

A set of available payout methods for this bank account. Only values from this set should be passed as the `method` when creating a payout.

status string

For bank accounts, possible values are `new`, `validated`, `verified`, `verification_failed`, or `errored`. A bank account that hasn't had any activity or validation performed is `new`. If Stripe can determine that the bank account exists, its status will be `validated`. Note that there often isn't enough information to know (e.g., for smaller credit unions), and the validation is not always run. If customer bank account verification has succeeded, the bank account status will be `verified`. If the verification failed for any reason, such as microdeposit failure, the status will be `verification_failed`. If a transfer sent to this bank account fails, we'll set the status to `errored` and will not continue to send transfers until the bank details are updated.

For external accounts, possible values are `new` and `errored`. Validations aren't run against external accounts because they're only used for payouts. This means the other statuses don't apply. If a transfer fails, the status is set to `errored` and transfers are stopped until account details are updated.

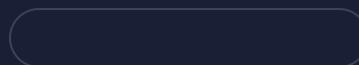
Create a bank account

When you create a new bank account, you must specify a `Customer` object on which to create it.

Parameters

source REQUIRED

Either a token, like the ones returned by [Stripe.js](#), or a dictionary containing a user's bank account details (with the options shown below).



metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the bank account object.

Retrieve a bank account

By default, you can see the 10 most recent sources stored on a Customer directly on the object, but you can also retrieve details about a specific bank account stored on the Stripe account.

Parameters

No parameters.

Returns

Returns the bank account object.

Update a bank account

Updates the `account_holder_name`, `account_holder_type`, and `metadata` of a bank account belonging to a customer. Other bank account details are not editable, by design.

Parameters

`account_holder_name` optional

The name of the person or business that owns the bank account.

`account_holder_type` optional

The type of entity that holds the account. This can be either `individual` or `company`.

`metadata` optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the bank account object.

Verify a bank account

A customer's bank account must first be [verified](#) before it can be charged. Stripe supports instant verification using [Plaid](#) for many of the most popular banks. If your customer's bank is not supported or you do not wish to integrate with Plaid, you must [manually verify](#) the customer's bank account using the API.

Parameters

amounts optional

Two positive integers, in *cents*, equal to the values of the microdeposits sent to the bank account.

Returns

Returns the bank account object with a `status` of [verified](#).

Delete a bank account

You can delete bank accounts from a Customer.

Parameters

No parameters.

Returns

Returns the deleted bank account object.

List all bank accounts

You can see a list of the bank accounts belonging to a Customer. Note that the 10 most recent sources are always available by default on the Customer. If you need more than those 10, you can use this API method and the `limit` and `starting_after` parameters to page through additional bank accounts.

Parameters

✓ `ending_before` optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ `limit` optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ `starting_after` optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

Returns a list of the bank accounts stored on the customer.

Cash balance

A customer's `Cash balance` represents real funds. Customers can add funds to their cash balance by sending a bank transfer. These funds can be used for payment and can eventually be paid out to your bank account.

The cash balance object

Attributes

object string, value is "cash_balance"

String representing the object's type. Objects of the same type share the same value.

available hash

A hash of all cash balances available to this customer. You cannot delete a customer with any cash balances, even if the balance is 0.

customer string

The ID of the customer whose cash balance this object represents.

livemode boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

settings hash

The customer's default cash balance settings.



Cards

You can store multiple cards on a customer in order to charge the customer later.

You can also store multiple debit cards on a recipient in order to transfer to those cards later.

Related guide: [Card Payments with Sources](#).

The card object

Attributes

id string

Unique identifier for the object.

address_city string

City/District/Suburb/Town/Village.

address_country string

Billing address country, if provided when creating card.

address_line1 string

Address line 1 (Street address/PO Box/Company name).

address_line2 string

Address line 2 (Apartment/Suite/Unit/Building).

address_state string

State/County/Province/Region.

address_zip string

ZIP or postal code.

address_zip_check string

If `address_zip` was provided, results of the check: `pass`, `fail`, `unavailable`, or `unchecked`.

brand string

Card brand. Can be `American Express`, `Diners Club`, `Discover`, `JCB`, `MasterCard`, `UnionPay`, `Visa`, or `Unknown`.

country string

Two-letter ISO code representing the country of the card. You could use this attribute to get a sense of the international breakdown of cards you've collected.

customer string EXPANDABLE

The customer that this card belongs to. This attribute will not be in the card object if the card belongs to an account or recipient instead.

cvc_check string

If a CVC was provided, results of the check: `pass`, `fail`, `unavailable`, or `unchecked`. A result of `unchecked` indicates that CVC was provided but hasn't been checked yet. Checks are typically performed when attaching a card to a Customer object, or when creating a charge. For more details, see [Check if a card is valid without a charge](#).

exp_month integer

Two-digit number representing the card's expiration month.

exp_year integer

Four-digit number representing the card's expiration year.

fingerprint string

Uniquely identifies this particular card number. You can use this attribute to check whether two customers who've signed up with you are using the same card number, for example. For payment methods that tokenize card information (Apple Pay, Google Pay), the tokenized number might be provided instead of the underlying card number.

Starting May 1, 2021, card fingerprint in India for Connect will change to allow two fingerprints for the same card — one for India and one for the rest of the world.

funding string

Card funding type. Can be `credit`, `debit`, `prepaid`, or `unknown`.

last4 string

The last four digits of the card.

metadata hash

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

name string

Cardholder name.

More attributes

✓ **object** string, value is "card"

String representing the object's type. Objects of the same type share the same value.

✓ **account** string EXPANDABLE CUSTOM CONNECT ONLY

The account this card belongs to. This attribute will not be in the card object if the card belongs to a customer or recipient instead.

✓ **address_line1_check** string

If `address_line1` was provided, results of the check: `pass`, `fail`, `unavailable`, or `unchecked`.

✓ **available_payout_methods** array

A set of available payout methods for this card. Only values from this set should be passed as the `method` when creating a payout.

✓ **currency** currency CUSTOM CONNECT ONLY

Three-letter **ISO code for currency**. Only applicable on accounts (not customers or recipients). The card can be used as a transfer destination for funds in this currency.

✓ **dynamic_last4** string

(For tokenized numbers only.) The last four digits of the device account number.

✓ **recipient** string EXPANDABLE

The recipient that this card belongs to. This attribute will not be in the card object if the card belongs to a customer or account instead.

✓ **tokenization_method** string

If the card number is tokenized, this is the method that was used. Can be `android_pay` (includes Google Pay), `apple_pay`, `masterpass`, `visa_checkout`, or null.

Create a card

When you create a new credit card, you must specify a customer or recipient on which to create it.

If the card's owner has no default card, then the new card will become the default. However, if the owner already has a default, then it will not change. To change the default, you should [update the customer](#) to have a new `default_source`.

Parameters

source REQUIRED

A token, like the ones returned by [Stripe.js](#). Stripe will automatically validate the card.



metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the `Card` object.

Retrieve a card

You can always see the 10 most recent cards directly on a customer; this method lets you retrieve details about a specific card stored on the customer.

Parameters

No parameters.

Returns

Returns the `Card` object.

Update a card

If you need to update only some card details, like the billing address or expiration date, you can do so without having to re-enter the full card details. Also, Stripe works directly with card networks so that your customers can [continue using your service](#) without interruption.

When you update a card, Stripe typically validates the card automatically. For more details, see [Check if a card is valid without a charge](#).

Parameters

`address_city` optional

City/District/Suburb/Town/Village.

`address_country` optional

Billing address country, if provided when creating card.

`address_line1` optional

Address line 1 (Street address/PO Box/Company name).

`address_line2` optional

Address line 2 (Apartment/Suite/Unit/Building).

`address_state` optional

State/County/Province/Region.

`address_zip` optional

ZIP or postal code.

exp_month optional

Two digit number representing the card's expiration month.

exp_year optional

Four digit number representing the card's expiration year.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

name optional

Cardholder name.

Returns

Returns the `Card` object.

Delete a card

You can delete cards from a customer.

If you delete a card that is currently the default source, then the most recently added source will become the new default. If you delete a card that is the last remaining source on the customer, then the `default_source` attribute will become null.

For recipients: if you delete the default card, then the most recently added card will become the new default. If you delete the last remaining card on a recipient, then the `default_card` attribute will become null.

Note that for cards belonging to customers, you might want to prevent customers on paid subscriptions from deleting all cards on file, so that there is at least one default card for the next invoice payment attempt.

Parameters

No parameters.

Returns

Returns the deleted `Card` object.

List all cards

You can see a list of the cards belonging to a customer. Note that the 10 most recent sources are always available on the `Customer` object. If you need more than those 10, you can use this API method and the `limit` and `starting_after` parameters to page through additional cards.

Parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

Returns a list of the cards stored on the customer.

Sources

`Source` objects allow you to accept a variety of payment methods. They represent a customer's payment instrument, and can be used with the Stripe API just like a `Card` object: once chargeable, they can be charged, or can be attached to customers.

Related guides: [Sources API](#) and [Sources & Customers](#).

The source object

Attributes

id string

Unique identifier for the object.

amount integer

A positive integer in the smallest currency unit (that is, 100 cents for \$1.00, or 1 for ¥1, Japanese Yen being a zero-decimal currency) representing the total amount associated with the source. This is the amount for which the source will be chargeable once ready. Required for `single_use` sources.

currency currency

Three-letter ISO code for the currency associated with the source. This is the currency for which the source will be chargeable once ready. Required for `single_use` sources.

customer string

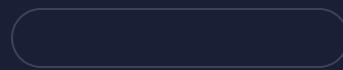
The ID of the customer to which this source is attached. This will not be present when the source has not been attached to a customer.

metadata hash

Set of key-value pairs that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

owner hash

Information about the owner of the payment instrument that may be used or required by particular source types.



redirect hash

Information related to the redirect flow. Present if the source is authenticated by a redirect (`flow` is `redirect`).

statement_descriptor string

Extra information about a source. This will appear on your customer's statement every time you charge the source.

status string

The status of the source, one of `canceled`, `chargeable`, `consumed`, `failed`, or `pending`. Only `chargeable` sources can be used to create a charge.

type string

The `type` of the source. The `type` is a payment method, one of `ach_credit_transfer`, `ach_debit`, `alipay`, `bancontact`, `card`, `card_present`, `eps`, `giropay`, `ideal`, `multibanco`, `klarna`, `p24`, `sepa_debit`, `sofort`, `three_d_secure`, or `wechat`. An additional hash is included on the source with a name matching this value. It contains additional information specific to the payment method used.

More attributes

✓ **object** string, value is "source"

String representing the object's type. Objects of the same type share the same value.

✓ **client_secret** string

The client secret of the source. Used for client-side retrieval using a publishable key.

✓ **code_verification** hash

Information related to the code verification flow. Present if the source is authenticated by a verification code (`flow` is `code_verification`).

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **flow** string

The authentication `flow` of the source. `flow` is one of `redirect`, `receiver`, `code_verification`, `none`.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **receiver** hash

Information related to the receiver flow. Present if the source is a receiver (`flow` is `receiver`).



source_order hash

Information about the items and shipping associated with the source. Required for transactional credit (for example Klarna) sources before you can charge it.



usage string

Either `reusable` or `single_use`. Whether this source should be reusable or not. Some source types may or may not be reusable by construction, while others may leave the option at creation. If an incompatible value is passed, an error will be returned.

Create a source

Creates a new source object.

Parameters

type REQUIRED

The `type` of the source to create. Required unless `customer` and `original_source` are specified (see the [Cloning card Sources](#) guide)

amount optional

Amount associated with the source. This is the amount for which the source will be chargeable once ready. Required for `single_use` sources. Not supported for `receiver` type sources, where charge amount may not be specified until funds land.

currency optional

Three-letter [ISO code for the currency](#) associated with the source. This is the currency for which the source will be chargeable once ready.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

owner optional dictionary

Information about the owner of the payment instrument that may be used or required by particular source types.

redirect optional dictionary

Parameters required for the redirect flow. Required if the source is authenticated by a redirect (`flow` is `redirect`).

statement_descriptor optional

An arbitrary string to be displayed on your customer's statement. As an example, if your website is `RunClub` and the item you're charging for is a race ticket, you may want to specify a `statement_descriptor` of `RunClub 5K race ticket`. While many payment types will display this information, some may not display it at all.

More parameters

✓ **flow** optional

The authentication `flow` of the source to create. `flow` is one of `redirect`, `receiver`, `code_verification`, `none`. It is generally inferred unless a type supports multiple flows.

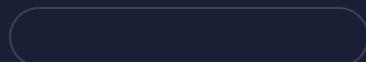
✓ **mandate** optional dictionary

Information about a mandate possibility attached to a source object (generally for bank debits) as well as its acceptance status.



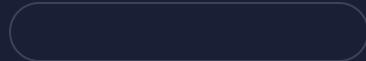
✓ **receiver** optional dictionary

Optional parameters for the receiver flow. Can be set only if the source is a receiver (`flow` is `receiver`).



✓ **source_order** optional dictionary

Information about the items and shipping associated with the source. Required for transactional credit (for example Klarna) sources before you can charge it.



✓ **token** optional

An optional token used to create the source. When passed, token properties will override source parameters.



usage optional

Either `reusable` or `single_use`. Whether this source should be reusable or not. Some source types may or may not be reusable by construction, while others may leave the option at creation. If an incompatible value is passed, an error will be returned.

Returns

Returns a newly created source.

Retrieve a source

Retrieves an existing source object. Supply the unique source ID from a source creation request and Stripe will return the corresponding up-to-date source object information.

Parameters

✓ `client_secret` optional

The client secret of the source. Required if a publishable key is used to retrieve the source.

Returns

Returns a source if a valid identifier was provided.

Update a source

Updates the specified source by setting the values of the parameters passed. Any parameters not provided will be left unchanged.

This request accepts the `metadata` and `owner` as arguments. It is also possible to update type specific information for selected payment methods. Please refer to our [payment method guides](#) for more detail.

Parameters

`amount` optional

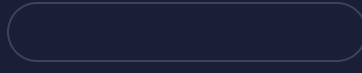
Amount associated with the source.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to **metadata**.

owner optional dictionary

Information about the owner of the payment instrument that may be used or required by particular source types.



More parameters

✓ **mandate** optional dictionary

Information about a mandate possibility attached to a source object (generally for bank debits) as well as its acceptance status.



✓ **source_order** optional dictionary

Information about the items and shipping associated with the source. Required for transactional credit (for example Klarna) sources before you can charge it.



Returns

Returns the source object if the update succeeded. This call will return **an error** if update parameters are invalid.

Attach a source

Attaches a Source object to a Customer. The source must be in a chargeable or pending state.

Parameters

source REQUIRED

The identifier of the source to be attached.

Returns

Returns the attached Source object.

Detach a source

Detaches a Source object from a Customer. The status of a source is changed to `consumed` when it is detached and it can no longer be used to create a charge.

Parameters

No parameters.

Returns

Returns the detached Source object.

Products

Products describe the specific goods or services you offer to your customers. For example, you might offer a Standard and Premium version of your goods or service; each version would be a separate Product. They can be used in conjunction with [Prices](#) to configure pricing in Payment Links, Checkout, and Subscriptions.

Related guides: [Set up a subscription, share a Payment Link, accept payments with Checkout](#), and more about [Products and Prices](#)

The product object

Attributes

id string

Unique identifier for the object.

active boolean

Whether the product is currently available for purchase.

description string

The product's description, meant to be displayable to the customer. Use this field to optionally store a long form explanation of the product being sold for your own rendering purposes.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

name string

The product's name, meant to be displayable to the customer.

More attributes

✓ **object** string, value is "product"

String representing the object's type. Objects of the same type share the same value.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **images** array containing strings

A list of up to 8 URLs of images for this product, meant to be displayable to the customer.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **package_dimensions** hash

The dimensions of this product for shipping purposes.

✓ **shippable** boolean

Whether this product is shipped (i.e., physical goods).

✓ **statement_descriptor** string

Extra information about a product which will appear on your customer's credit card statement. In the case that multiple products are billed at once, the first statement descriptor will be used.

✓ **tax_code** string EXPANDABLE

A tax code ID.

✓ **unit_label** string

A label that represents units of this product in Stripe and on customers' receipts and invoices. When set, this will be included in associated invoice line item descriptions.

✓ **updated** timestamp

Time at which the object was last updated. Measured in seconds since the Unix epoch.

✓ **url** string

A URL of a publicly-accessible webpage for this product.

Create a product

Creates a new product object.

Parameters

id optional

An identifier will be randomly generated by Stripe. You can optionally override this ID, but the ID must be unique across all products in your Stripe account.

name REQUIRED

The product's name, meant to be displayable to the customer.

active optional

Whether the product is currently available for purchase. Defaults to `true`.

description optional

The product's description, meant to be displayable to the customer. Use this field to optionally store a long form explanation of the product being sold for your own rendering purposes.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

More parameters

✓ **images** optional

A list of up to 8 URLs of images for this product, meant to be displayable to the customer.

✓ **package_dimensions** optional dictionary

The dimensions of this product for shipping purposes.

✓ **shippable** optional

Whether this product is shipped (i.e., physical goods).

✓ **statement_descriptor** optional

An arbitrary string to be displayed on your customer's credit card or bank statement. While most banks display this information consistently, some may display it incorrectly or not at all.

This may be up to 22 characters. The statement description may not include <, >, \, ", ' characters, and will appear on your customer's statement in capital letters. Non-ASCII characters are automatically stripped. It must contain at least one letter.

✓ **tax_code** optional

A [tax code](#) ID.

✓ **unit_label** optional

A label that represents units of this product in Stripe and on customers' receipts and invoices. When set, this will be included in associated invoice line item descriptions.

✓ **url** optional

A URL of a publicly-accessible webpage for this product.

Returns

Returns a product object if the call succeeded.

Retrieve a product

Retrieves the details of an existing product. Supply the unique product ID from either a product creation request or the product list, and Stripe will return the corresponding product information.

Parameters

No parameters.

Returns

Returns a product object if a valid identifier was provided.

Update a product

Updates the specific product by setting the values of the parameters passed. Any parameters not provided will be left unchanged.

Parameters

active optional

Whether the product is available for purchase.

description optional

The product's description, meant to be displayable to the customer. Use this field to optionally store a long form explanation of the product being sold for your own rendering purposes.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

name optional

The product's name, meant to be displayable to the customer.

✓ **images** optional

A list of up to 8 URLs of images for this product, meant to be displayable to the customer.

✓ **package_dimensions** optional dictionary

The dimensions of this product for shipping purposes.

✓ **shippable** optional

Whether this product is shipped (i.e., physical goods).

✓ **statement_descriptor** optional

An arbitrary string to be displayed on your customer's credit card or bank statement. While most banks display this information consistently, some may display it incorrectly or not at all.

This may be up to 22 characters. The statement description may not include <, >, \, ", ' characters, and will appear on your customer's statement in capital letters. Non-ASCII characters are automatically stripped. It must contain at least one letter. May only be set if `type=service`.

✓ **tax_code** optional

A [tax code](#) ID.

✓ **unit_label** optional

A label that represents units of this product in Stripe and on customers' receipts and invoices. When set, this will be included in associated invoice line item descriptions. May only be set if `type=service`.

✓ **url** optional

A URL of a publicly-accessible webpage for this product.

Returns

Returns the product object if the update succeeded.

List all products

Returns a list of your products. The products are returned sorted by creation date, with the most recently created products appearing first.

Parameters

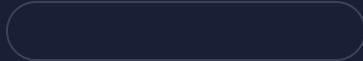
active optional

Only return products that are active or inactive (e.g., pass `false` to list all inactive products).

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **ids** optional

Only return products with the given IDs.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **shippable** optional

Only return products that can be shipped (i.e., physical, not digital products).

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

✓ **url** optional

Only return products with the given url.

Returns

A dictionary with a `data` property that contains an array of up to `limit` products, starting after product `starting_after`. Each entry in the array is a separate product object. If no more products are available, the resulting array will be empty. This request should never return an error.

Delete a product

Delete a product. Deleting a product is only possible if it has no prices associated with it. Additionally, deleting a product with `type=good` is only possible if it has no SKUs associated with it.

Parameters

No parameters.

Returns

Returns a deleted object on success. Otherwise, this call returns [an error](#).

Search products

Search for products you've previously created using Stripe's [Search Query Language](#). Don't use search in read-after-write flows where strict consistency is necessary. Under normal operating conditions, data is searchable in less than a minute. Occasionally, propagation of new or updated data can be up to an hour behind during outages. Search functionality is not available to merchants in India.

Parameters

query REQUIRED

The search query string. See [search query language](#) and the list of supported [query fields for products](#).

limit optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

page optional

A cursor for pagination across multiple pages of results. Don't include this parameter on the first call. Use the next_page value returned in a previous response to request subsequent results.

Returns

A dictionary with a `data` property that contains an array of up to `limit` products. If no objects match the query, the resulting array will be empty. See the related guide on [expanding properties in lists](#).

Prices

Prices define the unit cost, currency, and (optional) billing cycle for both recurring and one-time purchases of products. [Products](#) help you track inventory or provisioning, and prices help you track payment terms. Different physical goods or levels of service should be represented by products, and pricing options should be represented by prices. This approach lets you change prices without having to change your provisioning scheme.

For example, you might have a single "gold" product that has prices for \$10/month, \$100/year, and €9 once.

Related guides: [Set up a subscription](#), [create an invoice](#), and more about [products and prices](#).

The price object

Attributes

`id` string

Unique identifier for the object.

`active` boolean

Whether the price can be used for new purchases.

currency currency

Three-letter ISO currency code, in lowercase. Must be a supported currency.

metadata hash

Set of key-value pairs that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

nickname string

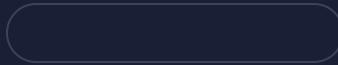
A brief description of the price, hidden from customers.

product string EXPANDABLE

The ID of the product this price is associated with.

recurring hash

The recurring components of a price such as `interval` and `usage_type`.

**type** string

One of `one_time` or `recurring` depending on whether the price is for a one-time purchase or a recurring (subscription) purchase.

unit_amount integer

The unit amount in cents to be charged, represented as a whole integer if possible. Only set if `billing_scheme=per_unit`.

More attributes

✓ **object** string, value is "price"

String representing the object's type. Objects of the same type share the same value.

✓ **billing_scheme** string

Describes how to compute the price per period. Either `per_unit` or `tiered`. `per_unit` indicates that the fixed amount (specified in `unit_amount` or `unit_amount_decimal`) will be charged per unit in `quantity` (for prices with `usage_type=licensed`), or per unit of total usage (for prices with `usage_type=metered`). `tiered` indicates that the unit pricing will be computed using a tiering strategy as defined using the `tiers` and `tiers_mode` attributes.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **lookup_key** string

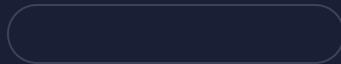
A lookup key used to retrieve prices dynamically from a static string. This may be up to 200 characters.

✓ **tax_behavior** string

Specifies whether the price is considered inclusive of taxes or exclusive of taxes. One of `inclusive`, `exclusive`, or `unspecified`. Once specified as either `inclusive` or `exclusive`, it cannot be changed.

✓ **tiers** array of hashes EXPANDABLE

Each element represents a pricing tier. This parameter requires `billing_scheme` to be set to `tiered`. See also the documentation for `billing_scheme`. This field is not included by default. To include it in the response, [expand](#) the `tiers` field.



✓ **tiers_mode** string

Defines if the tiering price should be `graduated` or `volume` based. In `volume`-based tiering, the maximum quantity within a period determines the per unit price. In `graduated` tiering, pricing can change as the quantity grows.

✓ **transform_quantity** hash

Apply a transformation to the reported usage or set quantity before computing the amount billed. Cannot be combined with `tiers`.

✓ **unit_amount_decimal** decimal string

The unit amount in cents to be charged, represented as a decimal string with at most 12 decimal places. Only set if `billing_scheme=per_unit`.

Create a price

Creates a new price for an existing product. The price can be recurring or one-time.

Parameters

currency REQUIRED

Three-letter ISO currency code, in lowercase. Must be a supported currency.

product REQUIRED UNLESS PRODUCT_DATA IS PROVIDED

The ID of the product that this price will belong to.

unit_amount REQUIRED UNLESS BILLING_SCHEME=TIERED

A positive integer in cents (or 0 for a free price) representing how much to charge.

active optional

Whether the price can be used for new purchases. Defaults to `true`.

metadata optional dictionary

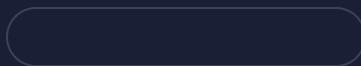
Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

nickname optional

A brief description of the price, hidden from customers.

recurring optional dictionary

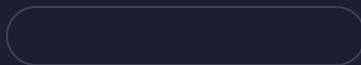
The recurring components of a price such as `interval` and `usage_type`.



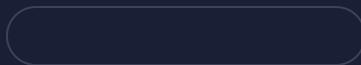
More parameters

product_data REQUIRED UNLESS PRODUCT IS PROVIDED

These fields can be used to create a new product that this price will belong to.

**tiers** REQUIRED IF BILLING_SCHEME=TIERED

Each element represents a pricing tier. This parameter requires `billing_scheme` to be set to `tiered`. See also the documentation for `billing_scheme`.

**tiers_mode** REQUIRED IF BILLING_SCHEME=TIERED

Defines if the tiering price should be `graduated` or `volume` based. In `volume`-based tiering, the maximum quantity within a period determines the per unit price, in `graduated` tiering pricing can successively change as the quantity grows.

Possible enum values

`volume`

✓ **billing_scheme** optional

Describes how to compute the price per period. Either `per_unit` or `tiered`. `per_unit` indicates that the fixed amount (specified in `unit_amount` or `unit_amount_decimal`) will be charged per unit in `quantity` (for prices with `usage_type=licensed`), or per unit of total usage (for prices with `usage_type=metered`). `tiered` indicates that the unit pricing will be computed using a tiering strategy as defined using the `tiers` and `tiers_mode` attributes.

✓ **lookup_key** optional

A lookup key used to retrieve prices dynamically from a static string. This may be up to 200 characters.

✓ **tax_behavior** optional

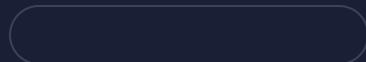
Specifies whether the price is considered inclusive of taxes or exclusive of taxes. One of `inclusive`, `exclusive`, or `unspecified`. Once specified as either `inclusive` or `exclusive`, it cannot be changed.

✓ **transfer_lookup_key** optional

If set to true, will atomically remove the lookup key from the existing price, and assign it to this price.

✓ **transform_quantity** optional dictionary

Apply a transformation to the reported usage or set quantity before computing the billed price. Cannot be combined with `tiers`.



✓ **unit_amount_decimal** optional

Same as `unit_amount`, but accepts a decimal value in cents with at most 12 decimal places. Only one of `unit_amount` and `unit_amount_decimal` can be set.

Returns

The newly created `Price` object is returned upon success. Otherwise, this call returns an error.

Retrieve a price

Retrieves the price with the given ID.

Parameters

No parameters.

Returns

Returns a price if a valid price or plan ID was provided. Returns [an error](#) otherwise.

Update a price

Updates the specified price by setting the values of the parameters passed. Any parameters not provided are left unchanged.

Parameters

active optional

Whether the price can be used for new purchases. Defaults to `true`.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

nickname optional

A brief description of the price, hidden from customers.

More parameters

[Collapse all](#)

✓ **lookup_key** optional

A lookup key used to retrieve prices dynamically from a static string. This may be up to 200 characters.

✓ **tax_behavior** optional

Specifies whether the price is considered inclusive of taxes or exclusive of taxes. One of `inclusive`, `exclusive`, or `unspecified`. Once specified as either `inclusive` or

`exclusive`, it cannot be changed.

✓ **transfer_lookup_key** optional

If set to true, will atomically remove the lookup key from the existing price, and assign it to this price.

Returns

The updated price object is returned upon success. Otherwise, this call returns [an error](#).

List all prices

Returns a list of your prices.

Parameters

active optional

Only return prices that are active or inactive (e.g., pass `false` to list all inactive prices).

currency optional

Only return prices for the given currency.

product optional

Only return prices for the given product.

type optional

Only return prices of type `recurring` or `one_time`.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **lookup_keys** optional

Only return the price with these lookup_keys, if any exist.

✓ **recurring** optional dictionary

Only return prices with these recurring fields.



✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` prices, starting after prices `starting_after`. Each entry in the array is a separate price object. If no more prices are available, the resulting array will be empty. This request should never return an error.

Search prices

Search for prices you've previously created using Stripe's [Search Query Language](#). Don't use search in read-after-write flows where strict consistency is necessary. Under normal operating conditions, data is searchable in less than a minute. Occasionally, propagation of new or updated data can be up to an hour behind during outages. Search functionality is not available to merchants in India.

Parameters

query REQUIRED

The search query string. See [search query language](#) and the list of supported [query fields for prices](#).

limit optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

page optional

A cursor for pagination across multiple pages of results. Don't include this parameter on the first call. Use the next_page value returned in a previous response to request subsequent results.

Returns

A dictionary with a `data` property that contains an array of up to `limit` prices. If no objects match the query, the resulting array will be empty. See the related guide on [expanding properties in lists](#).

Coupons

A coupon contains information about a percent-off or amount-off discount you might want to apply to a customer. Coupons may be applied to [invoices](#) or [orders](#). Coupons do not work with conventional one-off [charges](#).

The coupon object

Attributes

id string

Unique identifier for the object.

amount_off positive integer

Amount (in the `currency` specified) that will be taken off the subtotal of any invoices for this customer.

currency currency

If `amount_off` has been set, the three-letter ISO code for the currency of the amount to take off.

duration enum

One of `forever`, `once`, and `repeating`. Describes how long a customer who applies this coupon will get the discount.

Possible enum values

`once`

`repeating`

`forever`

duration_in_months positive integer

If `duration` is `repeating`, the number of months the coupon applies. Null if coupon `duration` is `forever` or `once`.

metadata hash

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

name string

Name of the coupon displayed to customers on for instance invoices or receipts.

percent_off decimal

Percent that will be taken off the subtotal of any invoices for this customer for the duration of the coupon. For example, a coupon with `percent_off` of 50 will make a \$100 invoice \$50 instead.

More attributes

✓ **object** string, value is "coupon"

String representing the object's type. Objects of the same type share the same value.

✓ **applies_to** hash EXPANDABLE

Contains information about what this coupon applies to. This field is not included by default.

To include it in the response, **expand** the `applies_to` field.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **max_redeemptions** positive integer

Maximum number of times this coupon can be redeemed, in total, across all customers, before it is no longer valid.

✓ **redeem_by** timestamp

Date after which the coupon can no longer be redeemed.

✓ **times_redeemed** positive integer or zero

Number of times this coupon has been applied to a customer.

✓ **valid** boolean

Taking account of the above properties, whether this coupon can still be applied to a customer.

Create a coupon

You can create coupons easily via the [coupon management](#) page of the Stripe dashboard. Coupon creation is also accessible via the API if you need to create coupons on the fly.

A coupon has either a `percent_off` or an `amount_off` and `currency`. If you set an `amount_off`, that amount will be subtracted from any invoice's subtotal. For example, an invoice with a subtotal of \$100 will have a final total of \$0 if a coupon with an `amount_off` of 20000 is applied to it and an invoice with a subtotal of \$300 will have a final total of \$100 if a coupon with an `amount_off` of 20000 is applied to it.

Parameters

amount_off optional

A positive integer representing the amount to subtract from an invoice total (required if `percent_off` is not passed).

currency optional

Three-letter [ISO code for the currency](#) of the `amount_off` parameter (required if `amount_off` is passed).

duration optional enum

Specifies how long the discount will be in effect if used on a subscription. Can be `forever`, `once`, or `repeating`. Defaults to `once`.

Possible enum values

`once`

`repeating`

`forever`

duration_in_months optional

Required only if `duration` is `repeating`, in which case it must be a positive integer that specifies the number of months the discount will be in effect.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

name optional

Name of the coupon displayed to customers on, for instance invoices, or receipts. By default the `id` is shown if `name` is not set.

percent_off optional

A positive float larger than 0, and smaller or equal to 100, that represents the discount the coupon will apply (required if `amount_off` is not passed).

More parameters

✓ **id** optional

Unique string of your choice that will be used to identify this coupon when applying it to a customer. If you don't want to specify a particular code, you can leave the ID blank and we'll generate a random code for you.

✓ **applies_to** optional dictionary

A hash containing directions for what this Coupon will apply discounts to.

✓ **max_redemptions** optional

A positive integer specifying the number of times the coupon can be redeemed before it's no longer valid. For example, you might have a 50% off coupon that the first 20 readers of your blog can use.

✓ **redeem_by** optional

Unix timestamp specifying the last time at which the coupon can be redeemed. After the redeem_by date, the coupon can no longer be applied to new customers.

Returns

Returns the coupon object.

Retrieve a coupon

Retrieves the coupon with the given ID.

Parameters

No parameters.

Returns

Returns a coupon if a valid coupon ID was provided. Returns [an error](#) otherwise.

Update a coupon

Updates the metadata of a coupon. Other coupon details (currency, duration, amount_off) are, by design, not editable.

Parameters

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

name optional

Name of the coupon displayed to customers on, for instance invoices, or receipts. By default the `id` is shown if `name` is not set.

Returns

The newly updated coupon object if the call succeeded. Otherwise, this call returns [an error](#), such as if the coupon has been deleted.

Delete a coupon

You can delete coupons via the [coupon management](#) page of the Stripe dashboard. However, deleting a coupon does not affect any customers who have already applied the coupon; it means that new customers can't redeem the coupon. You can also delete coupons via the API.

Parameters

No parameters.

Returns

An object with the deleted coupon's ID and a deleted flag upon success. Otherwise, this call returns [an error](#), such as if the coupon has already been deleted.

List all coupons

Returns a list of your coupons.

Parameters

[Collapse all](#)

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` coupons, starting after coupon `starting_after`. Each entry in the array is a separate coupon object. If no more coupons are available, the resulting array will be empty. This request should never return an error.

Promotion Code

A Promotion Code represents a customer-redeemable code for a coupon. It can be used to create multiple codes for a single coupon.

The promotion code object

Attributes

id string

Unique identifier for the object.

code string

The customer-facing code. Regardless of case, this code must be unique across all active promotion codes for each customer.

coupon hash, [coupon object](#)

Hash describing the coupon for this promotion code.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

More attributes

✓ **object** string, value is "promotion_code"

String representing the object's type. Objects of the same type share the same value.

✓ **active** boolean

Whether the promotion code is currently active. A promotion code is only active if the coupon is also valid.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **customer** string [EXPANDABLE](#)

The customer that this promotion code can be used by.

✓ **expires_at** timestamp

Date at which the promotion code can no longer be redeemed.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **max_redemptions** positive integer

Maximum number of times this promotion code can be redeemed.

✓ **restrictions** hash

Settings that restrict the redemption of the promotion code.

✓ **times_redeemed** positive integer or zero

Number of times this promotion code has been used.

Create a promotion code

A promotion code points to a coupon. You can optionally restrict the code to a specific customer, redemption limit, and expiration date.

Parameters

coupon REQUIRED

The coupon for this promotion code.

code optional

The customer-facing code. Regardless of case, this code must be unique across all active promotion codes for a specific customer. If left blank, we will generate one automatically.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ **active** optional

Whether the promotion code is currently active.

✓ **customer** optional

The customer that this promotion code can be used by. If not set, the promotion code can be used by all customers.

✓ **expires_at** optional

The timestamp at which this promotion code will expire. If the coupon has specified a `redeems_by`, then this value cannot be after the coupon's `redeems_by`.

✓ **max_redemptions** optional

A positive integer specifying the number of times the promotion code can be redeemed. If the coupon has specified a `max_redemptions`, then this value cannot be greater than the coupon's `max_redemptions`.

✓ **restrictions** optional dictionary

Settings that restrict the redemption of the promotion code.

Returns

Returns the promotion code object.

Update a promotion code

Updates the specified promotion code by setting the values of the parameters passed. Most fields are, by design, not editable.

Parameters

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to **metadata**.

More parameters

✓ **active** optional

Whether the promotion code is currently active. A promotion code can only be reactivated when the coupon is still valid and the promotion code is otherwise redeemable.

Returns

The updated promotion code object is returned upon success. Otherwise, this call returns an [error](#).

Retrieve a promotion code

Retrieves the promotion code with the given ID. In order to retrieve a promotion code by the customer-facing `code` use [list](#) with the desired `code`.

Parameters

No parameters.

Returns

Returns a promotion code if a valid promotion code ID was provided. Returns [an error](#) otherwise.

List all promotion codes

Returns a list of your promotion codes.

Parameters

✓ **active** optional

Filter promotion codes by whether they are active.

✓ **code** optional

Only return promotion codes that have this case-insensitive code.

✓ **coupon** optional

Only return promotion codes for this coupon.

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **customer** optional

Only return promotion codes that are restricted to this customer.

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`,

your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` promotion codes, starting after promotion code `starting_after`. Each entry in the array is a separate promotion code object. If no more promotion codes are available, the resulting array will be empty. This request should never return an error.

Discounts

A discount represents the actual application of a coupon to a particular customer. It contains information about when the discount began and when it will end.

Related guide: [Applying Discounts to Subscriptions](#).

The discount object

Attributes

id string

The ID of the discount object. Discounts cannot be fetched by ID. Use `expand[]="discounts"` in API calls to expand discount IDs in an array.

coupon hash, [coupon object](#)

Hash describing the coupon applied to create this discount.

customer string EXPANDABLE

The ID of the customer associated with this discount.

end timestamp

If the coupon has a duration of `repeating`, the date that this discount will end. If the coupon has a duration of `once` or `forever`, this attribute will be null.

start timestamp

Date that the coupon was applied.

subscription string

The subscription that this coupon is applied to, if it is applied to a particular subscription.

More attributes

✓ **object** string, value is "discount"

String representing the object's type. Objects of the same type share the same value.

✓ **checkout_session** string

The Checkout session that this coupon is applied to, if it is applied to a particular session in payment mode. Will not be present for subscription mode.

✓ **invoice** string

The invoice that the discount's coupon was applied to, if it was applied directly to a particular invoice.

✓ **invoice_item** string

The invoice item `id` (or invoice line item `id` for invoice line items of type='subscription') that the discount's coupon was applied to, if it was applied directly to a particular invoice item or invoice line item.

✓ **promotion_code** string EXPANDABLE

The promotion code applied to create this discount.

Delete a customer discount

Removes the currently applied discount on a customer.

Parameters

No parameters.

Returns

An object with a deleted flag set to true upon success. This call returns [an error](#) otherwise, such as if no discount exists on this customer.

Delete a subscription discount

Removes the currently applied discount on a subscription.

Parameters

No parameters.

Returns

An object with a deleted flag set to true upon success. This call returns [an error](#) otherwise, such as if no discount exists on this subscription.

Tax Code

[Tax codes](#) classify goods and services for tax purposes.

The tax code object

Attributes

id string

Unique identifier for the object.

object string, value is "tax_code"

String representing the object's type. Objects of the same type share the same value.

description string

A detailed description of which types of products the tax code represents.

name string

A short name for the tax code.

List all tax codes

A list of [all tax codes available](#) to add to Products in order to allow specific tax calculations.

Parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a data property that contains an array of up to limit tax codes, starting after tax code starting_after. Each entry in the array is a separate tax code object. If no more tax codes are available, the resulting array will be empty. This request should never return an error.

Retrieve a tax code

Retrieves the details of an existing tax code. Supply the unique tax code ID and Stripe will return the corresponding tax code information.

Parameters

No parameters.

Returns

Returns a tax code object if a valid identifier was provided.

Tax Rate

Tax rates can be applied to [invoices](#), [subscriptions](#) and [Checkout Sessions](#) to collect tax.

Related guide: [Tax Rates](#).

The tax rate object

Attributes

id string

Unique identifier for the object.

active boolean

Defaults to `true`. When set to `false`, this tax rate cannot be used with new applications or Checkout Sessions, but will still work for subscriptions and invoices that already have it set.

country string

Two-letter country code ([ISO 3166-1 alpha-2](#)).

description string

An arbitrary string attached to the tax rate for your internal use only. It will not be visible to your customers.

display_name string

The display name of the tax rates as it will appear to your customer on their receipt email, PDF, and the hosted invoice page.

inclusive boolean

This specifies if the tax rate is inclusive or exclusive.

jurisdiction string

The jurisdiction for the tax rate. You can use this label field for tax reporting purposes. It also appears on your customer's invoice.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

percentage decimal

This represents the tax rate percent out of 100.

state string

[ISO 3166-2 subdivision code](#), without country prefix. For example, "NY" for New York, United States.

More attributes

`**object** string, value is "tax_rate"

String representing the object's type. Objects of the same type share the same value.

`**created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

`**livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

`**tax_type** string

The high-level tax type, such as `vat` or `sales_tax`.

Create a tax rate

Creates a new tax rate.

Parameters

display_name REQUIRED

The display name of the tax rate, which will be shown to users.

inclusive REQUIRED

This specifies if the tax rate is inclusive or exclusive.

percentage REQUIRED

This represents the tax rate percent out of 100.

active optional

Flag determining whether the tax rate is active or inactive (archived). Inactive tax rates cannot be used with new applications or Checkout Sessions, but will still work for subscriptions and invoices that already have it set.

country optional

Two-letter country code ([ISO 3166-1 alpha-2](#)).

description optional

An arbitrary string attached to the tax rate for your internal use only. It will not be visible to your customers.

jurisdiction optional

The jurisdiction for the tax rate. You can use this label field for tax reporting purposes. It also appears on your customer's invoice.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

state optional

[ISO 3166-2 subdivision code](#), without country prefix. For example, "NY" for New York, United States.

More parameters

✓ **tax_type** optional

The high-level tax type, such as `vat` or `sales_tax`.

Returns

The created tax rate object.

Retrieves a tax rate

Retrieves a tax rate with the given ID

Parameters

No parameters.

Returns

Returns an tax rate if a valid tax rate ID was provided. Returns [an error](#) otherwise.

Update a tax rate

Updates an existing tax rate.

Parameters

active optional

Flag determining whether the tax rate is active or inactive (archived). Inactive tax rates cannot be used with new applications or Checkout Sessions, but will still work for subscriptions and invoices that already have it set.

country optional

Two-letter country code ([ISO 3166-1 alpha-2](#)).

description optional

An arbitrary string attached to the tax rate for your internal use only. It will not be visible to your customers.

display_name optional

The display name of the tax rate, which will be shown to users.

jurisdiction optional

The jurisdiction for the tax rate. You can use this label field for tax reporting purposes. It also appears on your customer's invoice.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

state optional

[ISO 3166-2 subdivision code](#), without country prefix. For example, "NY" for New York, United States.

More parameters

✓ tax_type optional

The high-level tax type, such as `vat` or `sales_tax`.

Returns

The updated tax rate.

List all tax rates

Returns a list of your tax rates. Tax rates are returned sorted by creation date, with the most recently created tax rates appearing first.

Parameters

active optional

Optional flag to filter by tax rates that are either active or inactive (archived).

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **inclusive** optional

Optional flag to filter by tax rates that are inclusive (or those that are not inclusive).

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` tax rates, starting after tax rate `starting_after`. Each entry in the array is a separate tax rate object. If no more tax rates are available, the resulting array will be empty. This request should never return an error.

Shipping Rates

Shipping rates describe the price of shipping presented to your customers and can be applied to [Checkout Sessions](#) to collect shipping costs.

The shipping rate object

Attributes

id string

Unique identifier for the object.

active boolean

Whether the shipping rate can be used for new purchases. Defaults to `true`.

display_name string

The name of the shipping rate, meant to be displayable to the customer. This will appear on CheckoutSessions.

fixed_amount hash

Describes a fixed amount to charge for shipping. Must be present if type is `fixed_amount`.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

type enum

The type of calculation to use on the shipping rate. Can only be `fixed_amount` for now.

Possible enum values

`fixed_amount`

More attributes

✓ **object** string, value is "shipping_rate"

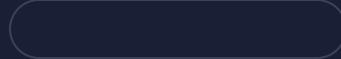
String representing the object's type. Objects of the same type share the same value.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **delivery_estimate** hash

The estimated range for how long shipping will take, meant to be displayable to the customer. This will appear on CheckoutSessions.



✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **tax_behavior** string

Specifies whether the rate is considered inclusive of taxes or exclusive of taxes. One of `inclusive`, `exclusive`, or `unspecified`.

✓ **tax_code** string EXPANDABLE

A tax code ID. The Shipping tax code is `txcd_92010001`.

Create a shipping rate

Creates a new shipping rate object.

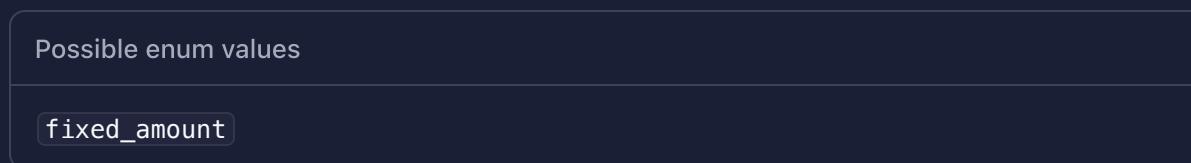
Parameters

display_name REQUIRED

The name of the shipping rate, meant to be displayable to the customer. This will appear on CheckoutSessions.

type REQUIRED

The type of calculation to use on the shipping rate. Can only be `fixed_amount` for now.



fixed_amount optional dictionary

Describes a fixed amount to charge for shipping. Must be present if type is `fixed_amount`.



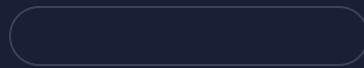
metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to **metadata**.

More parameters

✓ **delivery_estimate** optional dictionary

The estimated range for how long shipping will take, meant to be displayable to the customer. This will appear on CheckoutSessions.



✓ **tax_behavior** optional

Specifies whether the rate is considered inclusive of taxes or exclusive of taxes. One of **inclusive**, **exclusive**, or **unspecified**.

✓ **tax_code** optional

A **tax code** ID. The Shipping tax code is **txcd_92010001**.

Returns

Returns a shipping rate object if the call succeeded.

Retrieve a shipping rate

Returns the shipping rate object with the given ID.

Parameters

No parameters.

Returns

Returns a shipping rate object if a valid identifier was provided.

Update a shipping rate

Updates an existing shipping rate object.

Parameters

active optional

Whether the shipping rate can be used for new purchases. Defaults to `true`.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the modified shipping rate object if the call succeeded.

List all shipping rates

Returns a list of your shipping rates.

Parameters

active optional

Only return shipping rates that are active or inactive.

created optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



currency optional

Only return shipping rates for the given currency.

More parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` shipping rates, starting after shipping rate `starting_after`. Each entry in the array is a separate shipping rate object. If no more shipping rates are available, the resulting array will be empty. This require should never return an error.

Sessions

A Checkout Session represents your customer's session as they pay for one-time purchases or subscriptions through [Checkout](#) or [Payment Links](#). We recommend creating a new Session each time your customer attempts to pay.

Once payment is successful, the Checkout Session will contain a reference to the [Customer](#), and either the successful [PaymentIntent](#) or an active [Subscription](#).

You can create a Checkout Session on your server and pass its ID to the client to begin Checkout.

Related guide: [Checkout Server Quickstart](#).

The Session object

Attributes

id string

Unique identifier for the object. Used to pass to `redirectToCheckout` in Stripe.js.

cancel_url string

The URL the customer will be directed to if they decide to cancel payment and return to your website.

client_reference_id string

A unique string to reference the Checkout Session. This can be a customer ID, a cart ID, or similar, and can be used to reconcile the Session with your internal systems.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

customer string EXPANDABLE

The ID of the customer for this Session. For Checkout Sessions in `payment` or `subscription` mode, Checkout will create a new customer object based on information provided during the payment flow unless an existing customer was provided when the Session was created.

customer_email string

If provided, this value will be used when the Customer object is created. If not provided, customers will be asked to enter their email address. Use this parameter to prefill customer data if you already have an email on file. To access information about the customer once the payment flow is complete, use the `customer` attribute.

line_items list EXPANDABLE

The line items purchased by the customer. This field is not included by default. To include it in the response, [expand](#) the `line_items` field.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

mode enum

The mode of the Checkout Session.

Possible enum values

payment

Accept one-time payments for cards, iDEAL, and more.

setup

Save payment details to charge your customers later.

subscription

Use Stripe Billing to set up fixed-price subscriptions.

payment_intent string EXPANDABLE

The ID of the PaymentIntent for Checkout Sessions in `payment` mode.

payment_method_types array containing strings

A list of the types of payment methods (e.g. card) this Checkout Session is allowed to accept.

payment_status enum

The payment status of the Checkout Session, one of `paid`, `unpaid`, or `no_payment_required`. You can use this value to decide when to fulfill your customer's order.

Possible enum values

paid

The payment funds are available in your account.

unpaid

The payment funds are not yet available in your account.

no_payment_required

The Checkout Session is in `setup` mode and doesn't require a payment at this time.

success_url string

The URL the customer will be directed to after the payment or subscription creation is successful.

More attributes

✓ `object` string, value is "checkout.session"

String representing the object's type. Objects of the same type share the same value.

✓ `after_expiration` hash

When set, provides configuration for actions to take if this Checkout Session expires.

✓ **allow_promotion_codes** boolean

Enables user redeemable promotion codes.

✓ **amount_subtotal** integer

Total of all items before discounts or taxes are applied.

✓ **amount_total** integer

Total of all items after discounts and taxes are applied.

✓ **automatic_tax** hash

Details on the state of automatic tax for the session, including the status of the latest tax calculation.

✓ **billing_address_collection** enum

Describes whether Checkout should collect the customer's billing address.

Possible enum values

auto DEFAULT

Checkout will only collect the billing address when necessary.

required

Checkout will always collect the customer's billing address.

✓ **consent** hash

Results of `consent_collection` for this session.

✓ **consent_collection** hash

When set, provides configuration for the Checkout Session to gather active consent from customers.

✓ **customer_creation** enum

Configure whether a Checkout Session creates a Customer when the Checkout Session completes.

Possible enum values

`if_required`

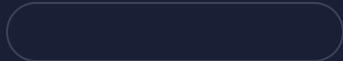
The Checkout Session will only create a **Customer** if it is required for Session confirmation. Currently, only `subscription` mode Sessions require a Customer.

`always`

The Checkout Session will always create a **Customer** when a Session confirmation is attempted.

✓ `customer_details` hash

The customer details including the customer's tax exempt status and the customer's tax IDs. Only present on Sessions in `payment` or `subscription` mode.



✓ `expires_at` timestamp

The timestamp at which the Checkout Session will expire.

✓ `livemode` boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ `locale` enum

The IETF language tag of the locale Checkout is displayed in. If blank or `auto`, the browser's locale is used.

Possible enum values

`auto`

`bg`

`cs`

`da`

`de`

`el`

`en`

[Show 34 more](#)

✓ `payment_link` string EXPANDABLE

The ID of the Payment Link that created this Session.

✓ **payment_method_options** hash

Payment-method-specific configuration for the PaymentIntent or SetupIntent of this CheckoutSession.

✓ **phone_number_collection** hash

Details on the state of phone number collection for the session.

✓ **recovered_from** string

The ID of the original expired Checkout Session that triggered the recovery flow.

✓ **setup_intent** string EXPANDABLE

The ID of the SetupIntent for Checkout Sessions in `setup` mode.

✓ **shipping** hash

Shipping information for this Checkout Session.

✓ **shipping_address_collection** hash

When set, provides configuration for Checkout to collect a shipping address from a customer.

✓ **shipping_options** array of hashes

The shipping rate options applied to this Session.

✓ **shipping_rate** string EXPANDABLE

The ID of the ShippingRate for Checkout Sessions in `payment` mode.

✓ **status** enum

The status of the Checkout Session, one of `open`, `complete`, or `expired`.

Possible enum values

<code>open</code>	The checkout session is still in progress. Payment processing has not started
<code>complete</code>	The checkout session is complete. Payment processing may still be in progress

expired

The checkout session has expired. No further processing will occur

✓ **submit_type** enum

Describes the type of transaction being performed by Checkout in order to customize relevant text on the page, such as the submit button. `submit_type` can only be specified on Checkout Sessions in `payment` mode, but not Checkout Sessions in `subscription` or `setup` mode.

Possible enum values

`auto`

`pay`

`book`

`donate`

✓ **subscription** string EXPANDABLE

The ID of the subscription for Checkout Sessions in `subscription` mode.

✓ **tax_id_collection** hash

Details on the state of tax ID collection for the session.



✓ **total_details** hash

Tax and discount details for the computed total amount.



✓ **url** string

The URL to the Checkout Session. Redirect customers to this URL to take them to Checkout.

If you're using [Custom Domains](#), the URL will use your subdomain. Otherwise, it'll use

`checkout.stripe.com`.

Create a Session

Creates a Session object.

Parameters

cancel_url REQUIRED

The URL the customer will be directed to if they decide to cancel payment and return to your website.

mode REQUIRED CONDITIONALLY

The mode of the Checkout Session. Required when using prices or `setup` mode. Pass `subscription` if the Checkout Session includes at least one recurring item.

Possible enum values

`payment`

Accept one-time payments for cards, iDEAL, and more.

`setup`

Save payment details to charge your customers later.

`subscription`

Use Stripe Billing to set up fixed-price subscriptions.

success_url REQUIRED

The URL to which Stripe should send customers when payment or setup is complete. If you'd like to use information from the successful Checkout Session on your page, read the guide on [customizing your success page](#).

client_reference_id optional

A unique string to reference the Checkout Session. This can be a customer ID, a cart ID, or similar, and can be used to reconcile the session with your internal systems.

customer optional

ID of an existing Customer, if one exists. In `payment` mode, the customer's most recent card payment method will be used to prefill the email, name, card details, and billing address on the Checkout page. In `subscription` mode, the customer's **default payment method** will be used if it's a card, and otherwise the most recent card will be used. A valid billing address, billing name and billing email are required on the payment method for Checkout to prefill the customer's card details.

If the Customer already has a valid `email` set, the email will be prefilled and not editable in Checkout. If the Customer does not have a valid `email`, Checkout will set the email entered during the session on the Customer.

If blank for Checkout Sessions in `payment` or `subscription` mode, Checkout will create a new Customer object based on information provided during the payment flow.

You can set `payment_intent_data.setup_future_usage` to have Checkout automatically attach the payment method to the Customer you pass in for future reuse.

customer_email optional

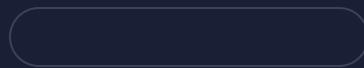
If provided, this value will be used when the Customer object is created. If not provided, customers will be asked to enter their email address. Use this parameter to prefill customer data if you already have an email on file. To access information about the customer once a session is complete, use the `customer` field.

line_items optional array of hashes

A list of items the customer is purchasing. Use this parameter to pass one-time or recurring Prices.

For `payment` mode, there is a maximum of 100 line items, however it is recommended to consolidate line items if there are more than a few dozen.

For `subscription` mode, there is a maximum of 20 line items with recurring Prices and 20 line items with one-time Prices. Line items with one-time Prices will be on the initial invoice only.



metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

payment_method_types optional enum

A list of the types of payment methods (e.g., `card`) this Checkout Session can accept.

Read more about the supported payment methods and their requirements in our [payment method details guide](#).

If multiple payment methods are passed, Checkout will dynamically reorder them to prioritize the most relevant payment methods based on the customer's location and other characteristics.

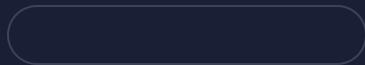
Possible enum values
<code>card</code>
<code>acss_debit</code>
<code>afterpay_clearpay</code>
<code>alipay</code>
<code>au_becs_debit</code>
<code>bacs_debit</code>
<code>bancontact</code>

Show 15 more

More parameters

✓ **after_expiration** optional dictionary

Configure actions after a Checkout Session has expired.



✓ **allow_promotion_codes** optional

Enables user redeemable promotion codes.

✓ **automatic_tax** optional dictionary

Settings for automatic tax lookup for this session and resulting payments, invoices, and subscriptions.



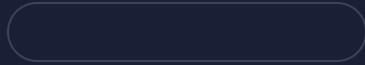
✓ **billing_address_collection** optional enum

Specify whether Checkout should collect the customer's billing address.

Possible enum values
<code>auto</code> DEFAULT Checkout will only collect the billing address when necessary.
<code>required</code> Checkout will always collect the customer's billing address.

✓ **consent_collection** optional dictionary

Configure fields for the Checkout Session to gather active consent from customers.



✓ **customer_creation** optional enum

Configure whether a Checkout Session creates a **Customer** during Session confirmation.

When a Customer is not created, you can still retrieve email, address, and other customer data entered in Checkout with `customer_details`.

Sessions that don't create Customers instead create **Guest Customers** in the Dashboard.

Promotion codes limited to first time customers will return invalid for these Sessions.

Can only be set in `payment` and `setup` mode.

Possible enum values

`if_required`

The Checkout Session will only create a **Customer** if it is required for Session confirmation. Currently, only `subscription` mode Sessions require a Customer.

`always`

The Checkout Session will always create a **Customer** when a Session confirmation is attempted.

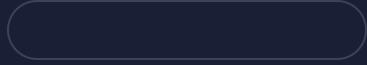
✓ `customer_update` optional dictionary

Controls what fields on Customer can be updated by the Checkout Session. Can only be provided when `customer` is provided.



✓ `discounts` optional array of hashes

The coupon or promotion code to apply to this Session. Currently, only up to one may be specified.



✓ `expires_at` optional

The Epoch time in seconds at which the Checkout Session will expire. It can be anywhere from 1 to 24 hours after Checkout Session creation. By default, this value is 24 hours from creation.

✓ `locale` optional enum

The IETF language tag of the locale Checkout is displayed in. If blank or `auto`, the browser's locale is used.

Possible enum values

`auto`

`bg`

`cs`

`da`

`de`

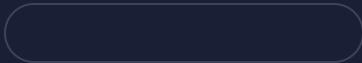
`el`

`en`

[Show 34 more](#)

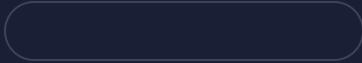
✓ **payment_intent_data** optional dictionary

A subset of parameters to be passed to PaymentIntent creation for Checkout Sessions in `payment` mode.



✓ **payment_method_options** optional dictionary

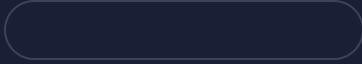
Payment-method-specific configuration.



✓ **phone_number_collection** optional dictionary

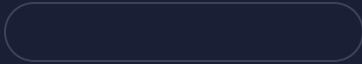
Controls phone number collection settings for the session.

We recommend that you review your privacy policy and check with your legal contacts before using this feature. Learn more about [collecting phone numbers with Checkout](#).



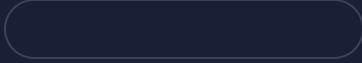
✓ **setup_intent_data** optional dictionary

A subset of parameters to be passed to SetupIntent creation for Checkout Sessions in `setup` mode.



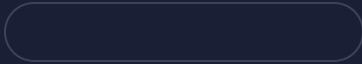
✓ **shipping_address_collection** optional dictionary

When set, provides configuration for Checkout to collect a shipping address from a customer.



✓ **shipping_options** optional array of hashes

The shipping rate options to apply to this Session.



✓ **submit_type** optional enum

Describes the type of transaction being performed by Checkout in order to customize relevant text on the page, such as the submit button. `submit_type` can only be specified on Checkout Sessions in `payment` mode, but not Checkout Sessions in `subscription` or `setup` mode.

Possible enum values

`auto`

`pay`

[book](#)

[donate](#)

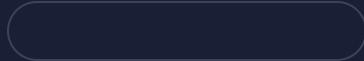
✓ **subscription_data** optional dictionary

A subset of parameters to be passed to subscription creation for Checkout Sessions in [subscription](#) mode.



✓ **tax_id_collection** optional dictionary

Controls tax ID collection settings for the session.



Returns

Returns a Session object.

Expire a Session

A Session can be expired when it is in one of these statuses: [open](#)

After it expires, a customer can't complete a Session and customers loading the Session see a message saying the Session is expired.

Parameters

No parameters.

Returns

Returns a Session object if the expiration succeeded. Returns an error if the Session has already expired or isn't in an expireable state.

Retrieve a Session

Retrieves a Session object.

Parameters

No parameters.

Returns

Returns a Session object.

List all Checkout Sessions

Returns a list of Checkout Sessions.

Parameters

payment_intent optional

Only return the Checkout Session for the PaymentIntent specified.

subscription optional

Only return the Checkout Session for the subscription specified.

More parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with

`obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` Checkout Sessions, starting after Checkout Session `starting_after`. Each entry in the array is a separate Checkout Session object. If no more Checkout Sessions are available, the resulting array will be empty.

Retrieve a Checkout Session's line items

When retrieving a Checkout Session, there is an includable `line_items` property containing the first handful of those items. There is also a URL where you can retrieve the full (paginated) list of line items.

Parameters

✓ `ending_before` optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ `limit` optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ `starting_after` optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` Checkout Session line items, starting after Line Item `starting_after`. Each entry in the array is a separate Line Item object. If no more line items are available, the resulting array will be empty.

Payment Link

A payment link is a shareable URL that will take your customers to a hosted payment page. A payment link can be shared and used multiple times.

When a customer opens a payment link it will open a new [checkout session](#) to render the payment page. You can use [checkout session events](#) to track payments through payment links.

Related guide: [Payment Links API](#)

The payment link object

Attributes

id string

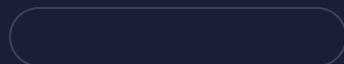
Unique identifier for the object.

active boolean

Whether the payment link's `url` is active. If `false`, customers visiting the URL will be shown a page saying that the link has been deactivated.

line_items list EXPANDABLE

The line items representing what is being sold. This field is not included by default. To include it in the response, [expand](#) the `line_items` field.



metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

payment_method_types array of enum values

The list of payment method types that customers can use. When `null`, Stripe will dynamically show relevant payment methods you've enabled in your [payment method settings](#).

Possible enum values

card

url string

The public URL that can be shared with customers.

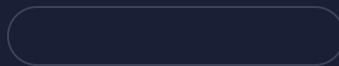
More attributes

✓ **object** string, value is "payment_link"

String representing the object's type. Objects of the same type share the same value.

✓ **after_completion** hash

Behavior after the purchase is complete.



✓ **allow_promotion_codes** boolean

Whether user redeemable promotion codes are enabled.

✓ **application_fee_amount** integer CONNECT ONLY

The amount of the application fee (if any) that will be requested to be applied to the payment and transferred to the application owner's Stripe account.

✓ **application_fee_percent** decimal CONNECT ONLY

This represents the percentage of the subscription invoice subtotal that will be transferred to the application owner's Stripe account.

✓ **automatic_tax** hash

Configuration details for automatic tax collection.



✓ **billing_address_collection** enum

Configuration for collecting the customer's billing address.

Possible enum values

auto DEFAULT

Checkout will only collect the billing address when necessary.

required

Checkout will always collect the customer's billing address.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **on_behalf_of** string EXPANDABLE CONNECT ONLY

The account on behalf of which to charge. See the [Connect documentation](#) for details.

✓ **phone_number_collection** hash

Controls phone number collection settings during checkout.

✓ **shipping_address_collection** hash

Configuration for collecting the customer's shipping address.

✓ **subscription_data** hash

When creating a subscription, the specified configuration data will be used. There must be at least one line item with a recurring price to use `subscription_data`.

✓ **transfer_data** hash CONNECT ONLY

The account (if any) the payments will be attributed to for tax reporting, and where funds from each payment will be transferred to.

Create a payment link

Creates a payment link.

Parameters

line_items REQUIRED

The line items representing what is being sold. Each line item represents an item being sold. Up to 20 line items are supported.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to **metadata**. Metadata associated with this Payment Link will automatically be copied to **checkout sessions** created by this payment link.

payment_method_types optional enum

The list of payment method types that customers can use. Only **card** is supported. If no value is passed, Stripe will dynamically show relevant payment methods from your **payment method settings** (20+ payment methods **supported**).

Possible enum values

card

More parameters

✓ **after_completion** optional dictionary

Behavior after the purchase is complete.



✓ **allow_promotion_codes** optional

Enables user redeemable promotion codes.

✓ **application_fee_amount** optional **CONNECT ONLY**

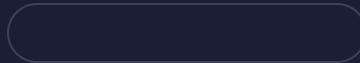
The amount of the application fee (if any) that will be requested to be applied to the payment and transferred to the application owner's Stripe account. Can only be applied when there are no line items with recurring prices.

✓ **application_fee_percent** optional **CONNECT ONLY**

A non-negative decimal between 0 and 100, with at most two decimal places. This represents the percentage of the subscription invoice subtotal that will be transferred to the application owner's Stripe account. There must be at least 1 line item with a recurring price to use this field.

✓ **automatic_tax** optional dictionary

Configuration for automatic tax collection.



✓ **billing_address_collection** optional enum

Configuration for collecting the customer's billing address.

Possible enum values

`auto` DEFAULT

Checkout will only collect the billing address when necessary.

`required`

Checkout will always collect the customer's billing address.

✓ **on_behalf_of** optional CONNECT ONLY

The account on behalf of which to charge.

✓ **phone_number_collection** optional dictionary

Controls phone number collection settings during checkout.

We recommend that you review your privacy policy and check with your legal contacts.



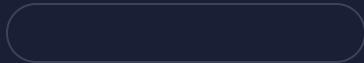
✓ **shipping_address_collection** optional dictionary

Configuration for collecting the customer's shipping address.



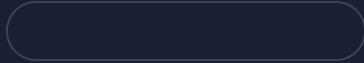
✓ **subscription_data** optional dictionary

When creating a subscription, the specified configuration data will be used. There must be at least one line item with a recurring price to use `subscription_data`.



✓ **transfer_data** optional dictionary CONNECT ONLY

The account (if any) the payments will be attributed to for tax reporting, and where funds from each payment will be transferred to.



Returns

Returns the payment link.

Retrieve payment link

Retrieve a payment link.

Parameters

No parameters.

Returns

Returns the payment link.

Update a payment link

Updates a payment link.

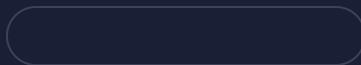
Parameters

active optional

Whether the payment link's `url` is active. If `false`, customers visiting the URL will be shown a page saying that the link has been deactivated.

line_items optional array of hashes

The line items representing what is being sold. Each line item represents an item being sold. Up to 20 line items are supported.



metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`. Metadata associated with this Payment Link will automatically be copied to [checkout sessions](#) created by this payment link.

payment_method_types optional enum

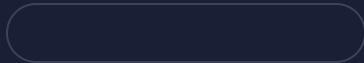
The list of payment method types that customers can use. Only `card` is supported. Pass an empty string to enable automatic payment methods that use your [payment method settings](#).

Possible enum values

`card`

✓ **after_completion** optional dictionary

Behavior after the purchase is complete.



✓ **allow_promotion_codes** optional

Enables user redeemable promotion codes.



✓ **billing_address_collection** optional enum

Configuration for collecting the customer's billing address.

Possible enum values

`auto` DEFAULT

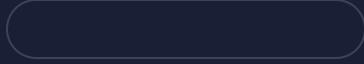
Checkout will only collect the billing address when necessary.

`required`

Checkout will always collect the customer's billing address.

✓ **shipping_address_collection** optional dictionary

Configuration for collecting the customer's shipping address.



Returns

Updated payment link.

List all payment links

Returns a list of your payment links.

Parameters

active optional

Only return payment links that are active or inactive (e.g., pass `false` to list all inactive payment links).

More parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` payment links, starting after payment link `starting_after`. Each entry in the array is a separate payment link object. If no more payment links are available, the resulting array will be empty. This request should never return an error.

Retrieve a payment link's line items

When retrieving a payment link, there is an includable `line_items` property containing the first handful of those items. There is also a URL where you can retrieve the full (paginated) list of line items.

Parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`,

your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` payment link line items, starting after Line Item `starting_after`. Each entry in the array is a separate Line Item object. If no more line items are available, the resulting array will be empty.

Credit Note

Issue a credit note to adjust an invoice's amount after the invoice is finalized.

Related guide: [Credit Notes](#).

The credit note object

Attributes

id string

Unique identifier for the object.

currency currency

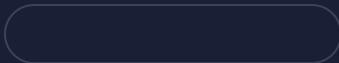
Three-letter ISO currency code, in lowercase. Must be a supported currency.

invoice string EXPANDABLE

ID of the invoice.

lines list

Line items that make up the credit note



memo string

Customer-facing text that appears on the credit note PDF.

metadata hash

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

reason string

Reason for issuing this credit note, one of `duplicate`, `fraudulent`, `order_change`, or `product_unsatisfactory`

status string

Status of this credit note, one of `issued` or `void`. Learn more about [voiding credit notes](#).

subtotal integer

The integer amount in cents representing the amount of the credit note, excluding tax and invoice level discounts.

total integer

The integer amount in cents representing the total amount of the credit note, including tax and all discount.

More attributes

object string, value is "credit_note"

String representing the object's type. Objects of the same type share the same value.

amount integer

The integer amount in cents representing the total amount of the credit note, including tax.

created timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

customer string EXPANDABLE

ID of the customer.

customer_balance_transaction string EXPANDABLE

Customer balance transaction related to this credit note.

✓ **discount_amount** integer DEPRECATED

The integer amount in cents representing the total amount of discount that was credited.

✓ **discount_amounts** array of hashes

The aggregate amounts calculated per discount for all line items.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **number** string

A unique number that identifies this particular credit note and appears on the PDF of the credit note and its associated invoice.

✓ **out_of_band_amount** integer

Amount that was credited outside of Stripe.

✓ **pdf** string

The link to download the PDF of the credit note.

✓ **refund** string EXPANDABLE

Refund related to this credit note.

✓ **tax_amounts** array of hashes

The aggregate amounts calculated per tax rate for all line items.

✓ **type** string

Type of this credit note, one of `pre_payment` or `post_payment`. A `pre_payment` credit note means it was issued when the invoice was open. A `post_payment` credit note means it was issued when the invoice was paid.

✓ **voided_at** timestamp

The time that the credit note was voided.

The (Credit Note) line item object

Attributes

id string

Unique identifier for the object.

object string, value is "credit_note_line_item"

String representing the object's type. Objects of the same type share the same value.

amount integer

The integer amount in cents representing the gross amount being credited for this line item, excluding (exclusive) tax and discounts.

description string

Description of the item being credited.

discount_amount integer DEPRECATED

The integer amount in cents representing the discount being credited for this line item.

discount_amounts array of hashes

The amount of discount calculated per discount for this line item

invoice_line_item string

ID of the invoice line item being credited

livemode boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

quantity integer

The number of units of product being credited.

tax_amounts array of hashes

The amount of tax calculated per tax rate for this line item

tax_rates array of hashes

The tax rates which apply to the line item.

type string

The type of the credit note line item, one of `invoice_line_item` or `custom_line_item`.

When the type is `invoice_line_item` there is an additional `invoice_line_item` property on the resource the value of which is the id of the credited line item on the invoice.

unit_amount integer

The cost of each unit of product being credited.

unit_amount_decimal decimal string

Same as `unit_amount`, but contains a decimal value with at most 12 decimal places.

Preview a credit note

Get a preview of a credit note without creating it.

Parameters

invoice REQUIRED

ID of the invoice.

lines optional array of hashes

Line items that make up the credit note.

**memo** optional

The credit note's memo appears on the credit note PDF.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

reason optional

Reason for issuing this credit note, one of `duplicate`, `fraudulent`, `order_change`, or `product_unsatisfactory`

[More parameters](#)

✓ **amount** optional

The integer amount in cents representing the total amount of the credit note.

✓ **credit_amount** optional

The integer amount in cents representing the amount to credit the customer's balance, which will be automatically applied to their next invoice.

✓ **out_of_band_amount** optional

The integer amount in cents representing the amount that is credited outside of Stripe.

✓ **refund** optional

ID of an existing refund to link this credit note to.

✓ **refund_amount** optional

The integer amount in cents representing the amount to refund. If set, a refund will be created for the charge associated with the invoice.

Returns

Returns a credit note object.

Create a credit note

Issue a credit note to adjust the amount of a finalized invoice. For a `status=open` invoice, a credit note reduces its `amount_due`. For a `status=paid` invoice, a credit note does not affect its `amount_due`. Instead, it can result in any combination of the following:

- Refund: create a new refund (using `refund_amount`) or link an existing refund (using `refund`).
- Customer balance credit: credit the customer's balance (using `credit_amount`) which will be automatically applied to their next invoice when it's finalized.
- Outside of Stripe credit: record the amount that is or will be credited outside of Stripe (using `out_of_band_amount`).

For post-payment credit notes the sum of the refund, credit and outside of Stripe amounts must equal the credit note total.

You may issue multiple credit notes for an invoice. Each credit note will increment the invoice's `pre_payment_credit_notes_amount` or `post_payment_credit_notes_amount` depending on its `status` at the time of credit note creation.

Parameters

invoice REQUIRED

ID of the invoice.

lines optional array of hashes

Line items that make up the credit note.

memo optional

The credit note's memo appears on the credit note PDF.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

reason optional

Reason for issuing this credit note, one of `duplicate`, `fraudulent`, `order_change`, or `product_unsatisfactory`

More parameters

✓ **amount** optional

The integer amount in cents representing the total amount of the credit note.

✓ **credit_amount** optional

The integer amount in cents representing the amount to credit the customer's balance, which will be automatically applied to their next invoice.

✓ **out_of_band_amount** optional

The integer amount in cents representing the amount that is credited outside of Stripe.

✓ **refund** optional

ID of an existing refund to link this credit note to.

✓ **refund_amount** optional

The integer amount in cents representing the amount to refund. If set, a refund will be created for the charge associated with the invoice.

Returns

Returns a credit note object if the call succeeded.

Retrieve a credit note

Retrieves the credit note object with the given identifier.

Parameters

No parameters.

Returns

Returns a credit note object if a valid identifier was provided.

Update a credit note

Updates an existing credit note.

Parameters

memo optional

Credit note memo.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to **metadata**.

Returns

Returns the updated credit note object if the call succeeded.

Retrieve a credit note's line items

When retrieving a credit note, you'll get a **lines** property containing the first handful of those items. There is also a URL where you can retrieve the full (paginated) list of line items.

Parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

Returns a list of [line_item](#) objects.

Retrieve a credit note preview's line items

When retrieving a credit note preview, you'll get a **lines** property containing the first handful of those items. This URL you can retrieve the full (paginated) list of line items.

Parameters

invoice REQUIRED

ID of the invoice.

lines optional array of hashes

Line items that make up the credit note.

memo optional

The credit note's memo appears on the credit note PDF.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

reason optional

Reason for issuing this credit note, one of `duplicate`, `fraudulent`, `order_change`, or `product_unsatisfactory`

More parameters

✓ **amount** optional

The integer amount in cents representing the total amount of the credit note.

✓ **credit_amount** optional

The integer amount in cents representing the amount to credit the customer's balance, which will be automatically applied to their next invoice.

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **out_of_band_amount** optional

The integer amount in cents representing the amount that is credited outside of Stripe.

✓ **refund** optional

ID of an existing refund to link this credit note to.

✓ **refund_amount** optional

The integer amount in cents representing the amount to refund. If set, a refund will be created for the charge associated with the invoice.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

Returns a list of [line_item objects](#).

Void a credit note

Marks a credit note as void. Learn more about [voiding credit notes](#).

Parameters

No parameters.

Returns

Returns the voided credit note object if the call succeeded.

List all credit notes

Returns a list of credit notes.

Parameters

invoice optional

Only return credit notes for the invoice specified by this invoice ID.

More parameters

✓ **customer** optional

Only return credit notes for the customer specified by this customer ID.

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` credit notes, starting after credit note `starting_after`. Each entry in the array is a separate credit note object. If no more credit notes are available, the resulting array will be empty.

Customer Balance Transaction

Each customer has a `balance` value, which denotes a debit or credit that's automatically applied to their next invoice upon finalization. You may modify the value directly by using the [update customer API](#), or by creating a Customer Balance Transaction, which increments or decrements the customer's `balance` by the specified `amount`.

Related guide: [Customer Balance](#) to learn more.

The customer balance transaction object

Attributes

id string

Unique identifier for the object.

amount integer

The amount of the transaction. A negative value is a credit for the customer's balance, and a positive value is a debit to the customer's `balance`.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

customer string EXPANDABLE

The ID of the customer the transaction belongs to.

description string

An arbitrary string attached to the object. Often useful for displaying to users.

ending_balance integer

The customer's `balance` after the transaction was applied. A negative value decreases the amount due on the customer's next invoice. A positive value increases the amount due on the customer's next invoice.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

type string

Transaction type: `adjustment`, `applied_to_invoice`, `credit_note`, `initial`, `invoice_too_large`, `invoice_too_small`, `unspent_receiver_credit`, or `unapplied_from_invoice`. See the [Customer Balance page](#) to learn more about transaction types.

More attributes

✓ **object** string, value is "customer_balance_transaction"

String representing the object's type. Objects of the same type share the same value.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **credit_note** string EXPANDABLE

The ID of the credit note (if any) related to the transaction.

✓ **invoice** string EXPANDABLE

The ID of the invoice (if any) related to the transaction.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

Create a customer balance transaction

Creates an immutable transaction that updates the customer's credit [balance](#).

Parameters

amount REQUIRED

The integer amount in **cents** to apply to the customer's credit balance.

currency REQUIRED

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#). If the customer's `currency` is set, this value must match it. If the customer's `currency` is not set, it will be updated to this value.

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns a customer balance transaction object if the call succeeded.

Retrieve a customer balance transaction

Retrieves a specific customer balance transaction that updated the customer's [balances](#).

Parameters

No parameters.

Returns

Returns a customer balance transaction object if a valid identifier was provided.

Update a customer credit balance transaction

Most credit balance transaction fields are immutable, but you may update its [description](#) and [metadata](#).

Parameters

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

Returns

Returns a customer balance transaction object if the call succeeded.

List customer balance transactions

Returns a list of transactions that updated the customer's [balances](#).

Parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` customer balance transactions, starting after item `starting_after`. Each entry in the array is a separate customer balance transaction object. If no more items are available, the resulting array will be empty. This request should never return an error.

Customer Portal

The Billing customer portal is a Stripe-hosted UI for subscription and billing management.

A portal configuration describes the functionality and features that you want to provide to your customers through the portal.

A portal session describes the instantiation of the customer portal for a particular customer. By visiting the session's URL, the customer can manage their subscriptions and billing details. For security reasons, sessions are short-lived and will expire if the customer does not visit the URL. Create sessions on-demand when customers intend to manage their subscriptions and billing details.

Learn more in the [integration guide](#).

The portal session object

Attributes

id string

Unique identifier for the object.

object string, value is "billing_portal.session"

String representing the object's type. Objects of the same type share the same value.

configuration string EXPANDABLE

The configuration used by this session, describing the features available.

created timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

customer string

The ID of the customer for this session.

livemode boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

locale enum

The IETF language tag of the locale Customer Portal is displayed in. If blank or auto, the customer's `preferred_locales` or browser's locale is used.

Possible enum values

`auto`

bg

cs

da

de

el

en

on_behalf_of string CONNECT ONLY

The account for which the session was created on behalf of. When specified, only subscriptions and invoices with this `on_behalf_of` account appear in the portal. For more information, see the [docs](#). Use the [Accounts API](#) to modify the `on_behalf_of` account's branding settings, which the portal displays.

return_url string

The URL to redirect customers to when they click on the portal's link to return to your website.

url string

The short-lived URL of the session that gives customers access to the customer portal.

The portal configuration object

Attributes

id string

Unique identifier for the object.

object string, value is "billing_portal.configuration"

String representing the object's type. Objects of the same type share the same value.

active boolean

Whether the configuration is active and can be used to create portal sessions.

application string EXPANDABLE "APPLICATION" CONNECT ONLY

ID of the Connect Application that created the configuration.

business_profile hash

The business information shown to customers in the portal.



created timestamp

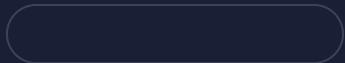
Time at which the object was created. Measured in seconds since the Unix epoch.

default_return_url string

The default URL to redirect customers to when they click on the portal's link to return to your website. This can be **overridden** when creating the session.

features hash

Information about the features available in the portal.



is_default boolean

Whether the configuration is the default. If `true`, this configuration can be managed in the Dashboard and portal sessions will use this configuration unless it is overridden when creating the session.

livemode boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

metadata hash

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

updated timestamp

Time at which the object was last updated. Measured in seconds since the Unix epoch.

Create a portal session

Creates a session of the customer portal.

Parameters

customer REQUIRED

The ID of an existing customer.

return_url REQUIRED IF THE CONFIGURATION'S DEFAULT RETURN URL IS NOT SET

The default URL to redirect customers to when they click on the portal's link to return to your website.

configuration optional

The ID of an existing [configuration](#) to use for this session, describing its functionality and features. If not specified, the session uses the default configuration.

locale optional enum

The IETF language tag of the locale Customer Portal is displayed in. If blank or auto, the customer's [preferred_locales](#) or browser's locale is used.

Possible enum values

`auto``bg``cs``da``de``el``en`**on_behalf_of** optional CONNECT ONLY

The [on_behalf_of](#) account to use for this session. When specified, only subscriptions and invoices with this [on_behalf_of](#) account appear in the portal. For more information, see the [docs](#). Use the [Accounts API](#) to modify the [on_behalf_of](#) account's branding settings, which the portal displays.

Returns

Returns a portal session object.

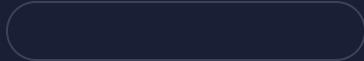
Create a portal configuration

Creates a configuration that describes the functionality and behavior of a PortalSession

Parameters

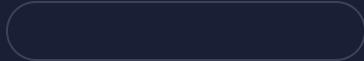
business_profile REQUIRED

The business information shown to customers in the portal.



features REQUIRED

Information about the features available in the portal.



default_return_url optional

The default URL to redirect customers to when they click on the portal's link to return to your website. This can be [overridden](#) when creating the session.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to

[metadata](#).

Returns

Returns a portal configuration object.

Update a portal configuration

Updates a configuration that describes the functionality of the customer portal.

Parameters

active optional

Whether the configuration is active and can be used to create portal sessions.

business_profile optional dictionary

The business information shown to customers in the portal.

**default_return_url** optional

The default URL to redirect customers to when they click on the portal's link to return to your website. This can be **overridden** when creating the session.

features optional dictionary

Information about the features available in the portal.

**metadata** optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to **metadata**.

Returns

Returns a portal configuration object.

Retrieve a portal configuration

Retrieves a configuration that describes the functionality of the customer portal.

Parameters

No parameters.

Returns

Returns a portal configuration object.

List portal configurations

Returns a list of configurations that describe the functionality of the customer portal.

Parameters

active optional

Only return configurations that are active or inactive (e.g., pass `true` to only list active configurations).

is_default optional

Only return the default or non-default configurations (e.g., pass `true` to only list the default configuration).

More parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

Returns a list of portal configuration objects.

Customer Tax IDs

You can add one or multiple tax IDs to a [customer](#). A customer's tax IDs are displayed on invoices and credit notes issued for the customer.

Related guide: [Customer Tax Identification Numbers](#).

The tax ID object

Attributes

id string

Unique identifier for the object.

country string

Two-letter ISO code representing the country of the tax ID.

customer string [EXPANDABLE](#)

ID of the customer.

type string

Type of the tax ID, one of `ae_trn`, `au_abn`, `au_arn`, `bg_uic`, `br_cnpj`, `br_cpf`, `ca_bn`, `ca_gst_hst`, `ca_pst_bc`, `ca_pst_mb`, `ca_pst_sk`, `ca_qst`, `ch_vat`, `cl_tin`, `es_cif`, `eu_vat`, `gb_vat`, `ge_vat`, `hk_br`, `hu_tin`, `id_npwp`, `il_vat`, `in_gst`, `is_vat`, `jp_cn`, `jp_rn`, `kr_brn`, `li_uid`, `mx_rfc`, `my_frp`, `my_itn`, `my_sst`, `no_vat`, `nz_gst`, `ru_inn`, `ru_kpp`, `sa_vat`, `sg_gst`, `sg_uen`, `si_tin`, `th_vat`, `tw_vat`, `ua_vat`, `us_ein`, or `za_vat`. Note that some legacy tax IDs have type `unknown`

value string

Value of the tax ID.

More attributes

✓ **object** string, value is "tax_id"

String representing the object's type. Objects of the same type share the same value.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **verification hash**

Tax ID verification information.

Create a tax ID

Creates a new `TaxID` object for a customer.

Parameters

type REQUIRED

Type of the tax ID, one of `ae_trn`, `au_abn`, `au_arn`, `bg_uic`, `br_cnpj`, `br_cpf`,
`ca_bn`, `ca_gst_hst`, `ca_pst_bc`, `ca_pst_mb`, `ca_pst_sk`, `ca_qst`, `ch_vat`, `cl_tin`,
`es_cif`, `eu_vat`, `gb_vat`, `ge_vat`, `hk_br`, `hu_tin`, `id_npwp`, `il_vat`, `in_gst`,
`is_vat`, `jp_cn`, `jp_rn`, `kr_brn`, `li_uid`, `mx_rfc`, `my_frp`, `my_itn`, `my_sst`,
`no_vat`, `nz_gst`, `ru_inn`, `ru_kpp`, `sa_vat`, `sg_gst`, `sg_uen`, `si_tin`, `th_vat`,
`tw_vat`, `ua_vat`, `us_ein`, or `za_vat`

value REQUIRED

Value of the tax ID.

Returns

The created `TaxID` object.

Retrieve a tax ID

Retrieves the `TaxID` object with the given identifier.

Parameters

No parameters.

Returns

Returns a `TaxID` object if a valid identifier was provided.

Delete a tax ID

Deletes an existing `TaxID` object.

Parameters

No parameters.

Returns

Returns an object with a deleted parameter on success. If the `TaxID` object does not exist, this call returns [an error](#).

List all tax IDs

Returns a list of tax IDs for a customer.

Parameters

✓ `ending_before` optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` tax IDs, starting after tax ID `starting_after`. Each entry in the array is a separate `TaxID` object. If no more tax IDs are available, the resulting array will be empty. Returns [an error](#) if the customer ID is invalid.

Invoices

Invoices are statements of amounts owed by a customer, and are either generated one-off, or generated periodically from a subscription.

They contain [invoice items](#), and proration adjustments that may be caused by subscription upgrades/downgrades (if necessary).

If your invoice is configured to be billed through automatic charges, Stripe automatically finalizes your invoice and attempts payment. Note that finalizing the invoice, [when automatic](#), does not happen immediately as the invoice is created. Stripe waits until one hour after the last webhook was successfully sent (or the last webhook timed out after failing). If you (and the platforms you may have connected to) have no webhooks configured, Stripe waits one hour after creation to finalize the invoice.

If your invoice is configured to be billed by sending an email, then based on your [email settings](#), Stripe will email the invoice to your customer and await payment. These emails can contain a link to a hosted page to pay the invoice.

Stripe applies any customer credit on the account before determining the amount due for the invoice (i.e., the amount that will be actually charged). If the amount due for the invoice is less than Stripe's [minimum allowed charge per currency](#), the invoice is automatically marked paid, and we add the amount due to the customer's credit balance which is applied to the next invoice.

More details on the customer's credit balance are [here](#).

Related guide: [Send Invoices to Customers](#).

The Invoice object

Attributes

id string

Unique identifier for the object.

auto_advance boolean

Controls whether Stripe will perform [automatic collection](#) of the invoice. When `false`, the invoice's state will not automatically advance without an explicit action.

charge string EXPANDABLE

ID of the latest charge generated for this invoice, if any.

collection_method string

Either `charge Automatically`, or `send_invoice`. When charging automatically, Stripe will attempt to pay this invoice using the default source attached to the customer. When sending an invoice, Stripe will email this invoice to the customer with payment instructions.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

customer string EXPANDABLE

The ID of the customer who will be billed.

description string

An arbitrary string attached to the object. Often useful for displaying to users. Referenced as 'memo' in the Dashboard.

hosted_invoice_url string

The URL for the hosted invoice page, which allows customers to view and pay an invoice. If the invoice has not been finalized yet, this will be null.

lines list

The individual line items that make up the invoice. `lines` is sorted as follows: invoice items in reverse chronological order, followed by the subscription, if any.

metadata hash

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

payment_intent string EXPANDABLE

The PaymentIntent associated with this invoice. The PaymentIntent is generated when the invoice is finalized, and can then be used to pay the invoice. Note that voiding an invoice will cancel the PaymentIntent.

period_end timestamp

End of the usage period during which invoice items were added to this invoice.

period_start timestamp

Start of the usage period during which invoice items were added to this invoice.

status string

The status of the invoice, one of `draft`, `open`, `paid`, `uncollectible`, or `void`. [Learn more](#)

subscription string EXPANDABLE

The subscription that this invoice was prepared for, if any.

total integer

Total after discounts and taxes.

More attributes

✓ **object** string, value is "invoice"

String representing the object's type. Objects of the same type share the same value.

✓ **account_country** string

The country of the business associated with this invoice, most often the business creating the invoice.

✓ **account_name** string

The public name of the business associated with this invoice, most often the business creating the invoice.

✓ **account_tax_ids** array containing strings [EXPANDABLE](#)

The account tax IDs associated with the invoice. Only editable when the invoice is a draft.

✓ **amount_due** integer

Final amount due at this time for this invoice. If the invoice's total is smaller than the minimum charge amount, for example, or if there is account credit that can be applied to the invoice, the `amount_due` may be 0. If there is a positive `starting_balance` for the invoice (the customer owes money), the `amount_due` will also take that into account. The charge that gets generated for the invoice will be for the amount specified in `amount_due`.

✓ **amount_paid** integer

The amount, in cents, that was paid.

✓ **amount_remaining** integer

The amount remaining, in cents, that is due.

✓ **application_fee_amount** integer [CONNECT ONLY](#)

The fee in cents that will be applied to the invoice and transferred to the application owner's Stripe account when the invoice is paid.

✓ **attempt_count** positive integer or zero

Number of payment attempts made for this invoice, from the perspective of the payment retry schedule. Any payment attempt counts as the first attempt, and subsequently only automatic retries increment the attempt count. In other words, manual payment attempts after the first attempt do not affect the retry schedule.

✓ **attempted** boolean

Whether an attempt has been made to pay the invoice. An invoice is not attempted until 1 hour after the `invoice.created` webhook, for example, so you might not want to display that invoice as unpaid to your users.

✓ **automatic_tax** hash

Settings and latest results for automatic tax lookup for this invoice.

✓ **billing_reason** string

Indicates the reason why the invoice was created. `subscription_cycle` indicates an invoice created by a subscription advancing into a new period. `subscription_create` indicates an invoice created due to creating a subscription. `subscription_update` indicates an invoice created due to updating a subscription. `subscription` is set for all old invoices

to indicate either a change to a subscription or a period advancement. `manual` is set for all invoices unrelated to a subscription (for example: created via the invoice editor). The `upcoming` value is reserved for simulated invoices per the upcoming invoice endpoint. `subscription_threshold` indicates an invoice created due to a billing threshold being reached.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **custom_fields** array of hashes

Custom fields displayed on the invoice.

✓ **customer_address** hash

The customer's address. Until the invoice is finalized, this field will equal `customer.address`. Once the invoice is finalized, this field will no longer be updated.

✓ **customer_email** string

The customer's email. Until the invoice is finalized, this field will equal `customer.email`. Once the invoice is finalized, this field will no longer be updated.

✓ **customer_name** string

The customer's name. Until the invoice is finalized, this field will equal `customer.name`. Once the invoice is finalized, this field will no longer be updated.

✓ **customer_phone** string

The customer's phone number. Until the invoice is finalized, this field will equal `customer.phone`. Once the invoice is finalized, this field will no longer be updated.

✓ **customer_shipping** hash

The customer's shipping information. Until the invoice is finalized, this field will equal `customer.shipping`. Once the invoice is finalized, this field will no longer be updated.

✓ **customer_tax_exempt** string

The customer's tax exempt status. Until the invoice is finalized, this field will equal `customer.tax_exempt`. Once the invoice is finalized, this field will no longer be updated.

✓ **customer_tax_ids** array of hashes

The customer's tax IDs. Until the invoice is finalized, this field will contain the same tax IDs as `customer.tax_ids`. Once the invoice is finalized, this field will no longer be updated.

✓ **default_payment_method** string EXPANDABLE

ID of the default payment method for the invoice. It must belong to the customer associated with the invoice. If not set, defaults to the subscription's default payment method, if any, or to the default payment method in the customer's invoice settings.

✓ **default_source** string EXPANDABLE

ID of the default payment source for the invoice. It must belong to the customer associated with the invoice and be in a chargeable state. If not set, defaults to the subscription's default source, if any, or to the customer's default source.

✓ **default_tax_rates** array of hashes

The tax rates applied to this invoice, if any.

✓ **discount** hash, discount object DEPRECATED

Describes the current discount applied to this invoice, if there is one. Not populated if there are multiple discounts.

✓ **discounts** array containing strings EXPANDABLE

The discounts applied to the invoice. Line item discounts are applied before invoice discounts. Use `expand[] = discounts` to expand each discount.

✓ **due_date** timestamp

The date on which payment for this invoice is due. This value will be `null` for invoices where `collection_method=charge Automatically`.

✓ **ending_balance** integer

Ending customer balance after the invoice is finalized. Invoices are finalized approximately an hour after successful webhook delivery or when payment collection is attempted for the invoice. If the invoice has not been finalized yet, this will be null.

✓ **footer** string

Footer displayed on the invoice.

✓ **invoice_pdf** string

The link to download the PDF for the invoice. If the invoice has not been finalized yet, this will be null.

✓ **last_finalization_error** hash

The error encountered during the previous attempt to finalize the invoice. This field is cleared when the invoice is successfully finalized.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **next_payment_attempt** timestamp

The time at which payment will next be attempted. This value will be `null` for invoices where `collection_method=send_invoice`.

✓ **number** string

A unique, identifying string that appears on emails sent to the customer for this invoice. This starts with the customer's unique `invoice_prefix` if it is specified.

✓ **on_behalf_of** string EXPANDABLE CONNECT ONLY

The account (if any) for which the funds of the invoice payment are intended. If set, the invoice will be presented with the branding and support information of the specified account. See the [Invoices with Connect](#) documentation for details.

✓ **paid** boolean

Whether payment was successfully collected for this invoice. An invoice can be paid (most commonly) with a charge or with credit from the customer's account balance.

✓ **paid_out_of_band** boolean

Returns true if the invoice was manually marked paid, returns false if the invoice hasn't been paid yet or was paid on Stripe.

✓ **payment_settings** hash

Configuration settings for the PaymentIntent that is generated when the invoice is finalized.

✓ **post_payment_credit_notes_amount** integer

Total amount of all post-payment credit notes issued for this invoice.

✓ **pre_payment_credit_notes_amount** integer

Total amount of all pre-payment credit notes issued for this invoice.

✓ **quote** string EXPANDABLE

The quote this invoice was generated from.

✓ **receipt_number** string

This is the transaction number that appears on email receipts sent for this invoice.

✓ **starting_balance** integer

Starting customer balance before the invoice is finalized. If the invoice has not been finalized yet, this will be the current customer balance.

✓ **statement_descriptor** string

Extra information about an invoice for the customer's credit card statement.

✓ **status_transitions** hash

The timestamps at which the invoice status was updated.

✓ **subscription_proration_date** integer

Only set for upcoming invoices that preview prorations. The time used to calculate prorations.

✓ **subtotal** integer

Total of all subscriptions, invoice items, and prorations on the invoice before any invoice level discount or tax is applied. Item discounts are already incorporated

✓ **tax** integer

The amount of tax on this invoice. This is the sum of all the tax amounts on this invoice.

✓ **test_clock** string EXPANDABLE

ID of the test clock this invoice belongs to.

✓ **threshold_reason** hash

If `billing_reason` is set to `subscription_threshold` this returns more information on which threshold rules triggered the invoice.

✓ **total_discount_amounts** array of hashes

The aggregate amounts calculated per discount across all line items.

✓ **total_tax_amounts** array of hashes

The aggregate amounts calculated per tax rate for all line items.

✓ **transfer_data** hash CONNECT ONLY

The account (if any) the payment will be attributed to for tax reporting, and where funds from the payment will be transferred to for the invoice.

✓ **webhooks_delivered_at** timestamp

Invoices are automatically paid or sent 1 hour after webhooks are delivered, or until all webhook delivery attempts have [been exhausted](#). This field tracks the time when webhooks for this invoice were successfully delivered. If the invoice had no webhooks to deliver, this will be set while the invoice is being created.

The (Invoice) Line Item object

Attributes

id string

Unique identifier for the object.

amount integer

The amount, in cents.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

description string

An arbitrary string attached to the object. Often useful for displaying to users.

metadata hash

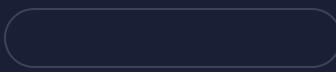
Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Note that for line items with `type=subscription` this will reflect the metadata of the subscription that caused the line item to be created.

period hash

The period this `line_item` covers. For subscription line items, this is the subscription period. For prorations, this starts when the proration was calculated, and ends at the period end of the subscription. For invoice items, this is the time at which the invoice item was created or the period of the item.

price hash

The price of the line item.



proration boolean

Whether this is a proration.

quantity integer

The quantity of the subscription, if the line item is a subscription or a proration.

type string

A string identifying the type of the source of this line item, either an `invoiceitem` or a `subscription`.

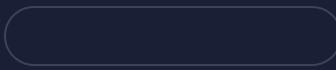
More attributes

✓ **object** string, value is "line_item"

String representing the object's type. Objects of the same type share the same value.

✓ **discount_amounts** array of hashes

The amount of discount calculated per discount for this line item.



✓ **discountable** boolean

If true, discounts will apply to this line item. Always false for prorations.

✓ **discounts** array containing strings EXPANDABLE

The discounts applied to the invoice line item. Line item discounts are applied before invoice discounts. Use `expand[] = discounts` to expand each discount.

✓ **invoice_item** string

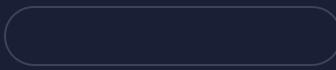
The ID of the `invoice item` associated with this line item if any.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **proration_details** hash

Additional details for proration line items



✓ **subscription** string

The subscription that the invoice item pertains to, if any.

✓ **subscription_item** string

The subscription item that generated this invoice item. Left empty if the line item is not an explicit result of a subscription.

✓ **tax_amounts** array of hashes

The amount of tax calculated per tax rate for this line item

✓ **tax_rates** array of hashes

The tax rates which apply to the line item.

Create an invoice

This endpoint creates a draft invoice for a given customer. The draft invoice created pulls in all pending invoice items on that customer, including prorations. The invoice remains a draft until you [finalize](#) the invoice, which allows you to [pay](#) or [send](#) the invoice to your customers.

Parameters

auto_advance optional

Controls whether Stripe will perform [automatic collection](#) of the invoice. When `false`, the invoice's state will not automatically advance without an explicit action.

collection_method optional

Either `charge Automatically`, or `send_invoice`. When charging automatically, Stripe will attempt to pay this invoice using the default source attached to the customer. When sending an invoice, Stripe will email this invoice to the customer with payment instructions. Defaults to `charge Automatically`.

customer optional

The ID of the customer who will be billed.

description optional

An arbitrary string attached to the object. Often useful for displaying to users. Referenced as 'memo' in the Dashboard.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

subscription optional

The ID of the subscription to invoice, if any. If not set, the created invoice will include all pending invoice items for the customer. If set, the created invoice will only include pending invoice items for that subscription and pending invoice items not associated with any subscription. The subscription's billing cycle and regular subscription events won't be affected.

More parameters

✓ **account_tax_ids** optional

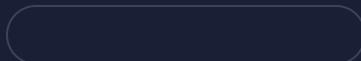
The account tax IDs associated with the invoice. Only editable when the invoice is a draft.

✓ **application_fee_amount** optional CONNECT ONLY

A fee in cents that will be applied to the invoice and transferred to the application owner's Stripe account. The request must be made with an OAuth key or the Stripe-Account header in order to take an application fee. For more information, see the [application fees documentation](#).

✓ **automatic_tax** optional dictionary

Settings for automatic tax lookup for this invoice.



✓ **custom_fields** optional array of hashes

A list of up to 4 custom fields to be displayed on the invoice.



✓ **days_until_due** optional

The number of days from when the invoice is created until it is due. Valid only for invoices where `collection_method=send_invoice`.

✓ **default_payment_method** optional

ID of the default payment method for the invoice. It must belong to the customer associated with the invoice. If not set, defaults to the subscription's default payment method, if any, or to the default payment method in the customer's invoice settings.

✓ **default_source** optional

ID of the default payment source for the invoice. It must belong to the customer associated with the invoice and be in a chargeable state. If not set, defaults to the subscription's default source, if any, or to the customer's default source.

✓ **default_tax_rates** optional

The tax rates that will apply to any line item that does not have `tax_rates` set.

✓ **discounts** optional array of hashes

The coupons to redeem into discounts for the invoice. If not specified, inherits the discount from the invoice's customer. Pass an empty string to avoid inheriting any discounts.

✓ **due_date** optional

The date on which payment for this invoice is due. Valid only for invoices where `collection_method=send_invoice`.

✓ **footer** optional

Footer to be displayed on the invoice.

✓ **on_behalf_of** optional CONNECT ONLY

The account (if any) for which the funds of the invoice payment are intended. If set, the invoice will be presented with the branding and support information of the specified account. See the [Invoices with Connect](#) documentation for details.

✓ **payment_settings** optional dictionary

Configuration settings for the PaymentIntent that is generated when the invoice is finalized.

✓ **pending_invoice_items_behavior** optional

How to handle pending invoice items on invoice creation. One of `include`, `exclude`, or `include_and_require`. `include` will include any pending invoice items, and will create an empty draft invoice if no pending invoice items exist. `include_and_require` will include any pending invoice items, if no pending invoice items exist then the request will fail. `exclude` will always create an empty invoice draft regardless if there are pending invoice items or not. Defaults to `include_and_require` if the parameter is omitted.

✓ **statement_descriptor** optional

Extra information about a charge for the customer's credit card statement. It must contain at least one letter. If not specified and this invoice is part of a subscription, the default `statement_descriptor` will be set to the first subscription item's product's `statement_descriptor`.

✓ **transfer_data** optional dictionary CONNECT ONLY

If specified, the funds from the invoice will be transferred to the destination and the ID of the resulting transfer will be found on the invoice's charge.

Returns

Returns the invoice object if there are pending invoice items to invoice. Returns [an error](#) if there are no pending invoice items or if the customer ID provided is invalid.

Retrieve an invoice

Retrieves the invoice with the given ID.

Parameters

No parameters.

Returns

Returns an invoice object if a valid invoice ID was provided. Returns [an error](#) otherwise.

The invoice object contains a `lines` hash that contains information about the subscriptions and invoice items that have been applied to the invoice, as well as any prorations that Stripe has automatically calculated. Each line on the invoice has an `amount` attribute that represents the amount actually contributed to the invoice's total. For invoice items and prorations, the amount attribute is the same as for the invoice item or proration respectively. For subscriptions, the amount may be different from the plan's regular price depending on whether the invoice covers a trial period or the invoice period differs from the plan's usual interval.

The invoice object has both a `subtotal` and a `total`. The subtotal represents the total before any discounts, while the total is the final amount to be charged to the customer after all coupons have been applied.

The invoice also has a `next_payment_attempt` attribute that tells you the next time (as a Unix timestamp) payment for the invoice will be automatically attempted. For invoices with manual payment collection, that have been closed, or that have reached the maximum

number of retries (specified in your [subscriptions settings](#)), the `next_payment_attempt` will be null.

Update an invoice

Draft invoices are fully editable. Once an invoice is [finalized](#), monetary values, as well as `collection_method`, become uneditable.

If you would like to stop the Stripe Billing engine from automatically finalizing, reattempting payments on, sending reminders for, or [automatically reconciling](#) invoices, pass `auto_advance=false`.

Parameters

auto_advance optional

Controls whether Stripe will perform [automatic collection](#) of the invoice.

collection_method optional

Either `charge Automatically` or `send_invoice`. This field can be updated only on `draft` invoices.

description optional

An arbitrary string attached to the object. Often useful for displaying to users. Referenced as 'memo' in the Dashboard.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ **account_tax_ids** optional

The account tax IDs associated with the invoice. Only editable when the invoice is a draft.

✓ **application_fee_amount** optional CONNECT ONLY

A fee in cents that will be applied to the invoice and transferred to the application owner's Stripe account. The request must be made with an OAuth key or the Stripe-Account header in order to take an application fee. For more information, see the [application fees documentation](#).

✓ **automatic_tax** optional dictionary

Settings for automatic tax lookup for this invoice.

✓ **custom_fields** optional array of hashes

A list of up to 4 custom fields to be displayed on the invoice. If a value for `custom_fields` is specified, the list specified will replace the existing custom field list on this invoice. Pass an empty string to remove previously-defined fields.

✓ **days_until_due** optional

The number of days from which the invoice is created until it is due. Only valid for invoices where `collection_method=send_invoice`. This field can only be updated on `draft` invoices.

✓ **default_payment_method** optional

ID of the default payment method for the invoice. It must belong to the customer associated with the invoice. If not set, defaults to the subscription's default payment method, if any, or to the default payment method in the customer's invoice settings.

✓ **default_source** optional

ID of the default payment source for the invoice. It must belong to the customer associated with the invoice and be in a chargeable state. If not set, defaults to the subscription's default source, if any, or to the customer's default source.

✓ **default_tax_rates** optional

The tax rates that will apply to any line item that does not have `tax_rates` set. Pass an empty string to remove previously-defined tax rates.

✓ **discounts** optional array of hashes

The discounts that will apply to the invoice. Pass an empty string to remove previously-defined discounts.

✓ **due_date** optional

The date on which payment for this invoice is due. Only valid for invoices where `collection_method=send_invoice`. This field can only be updated on `draft` invoices.

✓ **footer** optional

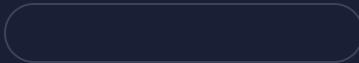
Footer to be displayed on the invoice.

✓ **on_behalf_of** optional CONNECT ONLY

The account (if any) for which the funds of the invoice payment are intended. If set, the invoice will be presented with the branding and support information of the specified account. See the [Invoices with Connect](#) documentation for details.

✓ **payment_settings** optional dictionary

Configuration settings for the PaymentIntent that is generated when the invoice is finalized.

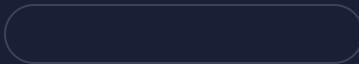


✓ **statement_descriptor** optional

Extra information about a charge for the customer's credit card statement. It must contain at least one letter. If not specified and this invoice is part of a subscription, the default `statement_descriptor` will be set to the first subscription item's product's `statement_descriptor`.

✓ **transfer_data** optional dictionary CONNECT ONLY

If specified, the funds from the invoice will be transferred to the destination and the ID of the resulting transfer will be found on the invoice's charge. This will be unset if you POST an empty value.



Returns

Returns the invoice object.

Delete a draft invoice

Permanently deletes a one-off invoice draft. This cannot be undone. Attempts to delete invoices that are no longer in a draft state will fail; once an invoice has been finalized or if an invoice is for a subscription, it must be [voided](#).

Parameters

No parameters.

Returns

A successfully deleted invoice. Otherwise, this call returns [an error](#), such as if the invoice has already been deleted.

Finalize an invoice

Stripe automatically finalizes drafts before sending and attempting payment on invoices. However, if you'd like to finalize a draft invoice manually, you can do so using this method.

Parameters

auto_advance optional

Controls whether Stripe will perform [automatic collection](#) of the invoice. When `false`, the invoice's state will not automatically advance without an explicit action.

Returns

Returns an invoice object with `status=open`.

Pay an invoice

Stripe automatically creates and then attempts to collect payment on invoices for customers on subscriptions according to your [subscriptions settings](#). However, if you'd like to attempt payment on an invoice out of the normal collection schedule or for some other reason, you can do so.

Parameters

✓ **forgive** optional

In cases where the source used to pay the invoice has insufficient funds, passing `forgive=true` controls whether a charge should be attempted for the full amount available on the source, up to the amount to fully pay the invoice. This effectively forgives the difference between the amount available on the source and the amount due.

Passing `forgive=false` will fail the charge if the source hasn't been pre-funded with the right amount. An example for this case is with ACH Credit Transfers and wires: if the amount wired is less than the amount due by a small amount, you might want to forgive the difference. Defaults to `false`.

✓ **off_session** optional

Indicates if a customer is on or off-session while an invoice payment is attempted. Defaults to `true` (off-session).

✓ **paid_out_of_band** optional

Boolean representing whether an invoice is paid outside of Stripe. This will result in no charge being made. Defaults to `false`.

✓ **payment_method** optional

A PaymentMethod to be charged. The PaymentMethod must be the ID of a PaymentMethod belonging to the customer associated with the invoice being paid.

✓ **source** optional

A payment source to be charged. The source must be the ID of a source belonging to the customer associated with the invoice being paid.

Returns

Returns the invoice object.

Send an invoice for manual payment

Stripe will automatically send invoices to customers according to your [subscriptions settings](#). However, if you'd like to manually send an invoice to your customer out of the normal schedule, you can do so. When sending invoices that have already been paid, there will be no reference to the payment in the email.

Requests made in test-mode result in no emails being sent, despite sending an `invoice.sent` event.

Parameters

No parameters.

Returns

Returns the invoice object.

Void an invoice

Mark a finalized invoice as void. This cannot be undone. Voiding an invoice is similar to [deletion](#), however it only applies to finalized invoices and maintains a papertrail where the invoice can still be found.

Parameters

No parameters.

Returns

Returns the voided invoice object.

Mark an invoice as uncollectible

Marking an invoice as uncollectible is useful for keeping track of bad debts that can be written off for accounting purposes.

Parameters

No parameters.

Returns

Returns the invoice object.

Retrieve an invoice's line items

When retrieving an invoice, you'll get a **lines** property containing the total count of line items and the first handful of those items. There is also a URL where you can retrieve the full (paginated) list of line items.

Parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

Returns a list of [line_item objects](#).

Retrieve an upcoming invoice

At any time, you can preview the upcoming invoice for a customer. This will show you all the charges that are pending, including subscription renewal charges, invoice item charges, etc. It will also show you any discounts that are applicable to the invoice.

Note that when you are viewing an upcoming invoice, you are simply viewing a preview – the invoice has not yet been created. As such, the upcoming invoice will not show up in invoice listing calls, and you cannot use the API to pay or edit the invoice. If you want to change the amount that your customer will be billed, you can add, remove, or update pending invoice items, or update the customer's discount.

You can preview the effects of updating a subscription, including a preview of what proration will take place. To ensure that the actual proration is calculated exactly the same as the previewed proration, you should pass a `proration_date` parameter when doing the actual subscription update. The value passed in should be the same as the `subscription_proration_date` returned on the upcoming invoice resource. The recommended way to get only the prorations being previewed is to consider only proration line items where `period[start]` is equal to the `subscription_proration_date` on the upcoming invoice resource.

Parameters

customer REQUIRED IF SUBSCRIPTION UNSET

The identifier of the customer whose upcoming invoice you'd like to retrieve.

subscription optional

The identifier of the subscription for which you'd like to retrieve the upcoming invoice. If not provided, but a `subscription_items` is provided, you will preview creating a subscription with those items. If neither `subscription` nor `subscription_items` is provided, you will retrieve the next upcoming invoice from among the customer's subscriptions.

More parameters

> **customer_details** REQUIRED IF SUBSCRIPTION AND CUSTOMER UNSET

> **automatic_tax** optional dictionary

> **coupon** optional

> **discounts** optional array of hashes

> **invoice_items** optional array of hashes

> **schedule** optional

> **subscription_billing_cycle_anchor** optional

> **subscription_cancel_at** optional

> **subscription_cancel_at_period_end** optional

> **subscription_cancel_now** optional

> **subscription_default_tax_rates** optional

> **subscription_items** optional array of hashes

> **subscription_proration_behavior** optional

> **subscription_proration_date** optional

> **subscription_start_date** optional

> **subscription_trial_end** optional

› **subscription_trial_from_plan** optional

Returns

Returns an invoice if valid customer information is provided. Returns [an error](#) otherwise.

Retrieve an upcoming invoice's line items

When retrieving an upcoming invoice, you'll get a **lines** property containing the total count of line items and the first handful of those items. There is also a URL where you can retrieve the full (paginated) list of line items.

Parameters

customer REQUIRED IF SUBSCRIPTION UNSET

The identifier of the customer whose upcoming invoice you'd like to retrieve.

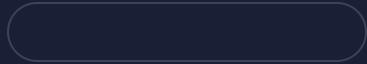
subscription optional

The identifier of the subscription for which you'd like to retrieve the upcoming invoice. If not provided, but a `subscription_items` is provided, you will preview creating a subscription with those items. If neither `subscription` nor `subscription_items` is provided, you will retrieve the next upcoming invoice from among the customer's subscriptions.

More parameters

› **customer_details** REQUIRED IF SUBSCRIPTION AND CUSTOMER UNSET

Details about the customer you want to invoice or overrides for an existing customer.



› **automatic_tax** optional dictionary

Settings for automatic tax lookup for this invoice preview.



› **coupon** optional

The code of the coupon to apply. If `subscription` or `subscription_items` is provided, the invoice returned will preview updating or creating a subscription with that coupon.

Otherwise, it will preview applying that coupon to the customer for the next upcoming invoice from among the customer's subscriptions. The invoice can be previewed without a coupon by passing this value as an empty string.

✓ **discounts** optional array of hashes

The coupons to redeem into discounts for the invoice preview. If not specified, inherits the discount from the customer or subscription. This only works for coupons directly applied to the invoice. To apply a coupon to a subscription, you must use the `coupon` parameter instead. Pass an empty string to avoid inheriting any discounts. To preview the upcoming invoice for a subscription that hasn't been created, use `coupon` instead.

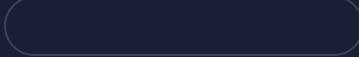


✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **invoice_items** optional array of hashes

List of invoice items to add or update in the upcoming invoice preview.



✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **schedule** optional

The identifier of the unstarted schedule whose upcoming invoice you'd like to retrieve.

Cannot be used with subscription or subscription fields.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

✓ **subscription_billing_cycle_anchor** optional

For new subscriptions, a future timestamp to anchor the subscription's [billing cycle](#). This is used to determine the date of the first full invoice, and, for plans with `month` or `year` intervals, the day of the month for subsequent invoices. For existing subscriptions, the value can only be set to `now` or `unchanged`.

✓ **subscription_cancel_at** optional

Timestamp indicating when the subscription should be scheduled to cancel. Will prorate if within the current period and prorations have been enabled using `proration_behavior`.

✓ **subscription_cancel_at_period_end** optional

Boolean indicating whether this subscription should cancel at the end of the current period.

✓ **subscription_cancel_now** optional

This simulates the subscription being canceled or expired immediately.

✓ **subscription_default_tax_rates** optional

If provided, the invoice returned will preview updating or creating a subscription with these default tax rates. The default tax rates will apply to any line item that does not have `tax_rates` set.

✓ **subscription_items** optional array of hashes

A list of up to 20 subscription items, each with an attached price.



✓ **subscription_proration_behavior** optional

Determines how to handle `prorations` when the billing cycle changes (e.g., when switching plans, resetting `billing_cycle_anchor=now`, or starting a trial), or if an item's `quantity` changes. Valid values are `create_prorations`, `none`, or `always_invoice`.

Passing `create_prorations` will cause proration invoice items to be created when applicable. These proration items will only be invoiced immediately under [certain conditions](#). In order to always invoice immediately for prorations, pass `always_invoice`.

Prorations can be disabled by passing `none`.

✓ **subscription_proration_date** optional

If previewing an update to a subscription, and doing proration, `subscription_proration_date` forces the proration to be calculated as though the update was done at the specified time. The time given must be within the current subscription period, and cannot be before the subscription was on its current plan. If set, `subscription`, and one of `subscription_items`, or `subscription_trial_end` are required. Also, `subscription_proration_behavior` cannot be set to 'none'.

✓ **subscription_start_date** optional

Date a subscription is intended to start (can be future or past)

✓ **subscription_trial_end** optional

If provided, the invoice returned will preview updating or creating a subscription with that trial end. If set, one of `subscription_items` or `subscription` is required.

✓ **subscription_trial_from_plan** optional

Indicates if a plan's `trial_period_days` should be applied to the subscription. Setting `subscription_trial_end` per subscription is preferred, and this defaults to `false`. Setting this flag to `true` together with `subscription_trial_end` is not allowed. See [Using trial periods on subscriptions](#) to learn more.

Returns

Returns a list of `line_item` objects.

List all invoices

You can list all invoices, or list the invoices for a specific customer. The invoices are returned sorted by creation date, with the most recently created invoices appearing first.

Parameters

customer optional

Only return invoices for the customer specified by this customer ID.

status optional

The status of the invoice, one of `draft`, `open`, `paid`, `uncollectible`, or `void`. [Learn more](#)

subscription optional

Only return invoices for the subscription specified by this subscription ID.

More parameters

✓ **collection_method** optional

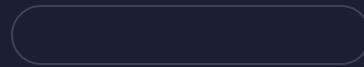
The collection method of the invoice to retrieve. Either `charge Automatically` or `send_invoice`.

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

✓ **due_date** optional dictionary

A filter on the list based on the object `due_date` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` invoices, starting after invoice `starting_after`. Each entry in the array is a separate invoice object. If no more invoices are available, the resulting array will be empty. Returns [an error](#) if the customer ID is invalid.

Search invoices

Search for invoices you've previously created using Stripe's [Search Query Language](#). Don't use search in read-after-write flows where strict consistency is necessary. Under normal operating conditions, data is searchable in less than a minute. Occasionally, propagation of new or updated data can be up to an hour behind during outages. Search functionality is not available to merchants in India.

Parameters

query REQUIRED

The search query string. See [search query language](#) and the list of supported [query fields for invoices](#).

limit optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

page optional

A cursor for pagination across multiple pages of results. Don't include this parameter on the first call. Use the next_page value returned in a previous response to request subsequent results.

Returns

A dictionary with a `data` property that contains an array of up to `limit` invoices. If no objects match the query, the resulting array will be empty. See the related guide on [expanding properties in lists](#).

Invoice Items

Sometimes you want to add a charge or credit to a customer, but actually charge or credit the customer's card only at the end of a regular billing cycle. This is useful for combining several charges (to minimize per-transaction fees), or for having Stripe tabulate your usage-based billing totals.

Related guide: [Subscription Invoices](#).

The invoiceitem object

Attributes

id string

Unique identifier for the object.

amount integer

Amount (in the `currency` specified) of the invoice item. This should always be equal to `unit_amount * quantity`.

currency currency

Three-letter ISO currency code, in lowercase. Must be a supported currency.

customer string EXPANDABLE

The ID of the customer who will be billed when this invoice item is billed.

description string

An arbitrary string attached to the object. Often useful for displaying to users.

metadata hash

Set of key-value pairs that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

period hash

The period associated with this invoice item. When set to different values, the period will be rendered on the invoice.

Show child attributes

price hash

The price of the invoice item.

Show child attributes

proration boolean

Whether the invoice item was created automatically as a proration adjustment when the customer switched plans.

More attributes

[Collapse all](#)

object string, value is "invoiceitem"

String representing the object's type. Objects of the same type share the same value.

date timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

discountable boolean

If true, discounts will apply to this invoice item. Always false for prorations.

discounts array containing strings EXPANDABLE

The discounts which apply to the invoice item. Item discounts are applied before invoice discounts. Use `expand[] = discounts` to expand each discount.

invoice string EXPANDABLE

The ID of the invoice this invoice item belongs to.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **quantity** integer

Quantity of units for the invoice item. If the invoice item is a proration, the quantity of the subscription that the proration was computed for.

✓ **subscription** string [EXPANDABLE](#)

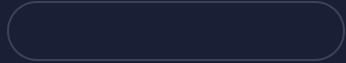
The subscription that this invoice item has been created for, if any.

✓ **subscription_item** string

The subscription item that this invoice item has been created for, if any.

✓ **tax_rates** array of hashes

The tax rates which apply to the invoice item. When set, the `default_tax_rates` on the invoice do not apply to this invoice item.



✓ **test_clock** string [EXPANDABLE](#)

ID of the test clock this invoice item belongs to.

✓ **unit_amount** integer

Unit amount (in the `currency` specified) of the invoice item.

✓ **unit_amount_decimal** decimal string

Same as `unit_amount`, but contains a decimal value with at most 12 decimal places.

Create an invoice item

Creates an item to be added to a draft invoice (up to 250 items per invoice). If no invoice is specified, the item will be on the next invoice created for the customer specified.

Parameters

customer REQUIRED

The ID of the customer who will be billed when this invoice item is billed.

amount optional

The integer amount in cents of the charge to be applied to the upcoming invoice. Passing in a negative `amount` will reduce the `amount_due` on the invoice.

currency optional

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

description optional

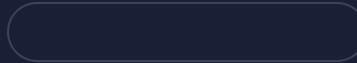
An arbitrary string which you can attach to the invoice item. The description is displayed in the invoice for easy tracking.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

period optional dictionary

The period associated with this invoice item. When set to different values, the period will be rendered on the invoice.



price optional

The ID of the price object.

More parameters

[Collapse all](#)

✓ **discountable** optional

Controls whether discounts apply to this invoice item. Defaults to false for prorations or negative invoice items, and true for all other invoice items.

✓ **discounts** optional array of hashes

The coupons to redeem into discounts for the invoice item or invoice line item.



✓ **invoice** optional

The ID of an existing invoice to add this invoice item to. When left blank, the invoice item will be added to the next upcoming scheduled invoice. This is useful when adding invoice items in response to an `invoice.created` webhook. You can only add invoice items to draft invoices and there is a maximum of 250 items per invoice.

✓ **price_data** optional dictionary

Data used to generate a new **Price** object inline.

✓ **quantity** optional

Non-negative integer. The quantity of units for the invoice item.

✓ **subscription** optional

The ID of a subscription to add this invoice item to. When left blank, the invoice item will be added to the next upcoming scheduled invoice. When set, scheduled invoices for subscriptions other than the specified subscription will ignore the invoice item. Use this when you want to express that an invoice item has been accrued within the context of a particular subscription.

✓ **tax_rates** optional

The tax rates which apply to the invoice item. When set, the `default_tax_rates` on the invoice do not apply to this invoice item.

✓ **unit_amount** optional

The integer unit amount in cents of the charge to be applied to the upcoming invoice. This `unit_amount` will be multiplied by the quantity to get the full amount. Passing in a negative `unit_amount` will reduce the `amount_due` on the invoice.

✓ **unit_amount_decimal** optional

Same as `unit_amount`, but accepts a decimal value in cents with at most 12 decimal places. Only one of `unit_amount` and `unit_amount_decimal` can be set.

Returns

The created invoice item object is returned if successful. Otherwise, this call returns an error.

Retrieve an invoice item

Retrieves the invoice item with the given ID.

Parameters

Returns

Returns an invoice item if a valid invoice item ID was provided. Returns [an error](#) otherwise.

Update an invoice item

Updates the amount or description of an invoice item on an upcoming invoice. Updating an invoice item is only possible before the invoice it's attached to is closed.

Parameters

amount optional

The integer amount in cents of the charge to be applied to the upcoming invoice. If you want to apply a credit to the customer's account, pass a negative amount.

description optional

An arbitrary string which you can attach to the invoice item. The description is displayed in the invoice for easy tracking.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

period optional dictionary

The period associated with this invoice item. When set to different values, the period will be rendered on the invoice.



price optional

The ID of the price object.

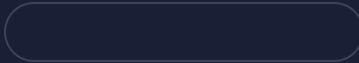
More parameters

✓ **discountable** optional

Controls whether discounts apply to this invoice item. Defaults to false for prorations or negative invoice items, and true for all other invoice items. Cannot be set to true for prorations.

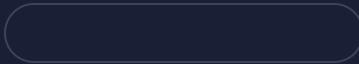
✓ **discounts** optional array of hashes

The coupons & existing discounts which apply to the invoice item or invoice line item. Item discounts are applied before invoice discounts. Pass an empty string to remove previously-defined discounts.



✓ **price_data** optional dictionary

Data used to generate a new [Price](#) object inline.



✓ **quantity** optional

Non-negative integer. The quantity of units for the invoice item.

✓ **tax_rates** optional

The tax rates which apply to the invoice item. When set, the `default_tax_rates` on the invoice do not apply to this invoice item. Pass an empty string to remove previously-defined tax rates.

✓ **unit_amount** optional

The integer unit amount in cents of the charge to be applied to the upcoming invoice. This unit_amount will be multiplied by the quantity to get the full amount. If you want to apply a credit to the customer's account, pass a negative unit_amount.

✓ **unit_amount_decimal** optional

Same as `unit_amount`, but accepts a decimal value in cents with at most 12 decimal places. Only one of `unit_amount` and `unit_amount_decimal` can be set.

Returns

The updated invoice item object is returned upon success. Otherwise, this call returns an error.

Delete an invoice item

Deletes an invoice item, removing it from an invoice. Deleting invoice items is only possible when they're not attached to invoices, or if it's attached to a draft invoice.

Parameters

No parameters.

Returns

An object with the deleted invoice item's ID and a deleted flag upon success. Otherwise, this call returns an error, such as if the invoice item has already been deleted.

List all invoice items

Returns a list of your invoice items. Invoice items are returned sorted by creation date, with the most recently created invoice items appearing first.

Parameters

customer optional

The identifier of the customer whose invoice items to return. If none is provided, all invoice items will be returned.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **invoice** optional

Only return invoice items belonging to this invoice. If none is provided, all invoice items will be returned. If specifying an invoice, no customer identifier is needed.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **pending** optional

Set to `true` to only show pending invoice items, which are not yet attached to any invoices. Set to `false` to only show invoice items already attached to invoices. If unspecified, no filter is applied.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` invoice items, starting after invoice item `starting_after`. Each entry in the array is a separate invoice item object. If no more invoice items are available, the resulting array will be empty. This request should never return an error.

Plans

You can now model subscriptions more flexibly using the [Prices API](#). It replaces the Plans API and is backwards compatible to simplify your migration.

Plans define the base price, currency, and billing cycle for recurring purchases of products. [Products](#) help you track inventory or provisioning, and plans help you track pricing. Different physical goods or levels of service should be represented by products, and pricing options should be represented by plans. This approach lets you change prices without having to change your provisioning scheme.

For example, you might have a single "gold" product that has plans for \$10/month, \$100/year, €9/month, and €90/year.

The plan object

Attributes

id string

Unique identifier for the object.

active boolean

Whether the plan can be used for new purchases.

amount positive integer or zero

The unit amount in cents to be charged, represented as a whole integer if possible. Only set if `billing_scheme=per_unit`.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

interval enum

The frequency at which a subscription is billed. One of `day`, `week`, `month` or `year`.

Possible enum values
<code>month</code>
<code>year</code>
<code>week</code>
<code>day</code>

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

nickname string

A brief description of the plan, hidden from customers.

product string [EXPANDABLE](#)

The product whose pricing this plan determines.

More attributes

✓ **object** string, value is "plan"

String representing the object's type. Objects of the same type share the same value.

✓ **aggregate_usage** string

Specifies a usage aggregation strategy for plans of `usage_type=metered`. Allowed values are `sum` for summing up all usage during a period, `last_during_period` for using the last usage record reported within a period, `last_ever` for using the last usage record ever (across period bounds) or `max` which uses the usage record with the maximum reported usage during a period. Defaults to `sum`.

✓ **amount_decimal** decimal string

The unit amount in cents to be charged, represented as a decimal string with at most 12 decimal places. Only set if `billing_scheme=per_unit`.

✓ **billing_scheme** string

Describes how to compute the price per period. Either `per_unit` or `tiered`. `per_unit` indicates that the fixed amount (specified in `amount`) will be charged per unit in `quantity` (for plans with `usage_type=licensed`), or per unit of total usage (for plans with `usage_type=metered`). `tiered` indicates that the unit pricing will be computed using a tiering strategy as defined using the `tiers` and `tiers_mode` attributes.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **interval_count** positive integer

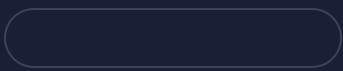
The number of intervals (specified in the `interval` attribute) between subscription billings. For example, `interval=month` and `interval_count=3` bills every 3 months.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **tiers** array of hashes EXPANDABLE

Each element represents a pricing tier. This parameter requires `billing_scheme` to be set to `tiered`. See also the documentation for `billing_scheme`. This field is not included by default. To include it in the response, [expand](#) the `tiers` field.



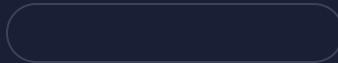
✓ **tiers_mode** string

Defines if the tiering price should be `graduated` or `volume` based. In `volume`-based tiering, the maximum quantity within a period determines the per unit price. In `graduated`

tiering, pricing can change as the quantity grows.

✓ **transform_usage** hash

Apply a transformation to the reported usage or set quantity before computing the amount billed. Cannot be combined with `tiers`.



✓ **trial_period_days** positive integer

Default number of trial days when subscribing a customer to this plan using `trial_from_plan=true`.

✓ **usage_type** enum

Configures how the quantity per period should be determined. Can be either `metered` or `licensed`. `licensed` automatically bills the `quantity` set when adding it to a subscription. `metered` aggregates the total usage based on usage records. Defaults to `licensed`.

Possible enum values

`metered`

`licensed`

Create a plan

You can now model subscriptions more flexibly using the [Prices API](#). It replaces the Plans API and is backwards compatible to simplify your migration.

Parameters

amount REQUIRED UNLESS BILLING_SCHEME=TIERED

A positive integer in cents (or 0 for a free plan) representing how much to charge on a recurring basis.

currency REQUIRED

Three-letter ISO currency code, in lowercase. Must be a [supported currency](#).

interval REQUIRED

Specifies billing frequency. Either `day`, `week`, `month` or `year`.

product REQUIRED

The product whose pricing the created plan will represent. This can either be the ID of an existing product, or a dictionary containing fields used to create a [service product](#).

active optional

Whether the plan is currently available for new subscriptions. Defaults to `true`.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

nickname optional

A brief description of the plan, hidden from customers.

More parameters

✓ **id** optional

An identifier randomly generated by Stripe. Used to identify this plan when subscribing a customer. You can optionally override this ID, but the ID must be unique across all plans in your Stripe account. You can, however, use the same plan ID in both live and test modes.

✓ **tiers** REQUIRED IF BILLING_SCHEME=TIERED

Each element represents a pricing tier. This parameter requires `billing_scheme` to be set to `tiered`. See also the documentation for `billing_scheme`.

✓ **tiers_mode** REQUIRED IF BILLING_SCHEME=TIERED

Defines if the tiering price should be `graduated` or `volume` based. In `volume`-based tiering, the maximum quantity within a period determines the per unit price, in `graduated` tiering pricing can successively change as the quantity grows.

Possible enum values

`volume`

`graduated`

✓ **aggregate_usage** optional

Specifies a usage aggregation strategy for plans of `usage_type=metered`. Allowed values are `sum` for summing up all usage during a period, `last_during_period` for using the last usage record reported within a period, `last_ever` for using the last usage record ever (across period bounds) or `max` which uses the usage record with the maximum reported usage during a period. Defaults to `sum`.

✓ **amount_decimal** optional

Same as `amount`, but accepts a decimal value with at most 12 decimal places. Only one of `amount` and `amount_decimal` can be set.

✓ **billing_scheme** optional

Describes how to compute the price per period. Either `per_unit` or `tiered`. `per_unit` indicates that the fixed amount (specified in `amount`) will be charged per unit in `quantity` (for plans with `usage_type=licensed`), or per unit of total usage (for plans with `usage_type=metered`). `tiered` indicates that the unit pricing will be computed using a tiering strategy as defined using the `tiers` and `tiers_mode` attributes.

✓ **interval_count** optional

The number of intervals between subscription billings. For example, `interval=month` and `interval_count=3` bills every 3 months. Maximum of one year interval allowed (1 year, 12 months, or 52 weeks).

✓ **transform_usage** optional dictionary

Apply a transformation to the reported usage or set quantity before computing the billed price. Cannot be combined with `tiers`.

✓ **trial_period_days** optional

Default number of trial days when subscribing a customer to this plan using `trial_from_plan=true`.

✓ **usage_type** optional

Configures how the quantity per period should be determined. Can be either `metered` or `licensed`. `licensed` automatically bills the `quantity` set when adding it to a subscription. `metered` aggregates the total usage based on usage records. Defaults to `licensed`.

Returns

Returns the plan object.

Retrieve a plan

Retrieves the plan with the given ID.

Parameters

No parameters.

Returns

Returns a plan if a valid plan ID was provided. Returns [an error](#) otherwise.

Update a plan

Updates the specified plan by setting the values of the parameters passed. Any parameters not provided are left unchanged. By design, you cannot change a plan's ID, amount, currency, or billing cycle.

Parameters

active optional

Whether the plan is currently available for new subscriptions.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

nickname optional

A brief description of the plan, hidden from customers.

More parameters

✗ **product** optional

The product the plan belongs to. This cannot be changed once it has been used in a subscription or subscription schedule.

✓ **trial_period_days** optional

Default number of trial days when subscribing a customer to this plan using `trial_from_plan=true`.

Returns

The updated plan object is returned upon success. Otherwise, this call returns [an error](#).

Delete a plan

Deleting plans means new subscribers can't be added. Existing subscribers aren't affected.

Parameters

No parameters.

Returns

An object with the deleted plan's ID and a deleted flag upon success. Otherwise, this call returns [an error](#), such as if the plan has already been deleted.

List all plans

Returns a list of your plans.

Parameters

active optional

Only return plans that are active or inactive (e.g., pass `false` to list all inactive plans).

product optional

Only return plans for the given product.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` plans, starting after plan `starting_after`. Each entry in the array is a separate plan object. If no more plans are available, the resulting array will be empty. This request should never return an error.

Quote

A Quote is a way to model prices that you'd like to provide to a customer. Once accepted, it will automatically create an invoice, subscription or subscription schedule.

The quote object

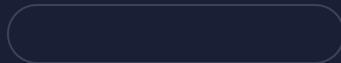
Attributes

id string

Unique identifier for the object.

line_items list EXPANDABLE

A list of items the customer is being quoted for. This field is not included by default. To include it in the response, [expand](#) the `line_items` field.



metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

More attributes

✓ **object** string, value is "quote"

String representing the object's type. Objects of the same type share the same value.

✓ **amount_subtotal** integer

Total before any discounts or taxes are applied.

✓ **amount_total** integer

Total after discounts and taxes are applied.

✓ **application_fee_amount** integer CONNECT ONLY

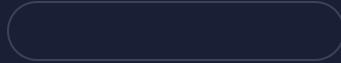
The amount of the application fee (if any) that will be requested to be applied to the payment and transferred to the application owner's Stripe account. Only applicable if there are no line items with recurring prices on the quote.

✓ **application_fee_percent** decimal CONNECT ONLY

A non-negative decimal between 0 and 100, with at most two decimal places. This represents the percentage of the subscription invoice subtotal that will be transferred to the application owner's Stripe account. Only applicable if there are line items with recurring prices on the quote.

✓ **automatic_tax** hash

Settings for automatic tax lookup for this quote and resulting invoices and subscriptions.



✓ **collection_method** string

Either `charge Automatically`, or `send invoice`. When charging automatically, Stripe will attempt to pay invoices at the end of the subscription cycle or on finalization using the default payment method attached to the subscription or customer. When sending an invoice, Stripe will email your customer an invoice with payment instructions. Defaults to `charge Automatically`.

✓ **computed** hash

The definitive totals and line items for the quote, computed based on your inputted line items as well as other configuration such as trials. Used for rendering the quote to your customer.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **currency** string

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

✓ **customer** string [EXPANDABLE](#)

The customer which this quote belongs to. A customer is required before finalizing the quote. Once specified, it cannot be changed.

✓ **default_tax_rates** array containing strings [EXPANDABLE](#)

The tax rates applied to this quote.

✓ **description** string

A description that will be displayed on the quote PDF.

✓ **discounts** array containing strings [EXPANDABLE](#)

The discounts applied to this quote.

✓ **expires_at** timestamp

The date on which the quote will be canceled if in `open` or `draft` status. Measured in seconds since the Unix epoch.

✓ **footer** string

A footer that will be displayed on the quote PDF.

✓ **from_quote** hash

Details of the quote that was cloned. See the [cloning documentation](#) for more details.



✓ **header** string

✓ **invoice** string EXPANDABLE

The invoice that was created from this quote.

✓ **invoice_settings** hash

All invoices will be billed using the specified settings.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **number** string

A unique number that identifies this particular quote. This number is assigned once the quote is **finalized**.

✓ **on_behalf_of** string EXPANDABLE CONNECT ONLY

The account on behalf of which to charge. See the [Connect documentation](#) for details.

✓ **status** enum

The status of the quote.

Possible enum values
<code>draft</code>
The quote can be edited while in this status and has not been sent to the customer.
<code>open</code>
The quote has been finalized and is awaiting action from the customer.
<code>accepted</code>
The customer has accepted the quote and invoice, subscription or subscription schedule has been created.
<code>canceled</code>
The quote has been canceled and is no longer valid.

✓ **status_transitions** hash

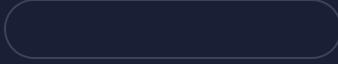
The timestamps of which the quote transitioned to a new status.

✓ **subscription** string EXPANDABLE

The subscription that was created or updated from this quote.

✓ **subscription_data** hash

When creating a subscription or subscription schedule, the specified configuration data will be used. There must be at least one line item with a recurring price for a subscription or subscription schedule to be created.



✓ **subscription_schedule** string [EXPANDABLE](#)

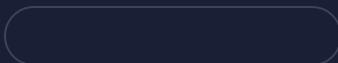
The subscription schedule that was created or updated from this quote.

✓ **test_clock** string [EXPANDABLE](#)

ID of the test clock this quote belongs to.

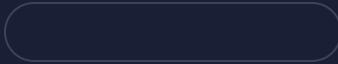
✓ **total_details** hash

Tax and discount details for the computed total amount.



✓ **transfer_data** hash [CONNECT ONLY](#)

The account (if any) the payments will be attributed to for tax reporting, and where funds from each payment will be transferred to for each of the invoices.



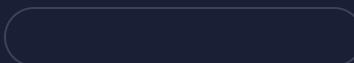
Create a quote

A quote models prices and services for a customer. Default options for [header](#), [description](#), [footer](#), and [expires_at](#) can be set in the dashboard via the [quote template](#).

Parameters

line_items optional array of hashes

A list of line items the customer is being quoted for. Each line item includes information about the product, the quantity, and the resulting cost.



metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ **application_fee_amount** optional CONNECT ONLY

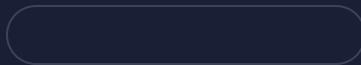
The amount of the application fee (if any) that will be requested to be applied to the payment and transferred to the application owner's Stripe account. There cannot be any line items with recurring prices when using this field.

✓ **application_fee_percent** optional CONNECT ONLY

A non-negative decimal between 0 and 100, with at most two decimal places. This represents the percentage of the subscription invoice subtotal that will be transferred to the application owner's Stripe account. There must be at least 1 line item with a recurring price to use this field.

✓ **automatic_tax** optional dictionary

Settings for automatic tax lookup for this quote and resulting invoices and subscriptions.



✓ **collection_method** optional

Either `charge_automatically`, or `send_invoice`. When charging automatically, Stripe will attempt to pay invoices at the end of the subscription cycle or at invoice finalization using the default payment method attached to the subscription or customer. When sending an invoice, Stripe will email your customer an invoice with payment instructions. Defaults to `charge_automatically`.

✓ **customer** optional

The customer for which this quote belongs to. A customer is required before finalizing the quote. Once specified, it cannot be changed.

✓ **default_tax_rates** optional

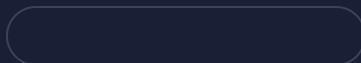
The tax rates that will apply to any line item that does not have `tax_rates` set.

✓ **description** optional

A description that will be displayed on the quote PDF. If no value is passed, the default description configured in your [quote template settings](#) will be used.

✓ **discounts** optional array of hashes

The discounts applied to the quote. You can only set up to one discount.



✓ **expires_at** optional

A future timestamp on which the quote will be canceled if in `open` or `draft` status.

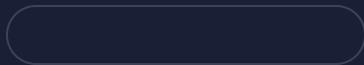
Measured in seconds since the Unix epoch. If no value is passed, the default expiration date configured in your [quote template settings](#) will be used.

✓ **footer** optional

A footer that will be displayed on the quote PDF. If no value is passed, the default footer configured in your [quote template settings](#) will be used.

✓ **from_quote** optional dictionary

Clone an existing quote. The new quote will be created in `status=draft`. When using this parameter, you cannot specify any other parameters except for `expires_at`.

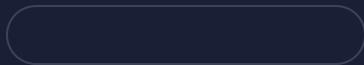


✓ **header** optional

A header that will be displayed on the quote PDF. If no value is passed, the default header configured in your [quote template settings](#) will be used.

✓ **invoice_settings** optional dictionary

All invoices will be billed using the specified settings.



✓ **on_behalf_of** optional CONNECT ONLY

The account on behalf of which to charge.

✓ **subscription_data** optional dictionary

When creating a subscription or subscription schedule, the specified configuration data will be used. There must be at least one line item with a recurring price for a subscription or subscription schedule to be created. A subscription schedule is created if `subscription_data[effective_date]` is present and in the future, otherwise a subscription is created.

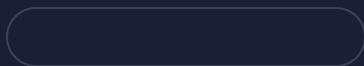


✓ **test_clock** optional

ID of the test clock to attach to the quote.

✓ **transfer_data** optional dictionary CONNECT ONLY

The data with which to automatically create a Transfer for each of the invoices.



Returns

Returns the quote object.

Retrieve a quote

Retrieves the quote with the given ID.

Parameters

No parameters.

Returns

Returns a quote if a valid quote ID was provided. Returns [an error](#) otherwise.

Update a quote

A quote models prices and services for a customer.

Parameters

line_items optional array of hashes

A list of line items the customer is being quoted for. Each line item includes information about the product, the quantity, and the resulting cost.



metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

More parameters

- › **application_fee_amount** optional CONNECT ONLY
- › **application_fee_percent** optional CONNECT ONLY
- › **automatic_tax** optional dictionary
- › **collection_method** optional
- › **customer** optional
- › **default_tax_rates** optional
- › **description** optional
- › **discounts** optional array of hashes
- › **expires_at** optional
- › **footer** optional
- › **header** optional
- › **invoice_settings** optional dictionary
- › **on_behalf_of** optional CONNECT ONLY
- › **subscription_data** optional dictionary
- › **transfer_data** optional dictionary CONNECT ONLY

Returns

Returns the updated quote object.

Finalize a quote

Finalizes the quote.

Parameters

- › **expires_at** optional

Returns

Returns an open quote. Returns [an error](#) otherwise.

Accept a quote

Accepts the specified quote.

Parameters

No parameters.

Returns

Returns an accepted quote and creates an invoice, subscription or subscription schedule.

Returns [an error](#) otherwise.

Cancel a quote

Cancels the quote.

Parameters

No parameters.

Returns

Returns a canceled quote. Returns [an error](#) otherwise.

Download quote PDF

Download the PDF for a finalized quote

Parameters

No parameters.

Returns

The PDF file for the quote.

List all quotes

Returns a list of your quotes.

Parameters

customer optional

The ID of the customer whose quotes will be retrieved.

status optional enum

The status of the quote.

Possible enum values

`draft`

`open`

`accepted`

`canceled`

More parameters

- > **ending_before** optional
- > **limit** optional
- > **starting_after** optional
- > **test_clock** optional

Returns

A dictionary with a `data` property that contains an array of up to `limit` quotes, starting after quote `starting_after`. Each entry in the array is a separate quote object. If no more quotes are available, the resulting array will be empty. This request should never return an error.

Retrieve a quote's line items

When retrieving a quote, there is an includable **line_items** property containing the first handful of those items. There is also a URL where you can retrieve the full (paginated) list of line items.

Parameters

- > **ending_before** optional
- > **limit** optional
- > **starting_after** optional

Returns

A dictionary with a `data` property that contains an array of up to `limit` quote line items, starting after Line Item `starting_after`. Each entry in the array is a separate Line Item object. If no more line items are available, the resulting array will be empty.

Retrieve a quote's upfront line items

When retrieving a quote, there is an includable `computed.upfront.line_items` property containing the first handful of those items. There is also a URL where you can retrieve the full (paginated) list of upfront line items.

Parameters

> `ending_before` optional

> `limit` optional

> `starting_after` optional

Returns

A dictionary with a `data` property that contains an array of up to `limit` upfront line items, starting after Line Item `starting_after`. Each entry in the array is a separate Line Item object. If no more upfront line items are available, the resulting array will be empty.

Subscriptions

Subscriptions allow you to charge a customer on a recurring basis.

Related guide: [Creating Subscriptions](#).

The subscription object

Attributes

`id` string

Unique identifier for the object.

cancel_at_period_end boolean

If the subscription has been canceled with the `at_period_end` flag set to `true`, `cancel_at_period_end` on the subscription will be true. You can use this attribute to determine whether a subscription that has a status of active is scheduled to be canceled at the end of the current period.

current_period_end timestamp

End of the current period that the subscription has been invoiced for. At the end of this period, a new invoice will be created.

current_period_start timestamp

Start of the current period that the subscription has been invoiced for.

customer string EXPANDABLE

ID of the customer who owns the subscription.

default_payment_method string EXPANDABLE

ID of the default payment method for the subscription. It must belong to the customer associated with the subscription. This takes precedence over `default_source`. If neither are set, invoices will use the customer's [invoice_settings.default_payment_method](#) or [default_source](#).

items list

List of subscription items, each with an attached price.

**latest_invoice** string EXPANDABLE

The most recent invoice this subscription has generated.

metadata hash

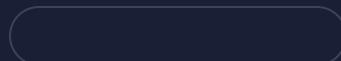
Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

pending_setup_intent string EXPANDABLE

You can use this [SetupIntent](#) to collect user authentication when creating a subscription without immediate payment or updating a subscription's payment method, allowing you to optimize for off-session payments. Learn more in the [SCA Migration Guide](#).

pending_update hash

If specified, [pending updates](#) that will be applied to the subscription once the `latest_invoice` has been paid.



status enum

Possible values are `incomplete`, `incomplete_expired`, `trialing`, `active`, `past_due`, `canceled`, or `unpaid`.

For `collection_method=charge_automatically` a subscription moves into `incomplete` if the initial payment attempt fails. A subscription in this state can only have metadata and `default_source` updated. Once the first invoice is paid, the subscription moves into an `active` state. If the first invoice is not paid within 23 hours, the subscription transitions to `incomplete_expired`. This is a terminal state, the open invoice will be voided and no further invoices will be generated.

A subscription that is currently in a trial period is `trialing` and moves to `active` when the trial period is over.

If subscription `collection_method=charge_automatically` it becomes `past_due` when payment to renew it fails and `canceled` or `unpaid` (depending on your subscriptions settings) when Stripe has exhausted all payment retry attempts.

If subscription `collection_method=send_invoice` it becomes `past_due` when its invoice is not paid by the due date, and `canceled` or `unpaid` if it is still not paid by an additional deadline after that. Note that when a subscription has a status of `unpaid`, no subsequent invoices will be attempted (invoices will be created, but then immediately automatically closed). After receiving updated payment information from a customer, you may choose to reopen and pay their closed invoices.

Possible enum values

`active`

`past_due`

`unpaid`

`canceled`

`incomplete`

`incomplete_expired`

`trialing`

More attributes

✓ `object` string, value is "subscription"

String representing the object's type. Objects of the same type share the same value.

✓ `application_fee_percent` decimal CONNECT ONLY

A non-negative decimal between 0 and 100, with at most two decimal places. This represents the percentage of the subscription invoice subtotal that will be transferred to the

application owner's Stripe account.

✓ **automatic_tax** hash

Automatic tax settings for this subscription.

✓ **billing_cycle_anchor** timestamp

Determines the date of the first full invoice, and, for plans with `month` or `year` intervals, the day of the month for subsequent invoices.

✓ **billing_thresholds** hash

Define thresholds at which an invoice will be sent, and the subscription advanced to a new billing period

✓ **cancel_at** timestamp

A date in the future at which the subscription will automatically get canceled

✓ **canceled_at** timestamp

If the subscription has been canceled, the date of that cancellation. If the subscription was canceled with `cancel_at_period_end`, `canceled_at` will reflect the time of the most recent update request, not the end of the subscription period when the subscription is automatically moved to a canceled state.

✓ **collection_method** string

Either `charge Automatically`, or `send_invoice`. When charging automatically, Stripe will attempt to pay this subscription at the end of the cycle using the default source attached to the customer. When sending an invoice, Stripe will email your customer an invoice with payment instructions.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **days_until_due** integer

Number of days a customer has to pay invoices generated by this subscription. This value will be `null` for subscriptions where `collection_method=charge Automatically`.

✓ **default_source** string EXPANDABLE

ID of the default payment source for the subscription. It must belong to the customer associated with the subscription and be in a chargeable state. If `default_payment_method` is also set, `default_payment_method` will take precedence. If neither are set, invoices will use the customer's `invoice_settings.default_payment_method` or `default_source`.

✓ **default_tax_rates** array of hashes

The tax rates that will apply to any subscription item that does not have `tax_rates` set.

Invoices created will have their `default_tax_rates` populated from the subscription.



✓ **discount** hash, [discount object](#)

Describes the current discount applied to this subscription, if there is one. When billing, a discount applied to a subscription overrides a discount applied on a customer-wide basis.

✓ **ended_at** timestamp

If the subscription has ended, the date the subscription ended.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **next_pending_invoice_item_invoice** timestamp

Specifies the approximate timestamp on which any pending invoice items will be billed according to the schedule provided at `pending_invoice_item_interval`.

✓ **pause_collection** hash

If specified, payment collection for this subscription will be paused.



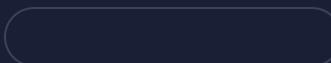
✓ **payment_settings** hash

Payment settings passed on to invoices created by the subscription.



✓ **pending_invoice_item_interval** hash

Specifies an interval for how often to bill for any pending invoice items. It is analogous to calling [Create an invoice](#) for the given subscription at the specified interval.



✓ **schedule** string [EXPANDABLE](#)

The schedule attached to the subscription

✓ **start_date** timestamp

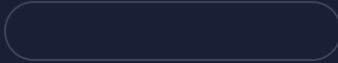
Date when the subscription was first created. The date might differ from the `created` date due to backdating.

✓ **test_clock** string [EXPANDABLE](#)

ID of the test clock this subscription belongs to.

✓ **transfer_data** hash CONNECT ONLY

The account (if any) the subscription's payments will be attributed to for tax reporting, and where funds from each payment will be transferred to for each of the subscription's invoices.



✓ **trial_end** timestamp

If the subscription has a trial, the end of that trial.

✓ **trial_start** timestamp

If the subscription has a trial, the beginning of that trial.

Create a subscription

Creates a new subscription on an existing customer. Each customer can have up to 500 active or scheduled subscriptions.

When you create a subscription with `collection_method=charge Automatically`, the first invoice is finalized as part of the request. The `payment_behavior` parameter determines the exact behavior of the initial payment.

To start subscriptions where the first invoice always begins in a `draft` status, use [subscription schedules](#) instead. Schedules provide the flexibility to model more complex billing configurations that change over time.

Parameters

customer REQUIRED

The identifier of the customer to subscribe.

items REQUIRED

A list of up to 20 subscription items, each with an attached price.



cancel_at_period_end optional

Boolean indicating whether this subscription should cancel at the end of the current period.

default_payment_method optional

ID of the default payment method for the subscription. It must belong to the customer associated with the subscription. This takes precedence over `default_source`. If neither are set, invoices will use the customer's `invoice_settings.default_payment_method` or `default_source`.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

payment_behavior optional enum

Use `allow_incomplete` to create subscriptions with `status=incomplete` if the first invoice cannot be paid. Creating subscriptions with this status allows you to manage scenarios where additional user actions are needed to pay a subscription's invoice. For example, SCA regulation may require 3DS authentication to complete payment. See the [SCA Migration Guide](#) for Billing to learn more. This is the default behavior.

Use `default_incomplete` to create Subscriptions with `status=incomplete` when the first invoice requires payment, otherwise start as active. Subscriptions transition to `status=active` when successfully confirming the payment intent on the first invoice. This allows simpler management of scenarios where additional user actions are needed to pay a subscription's invoice. Such as failed payments, [SCA regulation](#), or collecting a mandate for a bank debit payment method. If the payment intent is not confirmed within 23 hours subscriptions transition to `status=incomplete_expired`, which is a terminal state.

Use `error_if_incomplete` if you want Stripe to return an HTTP 402 status code if a subscription's first invoice cannot be paid. For example, if a payment method requires 3DS authentication due to SCA regulation and further user action is needed, this parameter does not create a subscription and returns an error instead. This was the default behavior for API versions prior to 2019-03-14. See the [changelog](#) to learn more.

`pending_if_incomplete` is only used with updates and cannot be passed when creating a subscription.

Possible enum values

`allow_incomplete`

`error_if_incomplete`

`pending_if_incomplete`

`default_incomplete`

More parameters

✓ **add_invoice_items** optional array of hashes

A list of prices and quantities that will generate invoice items appended to the first invoice for this subscription. You may pass up to 20 items.

✓ **application_fee_percent** optional `CONNECT ONLY`

A non-negative decimal between 0 and 100, with at most two decimal places. This represents the percentage of the subscription invoice subtotal that will be transferred to the application owner's Stripe account. The request must be made by a platform account on a connected account in order to set an application fee percentage. For more information, see the application fees [documentation](#).

✓ **automatic_tax** optional dictionary

Automatic tax settings for this subscription.

✓ **backdate_start_date** optional

For new subscriptions, a past timestamp to backdate the subscription's start date to. If set, the first invoice will contain a proration for the timespan between the start date and the current time. Can be combined with trials and the billing cycle anchor.

✓ **billing_cycle_anchor** optional

A future timestamp to anchor the subscription's [billing cycle](#). This is used to determine the date of the first full invoice, and, for plans with `month` or `year` intervals, the day of the month for subsequent invoices.

✓ **billing_thresholds** optional dictionary

Define thresholds at which an invoice will be sent, and the subscription advanced to a new billing period. Pass an empty string to remove previously-defined thresholds.

✓ **cancel_at** optional

A timestamp at which the subscription should cancel. If set to a date before the current period ends, this will cause a proration if prorations have been enabled using `proration_behavior`. If set during a future period, this will always cause a proration for that period.

✓ **collection_method** optional

Either `charge Automatically`, or `send_invoice`. When charging automatically, Stripe will attempt to pay this subscription at the end of the cycle using the default source attached to the customer. When sending an invoice, Stripe will email your customer an invoice with payment instructions. Defaults to `charge Automatically`.

✓ **coupon** optional

The ID of the coupon to apply to this subscription. A coupon applied to a subscription will only affect invoices created for that particular subscription.

✓ **days_until_due** optional

Number of days a customer has to pay invoices generated by this subscription. Valid only for subscriptions where `collection_method` is set to `send_invoice`.

✓ **default_source** optional

ID of the default payment source for the subscription. It must belong to the customer associated with the subscription and be in a chargeable state. If `default_payment_method` is also set, `default_payment_method` will take precedence. If neither are set, invoices will use the customer's `invoice_settings.default_payment_method` or `default_source`.

✓ **default_tax_rates** optional

The tax rates that will apply to any subscription item that does not have `tax_rates` set. Invoices created will have their `default_tax_rates` populated from the subscription.

✓ **off_session** optional

Indicates if a customer is on or off-session while an invoice payment is attempted.

✓ **payment_settings** optional dictionary

Payment settings to pass to invoices created by the subscription.

✓ **pending_invoice_item_interval** optional dictionary

Specifies an interval for how often to bill for any pending invoice items. It is analogous to calling [Create an invoice](#) for the given subscription at the specified interval.

✓ **promotion_code** optional

The API ID of a promotion code to apply to this subscription. A promotion code applied to a subscription will only affect invoices created for that particular subscription.

✓ **proration_behavior** optional

Determines how to handle **prorations** resulting from the `billing_cycle_anchor`. Valid values are `create_prorations` or `none`.

Passing `create_prorations` will cause proration invoice items to be created when applicable. Prorations can be disabled by passing `none`. If no value is passed, the default is `create_prorations`.

✓ **transfer_data** optional dictionary CONNECT ONLY

If specified, the funds from the subscription's invoices will be transferred to the destination and the ID of the resulting transfers will be found on the resulting charges.

◀ **trial_end** optional

Unix timestamp representing the end of the trial period the customer will get before being charged for the first time. This will always overwrite any trials that might apply via a subscribed plan. If set, `trial_end` will override the default trial period of the plan the customer is being subscribed to. The special value `now` can be provided to end the customer's trial immediately. Can be at most two years from `billing_cycle_anchor`. See [Using trial periods on subscriptions](#) to learn more.

◀ **trial_from_plan** optional

Indicates if a plan's `trial_period_days` should be applied to the subscription. Setting `trial_end` per subscription is preferred, and this defaults to `false`. Setting this flag to `true` together with `trial_end` is not allowed. See [Using trial periods on subscriptions](#) to learn more.

◀ **trial_period_days** optional

Integer representing the number of trial period days before the customer is charged for the first time. This will always overwrite any trials that might apply via a subscribed plan. See [Using trial periods on subscriptions](#) to learn more.

Returns

The newly created `Subscription` object, if the call succeeded. If the attempted charge fails, the subscription is created in an `incomplete` status.

Retrieve a subscription

Retrieves the subscription with the given ID.

Parameters

No parameters.

Returns

Returns the subscription object.

Update a subscription

Updates an existing subscription to match the specified parameters. When changing prices or quantities, we will optionally prorate the price we charge next month to make up for any price changes. To preview how the proration will be calculated, use the [upcoming invoice](#) endpoint.

By default, we prorate subscription changes. For example, if a customer signs up on May 1 for a \$100 price, she'll be billed \$100 immediately. If on May 15 she switches to a \$200 price, then on June 1 she'll be billed \$250 (\$200 for a renewal of her subscription, plus a \$50 prorating adjustment for half of the previous month's \$100 difference). Similarly, a downgrade will generate a credit to be applied to the next invoice. We also prorate when you make quantity changes.

Switching prices does not normally change the billing date or generate an immediate charge unless:

- The billing interval is changed (e.g., monthly to yearly).
- The subscription moves from free to paid, or paid to free.
- A trial starts or ends.

In these cases, we apply a credit for the time unused on the previous price and the customer is immediately charged using the new price. This also resets the billing date.

If you'd like to charge for an upgrade immediately, just pass `proration_behavior` as `create_prorations` (the default), and then [invoice the customer](#) as soon as you make the subscription change. That will collect the proration adjustments into a new invoice, and Stripe will automatically attempt to collect payment on the invoice.

If you don't want to prorate at all, set the `proration_behavior` option to `none` and the customer would be billed \$100 on May 1 and \$200 on June 1. Similarly, if you set `proration_behavior` to `none` when switching between different billing intervals (monthly to yearly, for example), we won't generate any credits for the old subscription's unused time —although we will still reset the billing date and will bill immediately for the new subscription.

Updating the quantity on a subscription many times in an hour may result in [rate limiting](#). If you need to bill for a frequently changing quantity, consider integrating [metered billing](#) instead.

Parameters

cancel_at_period_end optional

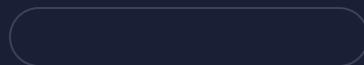
Boolean indicating whether this subscription should cancel at the end of the current period.

default_payment_method optional

ID of the default payment method for the subscription. It must belong to the customer associated with the subscription. This takes precedence over `default_source`. If neither are set, invoices will use the customer's [invoice_settings.default_payment_method](#) or `default_source`.

items optional array of hashes

A list of up to 20 subscription items, each with an attached price.

**metadata** optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

payment_behavior optional enum

Use `allow_incomplete` to transition the subscription to `status=past_due` if a payment is required but cannot be paid. This allows you to manage scenarios where additional user actions are needed to pay a subscription's invoice. For example, SCA regulation may require 3DS authentication to complete payment. See the [SCA Migration Guide](#) for Billing to learn more. This is the default behavior.

Use `default_incomplete` to transition the subscription to `status=past_due` when payment is required and await explicit confirmation of the invoice's payment intent. This allows simpler management of scenarios where additional user actions are needed to pay a subscription's invoice. Such as failed payments, [SCA regulation](#), or collecting a mandate for a bank debit payment method.

Use `pending_if_incomplete` to update the subscription using [pending updates](#). When you use `pending_if_incomplete` you can only pass the parameters [supported by pending updates](#).

Use `error_if_incomplete` if you want Stripe to return an HTTP 402 status code if a subscription's invoice cannot be paid. For example, if a payment method requires 3DS authentication due to SCA regulation and further user action is needed, this parameter does not update the subscription and returns an error instead. This was the default behavior for API versions prior to 2019-03-14. See the [changelog](#) to learn more.

Possible enum values

`allow_incomplete``error_if_incomplete`

`pending_if_incomplete`

`default_incomplete`

proration_behavior optional

Determines how to handle **prorations** when the billing cycle changes (e.g., when switching plans, resetting `billing_cycle_anchor=now`, or starting a trial), or if an item's `quantity` changes. Valid values are `create_prorations`, `none`, or `always_invoice`.

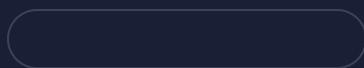
Passing `create_prorations` will cause proration invoice items to be created when applicable. These proration items will only be invoiced immediately under [certain conditions](#). In order to always invoice immediately for prorations, pass `always_invoice`.

Prorations can be disabled by passing `none`.

More parameters

✓ **add_invoice_items** optional array of hashes

A list of prices and quantities that will generate invoice items appended to the first invoice for this subscription. You may pass up to 20 items.

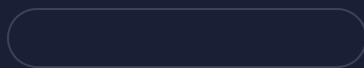


✓ **application_fee_percent** optional CONNECT ONLY

A non-negative decimal between 0 and 100, with at most two decimal places. This represents the percentage of the subscription invoice subtotal that will be transferred to the application owner's Stripe account. The request must be made by a platform account on a connected account in order to set an application fee percentage. For more information, see the application fees [documentation](#).

✓ **automatic_tax** optional dictionary

Automatic tax settings for this subscription.

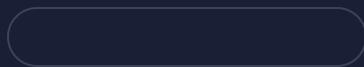


✓ **billing_cycle_anchor** optional

Either `now` or `unchanged`. Setting the value to `now` resets the subscription's billing cycle anchor to the current time. For more information, see the billing cycle [documentation](#).

✓ **billing_thresholds** optional dictionary

Define thresholds at which an invoice will be sent, and the subscription advanced to a new billing period. Pass an empty string to remove previously-defined thresholds.



✓ **cancel_at** optional

A timestamp at which the subscription should cancel. If set to a date before the current period ends, this will cause a proration if prorations have been enabled using `proration_behavior`. If set during a future period, this will always cause a proration for that period.

✓ **collection_method** optional

Either `charge Automatically`, or `send_invoice`. When charging automatically, Stripe will attempt to pay this subscription at the end of the cycle using the default source attached to the customer. When sending an invoice, Stripe will email your customer an invoice with payment instructions. Defaults to `charge Automatically`.

✓ **coupon** optional

The ID of the coupon to apply to this subscription. A coupon applied to a subscription will only affect invoices created for that particular subscription.

✓ **days_until_due** optional

Number of days a customer has to pay invoices generated by this subscription. Valid only for subscriptions where `collection_method` is set to `send_invoice`.

✓ **default_source** optional

ID of the default payment source for the subscription. It must belong to the customer associated with the subscription and be in a chargeable state. If `default_payment_method` is also set, `default_payment_method` will take precedence. If neither are set, invoices will use the customer's `invoice_settings.default_payment_method` or `default_source`.

✓ **default_tax_rates** optional

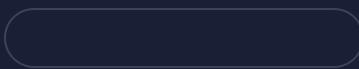
The tax rates that will apply to any subscription item that does not have `tax_rates` set. Invoices created will have their `default_tax_rates` populated from the subscription. Pass an empty string to remove previously-defined tax rates.

✓ **off_session** optional

Indicates if a customer is on or off-session while an invoice payment is attempted.

✓ **pause_collection** optional dictionary

If specified, payment collection for this subscription will be paused.



✓ **payment_settings** optional dictionary

Payment settings to pass to invoices created by the subscription.



✓ **pending_invoice_item_interval** optional dictionary

Specifies an interval for how often to bill for any pending invoice items. It is analogous to calling [Create an invoice](#) for the given subscription at the specified interval.

✓ **promotion_code** optional

The promotion code to apply to this subscription. A promotion code applied to a subscription will only affect invoices created for that particular subscription.

✓ **proration_date** optional

If set, the proration will be calculated as though the subscription was updated at the given time. This can be used to apply exactly the same proration that was previewed with [upcoming invoice](#) endpoint. It can also be used to implement custom proration logic, such as prorating by day instead of by second, by providing the time that you wish to use for proration calculations.

✓ **transfer_data** optional dictionary CONNECT ONLY

If specified, the funds from the subscription's invoices will be transferred to the destination and the ID of the resulting transfers will be found on the resulting charges. This will be unset if you POST an empty value.

✓ **trial_end** optional

Unix timestamp representing the end of the trial period the customer will get before being charged for the first time. This will always overwrite any trials that might apply via a subscribed plan. If set, trial_end will override the default trial period of the plan the customer is being subscribed to. The special value `now` can be provided to end the customer's trial immediately. Can be at most two years from `billing_cycle_anchor`.

✓ **trial_from_plan** optional

Indicates if a plan's `trial_period_days` should be applied to the subscription. Setting `trial_end` per subscription is preferred, and this defaults to `false`. Setting this flag to `true` together with `trial_end` is not allowed. See [Using trial periods on subscriptions](#) to learn more.

Returns

The newly updated `Subscription` object, if the call succeeded.

If `payment_behavior` is `error_if_incomplete` and a charge is required for the update and it fails, this call returns [an error](#), and the subscription update does not go into effect.

Cancel a subscription

Cancels a customer's subscription immediately. The customer will not be charged again for the subscription.

Note, however, that any pending invoice items that you've created will still be charged for at the end of the period, unless manually [deleted](#). If you've set the subscription to cancel at the end of the period, any pending prorations will also be left in place and collected at the end of the period. But if the subscription is set to cancel immediately, pending prorations will be removed.

By default, upon subscription cancellation, Stripe will stop automatic collection of all finalized invoices for the customer. This is intended to prevent unexpected payment attempts after the customer has canceled a subscription. However, you can resume automatic collection of the invoices manually after subscription cancellation to have us proceed. Or, you could check for unpaid invoices before allowing the customer to cancel the subscription at all.

Parameters

✓ **invoice_now** optional

Will generate a final invoice that invoices for any un-invoiced metered usage and new/pending proration invoice items.

✓ **prorate** optional

Will generate a proration invoice item that credits remaining unused time until the subscription period end.

Returns

The canceled `Subscription` object. Its subscription status will be set to `canceled`.

List subscriptions

By default, returns a list of subscriptions that have not been canceled. In order to list canceled subscriptions, specify `status=canceled`.

Parameters

customer optional

The ID of the customer whose subscriptions will be retrieved.

price optional

Filter for subscriptions that contain this recurring price ID.

status optional enum

The status of the subscriptions to retrieve. Passing in a value of `canceled` will return all canceled subscriptions, including those belonging to deleted customers. Pass `ended` to find subscriptions that are canceled and subscriptions that are expired due to [incomplete payment](#). Passing in a value of `all` will return subscriptions of all statuses. If no value is supplied, all subscriptions that have not been canceled are returned.

Possible enum values

`active`

`past_due`

`unpaid`

`canceled`

`incomplete`

`incomplete_expired`

`trialing`

`all`

`ended`

More parameters

✓ **collection_method** optional

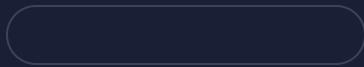
The collection method of the subscriptions to retrieve. Either `charge Automatically` or `Send Invoice`.

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

✓ **current_period_end** optional dictionary

A filter on the list based on the object `current_period_end` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **current_period_start** optional dictionary

A filter on the list based on the object `current_period_start` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

✓ **test_clock** optional

Filter for subscriptions that are associated with the specified test clock. The response will not include subscriptions with test clocks if this and the customer parameter is not set.

Returns

Returns a list of subscriptions.

Search subscriptions

Search for subscriptions you've previously created using Stripe's [Search Query Language](#).

Don't use search in read-after-write flows where strict consistency is necessary. Under normal operating conditions, data is searchable in less than a minute. Occasionally, propagation of new or updated data can be up to an hour behind during outages. Search functionality is not available to merchants in India.

Parameters

query REQUIRED

The search query string. See [search query language](#) and the list of supported [query fields for subscriptions](#).

limit optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

page optional

A cursor for pagination across multiple pages of results. Don't include this parameter on the first call. Use the next_page value returned in a previous response to request subsequent results.

Returns

A dictionary with a `data` property that contains an array of up to `limit` subscriptions. If no objects match the query, the resulting array will be empty. See the related guide on [expanding properties in lists](#).

Subscription Items

Subscription items allow you to create customer subscriptions with more than one plan, making it easy to represent complex billing relationships.

The subscription item object

Attributes

id string

Unique identifier for the object.

metadata hash

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

price hash

The price the customer is subscribed to.

quantity positive integer or zero

The **quantity** of the plan to which the customer should be subscribed.

subscription string

The `subscription` this `subscription_item` belongs to.

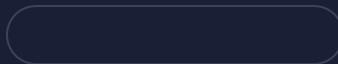
More attributes

✓ **object** string, value is "subscription_item"

String representing the object's type. Objects of the same type share the same value.

✓ **billing_thresholds** hash

Define thresholds at which an invoice will be sent, and the related subscription advanced to a new billing period

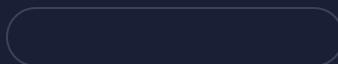


✓ **created** integer

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **tax_rates** array of hashes

The tax rates which apply to this `subscription_item`. When set, the `default_tax_rates` on the subscription do not apply to this `subscription_item`.



Create a subscription item

Adds a new item to an existing subscription. No existing items will be changed or replaced.

Parameters

subscription REQUIRED

The identifier of the subscription to modify.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

payment_behavior optional enum

Use `allow_incomplete` to transition the subscription to `status=past_due` if a payment is required but cannot be paid. This allows you to manage scenarios where additional user actions are needed to pay a subscription's invoice. For example, SCA regulation may require 3DS authentication to complete payment. See the [SCA Migration Guide](#) for Billing to learn more. This is the default behavior.

Use `default_incomplete` to transition the subscription to `status=past_due` when payment is required and await explicit confirmation of the invoice's payment intent. This allows simpler management of scenarios where additional user actions are needed to pay a subscription's invoice. Such as failed payments, [SCA regulation](#), or collecting a mandate for a bank debit payment method.

Use `pending_if_incomplete` to update the subscription using [pending updates](#). When you use `pending_if_incomplete` you can only pass the parameters [supported by pending updates](#).

Use `error_if_incomplete` if you want Stripe to return an HTTP 402 status code if a subscription's invoice cannot be paid. For example, if a payment method requires 3DS authentication due to SCA regulation and further user action is needed, this parameter does not update the subscription and returns an error instead. This was the default behavior for API versions prior to 2019-03-14. See the [changelog](#) to learn more.

Possible enum values

`allow_incomplete`

`error_if_incomplete`

`pending_if_incomplete`

`default_incomplete`

price optional

The ID of the price object.

proration_behavior optional

Determines how to handle **prorations** when the billing cycle changes (e.g., when switching plans, resetting `billing_cycle_anchor=now`, or starting a trial), or if an item's `quantity` changes. Valid values are `create_prorations`, `none`, or `always_invoice`.

Passing `create_prorations` will cause proration invoice items to be created when applicable. These proration items will only be invoiced immediately under **certain conditions**. In order to always invoice immediately for prorations, pass `always_invoice`.

Prorations can be disabled by passing `none`.

quantity optional

The quantity you'd like to apply to the subscription item you're creating.

More parameters

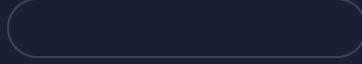
✓ **billing_thresholds** optional dictionary

Define thresholds at which an invoice will be sent, and the subscription advanced to a new billing period. When updating, pass an empty string to remove previously-defined thresholds.



✓ **price_data** optional dictionary

Data used to generate a new **Price** object inline.



✓ **proration_date** optional

If set, the proration will be calculated as though the subscription was updated at the given time. This can be used to apply the same proration that was previewed with the **upcoming invoice** endpoint.

✓ **tax_rates** optional

A list of **Tax Rate** ids. These Tax Rates will override the `default_tax_rates` on the Subscription. When updating, pass an empty string to remove previously-defined tax rates.

Returns

Returns the created `Subscription Item` object, if successful. Otherwise, this call returns an error.

Retrieve a subscription item

Retrieves the subscription item with the given ID.

Parameters

No parameters.

Returns

Returns a subscription item if a valid subscription item ID was provided. Returns [an error](#) otherwise.

Update a subscription item

Updates the plan or quantity of an item on a current subscription.

Parameters

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

payment_behavior optional enum

Use `allow_incomplete` to transition the subscription to `status=past_due` if a payment is required but cannot be paid. This allows you to manage scenarios where additional user actions are needed to pay a subscription's invoice. For example, SCA regulation may require 3DS authentication to complete payment. See the [SCA Migration Guide](#) for Billing to learn more. This is the default behavior.

Use `default_incomplete` to transition the subscription to `status=past_due` when payment is required and await explicit confirmation of the invoice's payment intent. This allows simpler management of scenarios where additional user actions are needed to pay a subscription's invoice. Such as failed payments, [SCA regulation](#), or collecting a mandate for a bank debit payment method.

Use `pending_if_incomplete` to update the subscription using [pending updates](#). When you use `pending_if_incomplete` you can only pass the parameters [supported by pending updates](#).

Use `error_if_incomplete` if you want Stripe to return an HTTP 402 status code if a subscription's invoice cannot be paid. For example, if a payment method requires 3DS authentication due to SCA regulation and further user action is needed, this parameter does not update the subscription and returns an error instead. This was the default behavior for API versions prior to 2019-03-14. See the [changelog](#) to learn more.

Possible enum values

`allow_incomplete`

`error_if_incomplete`

`pending_if_incomplete`

`default_incomplete`

price optional

The ID of the price object. When changing a subscription item's price, `quantity` is set to 1 unless a `quantity` parameter is provided.

proration_behavior optional

Determines how to handle [prorations](#) when the billing cycle changes (e.g., when switching plans, resetting `billing_cycle_anchor=now`, or starting a trial), or if an item's `quantity` changes. Valid values are `create_prorations`, `none`, or `always_invoice`.

Passing `create_prorations` will cause proration invoice items to be created when applicable. These proration items will only be invoiced immediately under [certain conditions](#). In order to always invoice immediately for prorations, pass `always_invoice`.

Prorations can be disabled by passing `none`.

quantity optional

The quantity you'd like to apply to the subscription item you're creating.

More parameters

✓ **billing_thresholds** optional dictionary

Define thresholds at which an invoice will be sent, and the subscription advanced to a new billing period. When updating, pass an empty string to remove previously-defined thresholds.

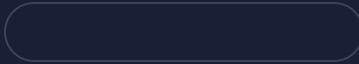


off_session optional

Indicates if a customer is on or off-session while an invoice payment is attempted.

✓ **price_data** optional dictionary

Data used to generate a new **Price** object inline.



✓ **proration_date** optional

If set, the proration will be calculated as though the subscription was updated at the given time. This can be used to apply the same proration that was previewed with the [upcoming invoice](#) endpoint.

✓ **tax_rates** optional

A list of **Tax Rate** ids. These Tax Rates will override the `default_tax_rates` on the Subscription. When updating, pass an empty string to remove previously-defined tax rates.

Returns

Delete a subscription item

Deletes an item from the subscription. Removing a subscription item from a subscription will not cancel the subscription.

Parameters

proration_behavior optional

Determines how to handle **prorations** when the billing cycle changes (e.g., when switching plans, resetting `billing_cycle_anchor=now`, or starting a trial), or if an item's `quantity` changes. Valid values are `create_prorations`, `none`, or `always_invoice`.

Passing `create_prorations` will cause proration invoice items to be created when applicable. These proration items will only be invoiced immediately under [certain conditions](#). In order to always invoice immediately for prorations, pass `always_invoice`.

Prorations can be disabled by passing `none`.

More parameters

[Collapse all](#)

✓ **clear_usage** optional

Delete all usage for the given subscription item. Allowed only when the current plan's `usage_type` is `metered`.

✓ **proration_date** optional

If set, the proration will be calculated as though the subscription was updated at the given time. This can be used to apply the same proration that was previewed with the [upcoming invoice](#) endpoint.

Returns

An subscription item object with a deleted flag upon success. Otherwise, this call returns an [error](#), such as if the subscription item has already been deleted.

List all subscription items

Returns a list of your subscription items for a given subscription.

Parameters

subscription REQUIRED

The ID of the subscription whose items will be retrieved.

More parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with

`obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` subscription items, starting after subscription item `starting_after`. Each entry in the array is a separate subscription item object. If no more subscription items are available, the resulting array will be empty. This request should never return an error.

Subscription Schedule

A subscription schedule allows you to create and manage the lifecycle of a subscription by predefining expected changes.

Related guide: [Subscription Schedules](#).

The schedule object

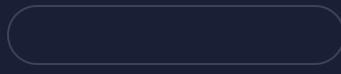
Attributes

id string

Unique identifier for the object.

current_phase hash

Object representing the start and end dates for the current phase of the subscription schedule, if it is `active`.



customer string EXPANDABLE

ID of the customer who owns the subscription schedule.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

phases array of hashes

status string

The present status of the subscription schedule. Possible values are `not_started`, `active`, `completed`, `released`, and `canceled`. You can read more about the different states in our [behavior guide](#).

subscription string EXPANDABLE

ID of the subscription managed by the subscription schedule.

More attributes

✓ **object** string, value is "subscription_schedule"

String representing the object's type. Objects of the same type share the same value.

✓ **canceled_at** timestamp

Time at which the subscription schedule was canceled. Measured in seconds since the Unix epoch.

✓ **completed_at** timestamp

Time at which the subscription schedule was completed. Measured in seconds since the Unix epoch.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **default_settings** hash

Object representing the subscription schedule's default settings.

✓ **end_behavior** string

Behavior of the subscription schedule and underlying subscription when it ends. Possible values are `release` and `cancel`.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **released_at** timestamp

Time at which the subscription schedule was released. Measured in seconds since the Unix epoch.

✓ **released_subscription** string

ID of the subscription once managed by the subscription schedule (if it is released).

✓ **test_clock** string EXPANDABLE

ID of the test clock this subscription schedule belongs to.

Create a schedule

Creates a new subscription schedule object. Each customer can have up to 500 active or scheduled subscriptions.

Parameters

customer optional

The identifier of the customer to create the subscription schedule for.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

phases optional array of hashes

List representing phases of the subscription schedule. Each phase can be customized to have different durations, plans, and coupons. If there are multiple phases, the `end_date` of one phase will always equal the `start_date` of the next phase.

start_date optional

When the subscription schedule starts. We recommend using `now` so that it starts the subscription immediately. You can also use a Unix timestamp to backdate the subscription so that it starts on a past date, or set a future date for the subscription to start on.

More parameters

✓ **default_settings** optional dictionary

Object representing the subscription schedule's default settings.

✓ **end_behavior** optional

Configures how the subscription schedule behaves when it ends. Possible values are `release` or `cancel` with the default being `release`. `release` will end the subscription schedule and keep the underlying subscription running. `cancel` will end the subscription schedule and cancel the underlying subscription.

✓ **from_subscription** optional

Migrate an existing subscription to be managed by a subscription schedule. If this parameter is set, a subscription schedule will be created using the subscription's item(s), set to auto-renew using the subscription's interval. When using this parameter, other parameters (such as phase values) cannot be set. To create a subscription schedule with other modifications, we recommend making two separate API calls.

Returns

Returns a subscription schedule object if the call succeeded.

Retrieve a schedule

Retrieves the details of an existing subscription schedule. You only need to supply the unique subscription schedule identifier that was returned upon subscription schedule creation.

Parameters

No parameters.

Returns

Returns a subscription schedule object if a valid identifier was provided.

Update a schedule

Updates an existing subscription schedule.

Parameters

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

phases optional array of hashes

List representing phases of the subscription schedule. Each phase can be customized to have different durations, plans, and coupons. If there are multiple phases, the `end_date` of one phase will always equal the `start_date` of the next phase. Note that past phases can be omitted.

More parameters

✓ **default_settings** optional dictionary

Object representing the subscription schedule's default settings.

✓ **end_behavior** optional

Configures how the subscription schedule behaves when it ends. Possible values are `release` or `cancel` with the default being `release`. `release` will end the subscription schedule and keep the underlying subscription running. `cancel` will end the subscription schedule and cancel the underlying subscription.

✓ **proration_behavior** optional

If the update changes the current phase, indicates if the changes should be prorated. Possible values are `create_prorations` or `none`, and the default value is `create_prorations`.

Returns

Returns an updated subscription schedule object if the call succeeded.

Cancel a schedule

Cancels a subscription schedule and its associated subscription immediately (if the subscription schedule has an active subscription). A subscription schedule can only be canceled if its status is `not_started` or `active`.

Parameters

`invoice_now` optional

If the subscription schedule is `active`, indicates if a final invoice will be generated that contains any un-invoiced metered usage and new/pending proration invoice items. Defaults to `true`.

More parameters

✓ `prorate` optional

If the subscription schedule is `active`, indicates if the cancellation should be prorated. Defaults to `true`.

Returns

The canceled `subscription_schedule` object. Its status will be `canceled` and `canceled_at` will be the current time.

Release a schedule

Releases the subscription schedule immediately, which will stop scheduling of its phases, but leave any existing subscription in place. A schedule can only be released if its status is `not_started` or `active`. If the subscription schedule is currently associated with a subscription, releasing it will remove its `subscription` property and set the subscription's ID to the `released_subscription` property.

Parameters

✓ `preserve_cancel_date` optional

Keep any cancellation on the subscription that the schedule has set

Returns

The released `subscription_schedule` object. Its status will be `released`, `released_at` will be the current time, and `released_subscription` will be the ID of the subscription the subscription schedule managed prior to being released.

List all schedules

Retrieves the list of your subscription schedules.

Parameters

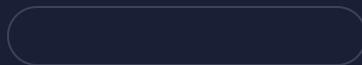
customer optional

Only return subscription schedules for the given customer.

More parameters

✓ **canceled_at** optional dictionary

A filter on the list based on the object `canceled_at` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **completed_at** optional dictionary

A filter on the list based on the object `completed_at` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

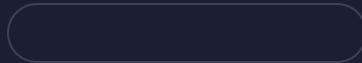
A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **released_at** optional dictionary

A filter on the list based on the object `released_at` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **scheduled** optional

Only return subscription schedules that have not started yet.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` subscription schedules, starting after subscription schedule `starting_after`. Each entry in the array is a separate subscription schedule object. If no more subscription schedules are available, the resulting array will be empty. This request should never return an error.

Test Clocks

A test clock enables deterministic control over objects in testmode. With a test clock, you can create objects at a frozen time in the past or future, and advance to a specific future time to observe webhooks and state changes. After the clock advances, you can either validate the current state of your scenario (and test your assumptions), change the current state of your scenario (and test more complex scenarios), or keep advancing forward in time.

The test clock object

Attributes

id string

Unique identifier for the object.

object string, value is "test_helpers.test_clock"

String representing the object's type. Objects of the same type share the same value.

created timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

deletes_after timestamp

Time at which this clock is scheduled to auto delete.

frozen_time timestamp

Time at which all objects belonging to this clock are frozen.

livemode boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

name string

The custom name supplied at creation.

status enum

The status of the Test Clock.

Possible enum values

`ready`

All test clock objects have advanced to the `frozen_time`.

`advancing`

In the process of advancing time for the test clock objects.

`internal_failure`

Failed to advance time. Future requests to advance time will fail.

Create a test clock

Creates a new test clock that can be attached to new customers and quotes.

Parameters

frozen_time REQUIRED

The initial frozen time for this test clock.

name optional

The name for this test clock.

Returns

The newly created `TestClock` object is returned upon success. Otherwise, this call returns [an error](#).

Retrieve a test clock

Retrieves a test clock.

Parameters

No parameters.

Returns

Returns the `TestClock` object. Otherwise, this call returns [an error](#).

Delete a test clock

Deletes a test clock.

Parameters

No parameters.

Returns

The deleted `TestClock` object is returned upon success. Otherwise, this call returns [an error](#).

Advance a test clock

Starts advancing a test clock to a specified time in the future. Advancement is done when status changes to `Ready`.

Parameters

`frozen_time` REQUIRED

The time to advance the test clock. Must be after the test clock's current frozen time. Cannot be more than two intervals in the future from the shortest subscription in this test clock. If there are no subscriptions in this test clock, it cannot be more than two years in the future.

Returns

A `TestClock` object with status `Advancing` is returned upon success. Otherwise, this call returns [an error](#).

List all test clocks

Returns a list of your test clocks.

Parameters

> `ending_before` optional

> `limit` optional

> `starting_after` optional

Returns

A dictionary with a `data` property that contains an array of up to `limit` test clocks, starting after `starting_after`. Each entry in the array is a separate test clock object. If no more test clocks are available, the resulting array will be empty. This request should never return an error.

Usage Records

Usage records allow you to report customer usage and metrics to Stripe for metered billing of subscription prices.

Related guide: [Metered Billing](#).

The usage record object

Attributes

id string

Unique identifier for the object.

quantity positive integer or zero

The usage quantity for the specified date.

subscription_item string

The ID of the subscription item this usage record contains data for.

timestamp timestamp

The timestamp when this usage occurred.

More attributes

[Collapse all](#)

✓ **object** string, value is "usage_record"

String representing the object's type. Objects of the same type share the same value.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

Create a usage record

Creates a usage record for a specified subscription item and date, and fills it with a quantity.

Usage records provide `quantity` information that Stripe uses to track how much a customer is using your service. With usage information and the pricing model set up by the **metered billing** plan, Stripe helps you send accurate invoices to your customers.

The default calculation for usage is to add up all the `quantity` values of the usage records within a billing period. You can change this default behavior with the billing plan's `aggregate_usage` parameter. When there is more than one usage record with the same timestamp, Stripe adds the `quantity` values together. In most cases, this is the desired resolution, however, you can change this behavior with the `action` parameter.

The default pricing model for metered billing is **per-unit pricing**. For finer granularity, you can configure metered billing to have a **tiered pricing** model.

Parameters

quantity REQUIRED

The usage quantity for the specified timestamp.

action optional enum

Valid values are `increment` (default) or `set`. When using `increment` the specified `quantity` will be added to the usage at the specified timestamp. The `set` action will overwrite the usage quantity at that timestamp. If the subscription has **billing thresholds**, `increment` is the only allowed value.

Possible enum values

`set`

`increment`

timestamp optional

The timestamp for the usage event. This timestamp must be within the current billing period of the subscription of the provided `subscription_item`, and must not be in the future.

When passing `"now"`, Stripe records usage for the current time. Default is `"now"` if a value is not provided.

Returns

Returns the usage record object.

List all subscription item period summaries

For the specified subscription item, returns a list of summary objects. Each object in the list provides usage information that's been summarized from multiple usage records and over a subscription billing period (e.g., 15 usage records in the month of September).

The list is sorted in reverse-chronological order (newest first). The first list item represents the most current usage period that hasn't ended yet. Since new usage records can still be added, the returned summary information for the subscription item's ID should be seen as unstable until the subscription billing period ends.

Parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` summaries, starting after summary `starting_after`. Each entry in the array is a separate summary object. If no more summaries are available, the resulting array is empty.

Accounts

This is an object representing a Stripe account. You can retrieve it to see properties on the account like its current e-mail address or if the account is enabled yet to make live charges.

Some properties, marked below, are available only to platforms that want to [create and manage Express or Custom accounts](#).

The account object

Attributes

id string

Unique identifier for the object.

business_type enum CUSTOM ONLY

The business type.

Possible enum values

`individual`

`company`

`non_profit`

`government_entity`

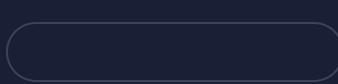
US only

capabilities hash

A hash containing the set of capabilities that was requested for this account and their associated states. Keys are names of capabilities. You can see the full list [here](#). Values may be `active`, `inactive`, or `pending`.

company hash CUSTOM ONLY

Information about the company or business. This field is available for any `business_type`.



country string

The account's country.

email string

An email address associated with the account. You can treat this as metadata: it is not used for authentication or messaging account holders.

individual hash CUSTOM ONLY

Information about the person represented by the account. This field is null unless `business_type` is set to `individual`.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

requirements hash

Information about the requirements for the account, including what information needs to be collected, and by when.

tos_acceptance hash CUSTOM AND EXPRESS

Details on the [acceptance of the Stripe Services Agreement](#)

type string

The Stripe account type. Can be `standard`, `express`, or `custom`.

More attributes

✓ **object** string, value is "account"

String representing the object's type. Objects of the same type share the same value.

✓ **business_profile** hash

Business information about the account.



✓ **charges_enabled** boolean

Whether the account can create live charges.

✓ **controller** hash

The controller of the account.

✓ **created** timestamp

Time at which the account was connected. Measured in seconds since the Unix epoch.

✓ **default_currency** string

Three-letter ISO currency code representing the default currency for the account. This must be a currency that [Stripe supports in the account's country](#).

✓ **details_submitted** boolean

Whether account details have been submitted. Standard accounts cannot receive payouts before this is true.

✓ **external_accounts** list

External accounts (bank accounts and debit cards) currently attached to this account

✓ **future_requirements** hash

Information about the upcoming new requirements for the account, including what information needs to be collected, and by when.

✓ **payouts_enabled** boolean

Whether Stripe can send payouts to this account.

✓ **settings** hash

Options for customizing how the account functions within Stripe.

Create an account

With [Connect](#), you can create Stripe accounts for your users. To do this, you'll first need to [register your platform](#).

Parameters

type REQUIRED

The type of Stripe account to create. May be one of `custom`, `express` or `standard`.

country optional DEFAULT IS YOUR OWN COUNTRY

The country in which the account holder resides, or in which the business is legally established. This should be an ISO 3166-1 alpha-2 country code. For example, if you are in the United States and the business for which you're creating an account is legally represented in Canada, you would use `CA` as the country for the account being created.

email optional

The email address of the account holder. This is only to make the account easier to identify to you. Stripe only emails Custom accounts with your consent.

capabilities REQUIRED FOR CUSTOM ACCOUNTS

Each key of the dictionary represents a capability, and each capability maps to its settings (e.g. whether it has been requested or not). Each capability will be inactive until you have provided its specific requirements and Stripe has verified them. An account may have some of its requested capabilities be active and some be inactive.

business_type optional enum

The business type.

Possible enum values

`individual`

`company`

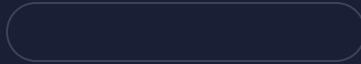
`non_profit`

`government_entity`

US only

company optional dictionary

Information about the company or business. This field is available for any `business_type`.



individual optional dictionary

Information about the person represented by the account. This field is null unless `business_type` is set to `individual`.

+ Show child parameters

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

tos_acceptance optional dictionary

Details on the account's acceptance of the [Stripe Services Agreement](#).

More parameters

✓ **account_token** optional

An [account token](#), used to securely provide details to the account.

✓ **business_profile** optional dictionary

Business information about the account.

✓ **default_currency** optional

Three-letter ISO currency code representing the default currency for the account. This must be a currency that [Stripe supports in the account's country](#).

✓ **documents** optional dictionary

Documents that may be submitted to satisfy various informational requests.

✓ **external_account** optional dictionary

A card or bank account to attach to the account for receiving [payouts](#) (you won't be able to use it for top-ups). You can provide either a token, like the ones returned by [Stripe.js](#), or a dictionary, as documented in the [external_account](#) parameter for [bank account creation](#).

By default, providing an external account sets it as the new default external account for its currency, and deletes the old default if one exists. To add additional external accounts without replacing the existing default for the currency, use the bank account or card creation API.

✓ **settings** optional dictionary

Options for customizing how the account functions within Stripe.



Returns

Returns an [Account](#) object if the call succeeds.

Retrieve account

Retrieves the details of an account.

Parameters

No parameters.

Returns

Returns an [Account](#) object if the call succeeds. If the account ID does not exist, this call returns [an error](#).

Update an account

Updates a [connected account](#) by setting the values of the parameters passed. Any parameters not provided are left unchanged. Most parameters can be changed only for Custom accounts. (These are marked **Custom Only** below.) Parameters marked **Custom and Express** are not supported for Standard accounts.

To update your own account, use the [Dashboard](#). Refer to our [Connect](#) documentation to learn more about updating accounts.

Parameters

business_type optional enum CUSTOM ONLY

The business type.

Possible enum values

`individual`

`company`

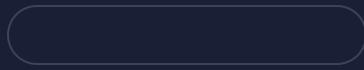
`non_profit`

`government_entity`

US only

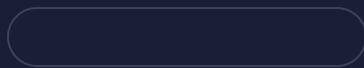
capabilities optional dictionary

Each key of the dictionary represents a capability, and each capability maps to its settings (e.g. whether it has been requested or not). Each capability will be inactive until you have provided its specific requirements and Stripe has verified them. An account may have some of its requested capabilities be active and some be inactive.



company optional dictionary CUSTOM ONLY

Information about the company or business. This field is available for any `business_type`.

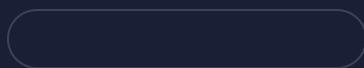


email optional CUSTOM ONLY

The email address of the account holder. This is only to make the account easier to identify to you. Stripe only emails Custom accounts with your consent.

individual optional dictionary CUSTOM ONLY

Information about the person represented by the account. This field is null unless `business_type` is set to `individual`.

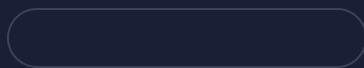


metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

tos_acceptance optional dictionary CUSTOM ONLY

Details on the account's acceptance of the [Stripe Services Agreement](#).



More parameters

✓ **account_token** optional

An [account token](#), used to securely provide details to the account.

✓ **business_profile** optional dictionary CUSTOM AND EXPRESS

Business information about the account.

✓ **default_currency** optional CUSTOM ONLY

Three-letter ISO currency code representing the default currency for the account. This must be a currency that [Stripe supports in the account's country](#).

✓ **documents** optional dictionary CUSTOM AND EXPRESS

Documents that may be submitted to satisfy various informational requests.

✓ **external_account** optional dictionary CUSTOM ONLY

A card or bank account to attach to the account for receiving [payouts](#) (you won't be able to use it for top-ups). You can provide either a token, like the ones returned by [Stripe.js](#), or a dictionary, as documented in the `external_account` parameter for [bank account creation](#).

By default, providing an external account sets it as the new default external account for its currency, and deletes the old default if one exists. To add additional external accounts without replacing the existing default for the currency, use the bank account or card creation API.

✓ **settings** optional dictionary

Options for customizing how the account functions within Stripe.

Returns

Returns an `Account` object if the call succeeds. If the account ID does not exist or another issue occurs, this call returns an [error](#).

Delete an account

With [Connect](#), you can delete accounts you manage.

Accounts created using test-mode keys can be deleted at any time. Standard accounts created using live-mode keys cannot be deleted. Custom or Express accounts created using live-mode keys can only be deleted once all balances are zero.

If you want to delete your own account, use the [account information tab in your account settings](#) instead.

Parameters

No parameters.

Returns

Returns an object with a `deleted` parameter if the call succeeds. If the account ID does not exist, this call returns [an error](#).

Reject an account

With [Connect](#), you may flag accounts as suspicious.

Test-mode Custom and Express accounts can be rejected at any time. Accounts created using live-mode keys may only be rejected once all balances are zero.

Parameters

reason REQUIRED

The reason for rejecting the account. Can be `fraud`, `terms_of_service`, or `other`.

Returns

Returns an account with `payouts_enabled` and `charges_enabled` set to false on success. If the account ID does not exist, this call returns [an error](#).

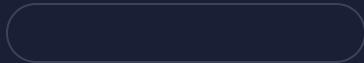
List all connected accounts

Returns a list of accounts connected to your platform via [Connect](#). If you're not a platform, the list is empty.

Parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` accounts, starting after account `starting_after`. Each entry in the array is a separate `Account` object. If no more accounts are available, the resulting array is empty.

The login link object

Attributes

url string

The URL for the login link.

More attributes

✓ **object** string, value is "login_link"

String representing the object's type. Objects of the same type share the same value.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

Create a login link

Creates a single-use login link for an Express account to access their Stripe dashboard.

You may only create login links for Express accounts connected to your platform.

Parameters

redirect_url optional

Where to redirect the user after they log out of their dashboard.

Returns

Returns a login link object if the call succeeded.

Account Links

Account Links are the means by which a Connect platform grants a connected account permission to access Stripe-hosted applications, such as Connect Onboarding.

Related guide: [Connect Onboarding](#).

The account link object

Attributes

expires_at timestamp

The timestamp at which this account link will expire.

url string

The URL for the account link.

More attributes

✓ **object** string, value is "account_link"

String representing the object's type. Objects of the same type share the same value.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

Create an account link

Creates an AccountLink object that includes a single-use Stripe URL that the platform can redirect their user to in order to take them through the Connect Onboarding flow.

Parameters

account REQUIRED

The identifier of the account to create an account link for.

refresh_url REQUIRED

The URL the user will be redirected to if the account link is expired, has been previously-visited, or is otherwise invalid. The URL you specify should attempt to generate a new account link with the same parameters used to create the original account link, then redirect the user to the new account link's URL so they can continue with Connect Onboarding. If a new account link cannot be generated or the redirect fails you should display a useful error to the user.

return_url REQUIRED

The URL that the user will be redirected to upon leaving or completing the linked flow.

type REQUIRED

The type of account link the user is requesting. Possible values are `account_onboarding` or `account_update`.

Possible enum values

`account_onboarding`

Provides a form for inputting outstanding requirements. Send the user to the form in this mode to just collect the new information you need.

`account_update` CUSTOM ONLY

Displays the fields that are already populated on the account object, and allows your user to edit previously provided information. Consider framing this as "edit my profile" or "update my verification information".

More parameters

✓ `collect` optional CUSTOM & EXPRESS ONLY

Which information the platform needs to collect from the user. One of `currently_due` or `eventually_due`. Default is `currently_due`.

Returns

Returns an account link object if the call succeeded.

Application Fees

When you collect a transaction fee on top of a charge made for your user (using [Connect](#)), an `Application Fee` object is created in your account. You can list, retrieve, and refund application fees.

Related guide: [Collecting Application Fees](#).

The application fee object

Attributes

id string

Unique identifier for the object.

account string EXPANDABLE

ID of the Stripe account this fee was taken from.

amount integer

Amount earned, in cents.

amount_refunded positive integer or zero

Amount in cents refunded (can be less than the amount attribute on the fee if a partial refund was issued)

charge string EXPANDABLE

ID of the charge that the application fee was taken from.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

refunded boolean

Whether the fee has been fully refunded. If the fee is only partially refunded, this attribute will still be false.

More attributes

✓ **object** string, value is "application_fee"

String representing the object's type. Objects of the same type share the same value.

✓ **application** string EXPANDABLE "APPLICATION"

ID of the Connect application that earned the fee.

✓ **balance_transaction** string EXPANDABLE

Balance transaction that describes the impact of this collected application fee on your account balance (not including refunds).

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **originating_transaction** string EXPANDABLE

ID of the corresponding charge on the platform account, if this fee was the result of a charge using the `destination` parameter.

✓ **refunds** list

A list of refunds that have been applied to the fee.

Retrieve an application fee

Retrieves the details of an application fee that your account has collected. The same information is returned when refunding the application fee.

Parameters

No parameters.

Returns

Returns an application fee object if a valid identifier was provided, and returns [an error](#) otherwise.

List all application fees

Returns a list of application fees you've previously collected. The application fees are returned in sorted order, with the most recent fees appearing first.

Parameters

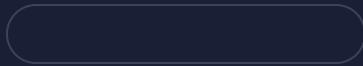
charge optional

Only return application fees for the charge specified by this charge ID.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` application fees, starting after application fee `starting_after`. Each entry in the array is a separate application fee object. If no more fees are available, the resulting array will be empty.

Application Fee Refunds

`Application Fee Refund` objects allow you to refund an application fee that has previously been created but not yet refunded. Funds will be refunded to the Stripe account from which the fee was originally collected.

Related guide: [Refunding Application Fees](#).

The application fee refund object

Attributes

id string

Unique identifier for the object.

amount integer

Amount, in cents.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

fee string [EXPANDABLE](#)

ID of the application fee that was refunded.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

More attributes

✓ **object** string, value is "fee_refund"

String representing the object's type. Objects of the same type share the same value.

✓ **balance_transaction** string [EXPANDABLE](#)

Balance transaction that describes the impact on your account balance.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

Create an application fee refund

Refunds an application fee that has previously been collected but not yet refunded. Funds will be refunded to the Stripe account from which the fee was originally collected.

You can optionally refund only part of an application fee. You can do so multiple times, until the entire fee has been refunded.

Once entirely refunded, an application fee can't be refunded again. This method will return an error when called on an already-refunded application fee, or when trying to refund more money than is left on an application fee.

Parameters

amount optional, default is entire application fee

A positive integer, in *cents*, representing how much of this fee to refund. Can refund only up to the remaining unrefunded amount of the fee.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the `Application Fee Refund` object if the refund succeeded. Returns [an error](#) if the fee has already been refunded, or if an invalid fee identifier was provided.

Retrieve an application fee refund

By default, you can see the 10 most recent refunds stored directly on the application fee object, but you can also retrieve details about a specific refund stored on the application fee.

Parameters

No parameters.

Returns

Returns the application fee refund object.

Update an application fee refund

Updates the specified application fee refund by setting the values of the parameters passed. Any parameters not provided will be left unchanged.

This request only accepts metadata as an argument.

Parameters

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns the application fee refund object if the update succeeded. This call will return [an error](#) if update parameters are invalid.

List all application fee refunds

You can see a list of the refunds belonging to a specific application fee. Note that the 10 most recent refunds are always available by default on the application fee object. If you need more than those 10, you can use this API method and the `limit` and `starting_after` parameters to page through additional refunds.

Parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with

`obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` refunds, starting after `starting_after`. Each entry in the array is a separate application fee refund object. If no more refunds are available, the resulting array will be empty. If you provide a non-existent application fee ID, this call returns [an error](#).

Capabilities

This is an object representing a capability for a Stripe account.

Related guide: [Account capabilities](#).

The capability object

Attributes

id string

The identifier for the capability.

account string EXPANDABLE

The account for which the capability enables functionality.

requested boolean

Whether the capability has been requested.

requirements hash

Information about the requirements for the capability, including what information needs to be collected, and by when.



status string

The status of the capability. Can be `active`, `inactive`, `pending`, or `unrequested`.

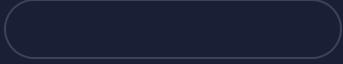
More attributes

✓ **object** string, value is "capability"

String representing the object's type. Objects of the same type share the same value.

✓ **future_requirements** hash

Information about the upcoming new requirements for the capability, including what information needs to be collected, and by when.



✓ **requested_at** timestamp

Time at which the capability was requested. Measured in seconds since the Unix epoch.

Retrieve an Account Capability

Retrieves information about the specified Account Capability.

Parameters

No parameters.

Returns

Returns an Account Capability object.

Update an Account Capability

Updates an existing Account Capability.

Parameters

requested optional

Passing true requests the capability for the account, if it is not already requested. A requested capability may not immediately become active. Any requirements to activate the capability are returned in the `requirements` arrays.

Returns

Returns an Account Capability object.

List all account capabilities

Returns a list of capabilities associated with the account. The capabilities are returned sorted by creation date, with the most recent capability appearing first.

Parameters

No parameters.

Returns

A dictionary with a `data` property that contains an array of the capabilities of this account. Each entry in the array is a separate capability object.

Country Specs

Stripe needs to collect certain pieces of information about each account created. These requirements can differ depending on the account's country. The Country Specs API makes these rules available to your integration.

You can also view the information from this API call as [an online guide](#).

The country spec object

Attributes

id string

Unique identifier for the object. Represented as the ISO country code for this country.

default_currency string

The default currency for this country. This applies to both payment methods and bank accounts.

supported_bank_account_currencies hash

Currencies that can be accepted in the specific country (for transfers).

supported_payment_currencies array containing strings

Currencies that can be accepted in the specified country (for payments).

supported_payment_methods array containing strings

Payment methods available in the specified country. You may need to enable some payment methods (e.g., ACH) on your account before they appear in this list. The `stripe` payment method refers to [charging through your platform](#).

supported_transfer_countries array containing strings

Countries that can accept transfers from the specified country.

More attributes

✓ **object** string, value is "country_spec"

String representing the object's type. Objects of the same type share the same value.

✓ **verification_fields** hash

Lists the types of verification data needed to keep an account open.

List Country Specs

Lists all Country Spec objects available in the API.

Parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

Returns a list of `country_spec` objects.

Retrieve a Country Spec

Returns a Country Spec for a given Country code.

Parameters

No parameters.

Returns

Returns a `country_spec` object if a valid country code is provided, and returns [an error](#) otherwise.

External Accounts

External Accounts are transfer destinations on `Account` objects for [connected accounts](#). They can be bank accounts or debit cards.

[Bank accounts](#) and [debit cards](#) can also be used as payment sources on regular charges, and are documented in the links above.

Related guide: [Setting Bank and Debit Card Payouts](#).

The (account) bank account object

Attributes

id string

Unique identifier for the object.

account string [EXPANDABLE](#)

The ID of the account that the bank account is associated with.

bank_name string

Name of the bank associated with the routing number (e.g., `WELLS_FARGO`).

country string

Two-letter ISO code representing the country the bank account is located in.

currency currency

Three-letter ISO code for the currency paid out to the bank account.

default_for_currency boolean

Whether this bank account is the default external account for its currency.

last4 string

The last four digits of the bank account number.

metadata hash

Set of key-value pairs that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

routing_number string

The routing transit number for the bank account.

status string

For bank accounts, possible values are `new`, `validated`, `verified`, `verification_failed`, or `errored`. A bank account that hasn't had any activity or validation performed is `new`. If Stripe can determine that the bank account exists, its status will be `validated`. Note that there often isn't enough information to know (e.g., for smaller credit unions), and the validation is not always run. If customer bank account verification has succeeded, the bank account status will be `verified`. If the verification failed for any reason, such as microdeposit failure, the status will be `verification_failed`. If a transfer sent to this bank account fails, we'll set the status to `errored` and will not continue to send transfers until the bank details are updated.

For external accounts, possible values are `new` and `errored`. Validations aren't run against external accounts because they're only used for payouts. This means the other statuses don't apply. If a transfer fails, the status is set to `errored` and transfers are stopped until account details are updated.

More attributes

✓ **object** string, value is "bank_account"

String representing the object's type. Objects of the same type share the same value.

✓ **account_holder_name** string

The name of the person or business that owns the bank account.

✓ **account_holder_type** string

The type of entity that holds the account. This can be either `individual` or `company`.

✓ **account_type** string

The bank account type. This can only be `checking` or `savings` in most countries. In Japan, this can only be `futsu` or `toza`.

✓ **available_payout_methods** array

A set of available payout methods for this bank account. Only values from this set should be passed as the `method` when creating a payout.

✓ **customer** string EXPANDABLE

The ID of the customer that the bank account is associated with.

✓ **fingerprint** string

Uniquely identifies this particular bank account. You can use this attribute to check whether two bank accounts are the same.

Create a bank account

When you create a new bank account, you must specify a [Custom account](#) to create it on.

If the bank account's owner has no other external account in the bank account's currency, the new bank account will become the default for that currency. However, if the owner already has a bank account for that currency, the new account will become the default only if the `default_for_currency` parameter is set to `true`.

Parameters

external_account REQUIRED

Either a token, like the ones returned by [Stripe.js](#), or a dictionary containing a user's bank account details (with the options shown below).

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ **default_for_currency** optional

When set to true, or if this is the first external account added in this currency, this account becomes the default external account for its currency.

Returns

Returns the bank account object.

Retrieve a bank account

By default, you can see the 10 most recent external accounts stored on a [connected account](#) directly on the object. You can also retrieve details about a specific bank account stored on the account.

Parameters

Returns

Returns the bank account object.

Update a bank account

Updates the metadata, account holder name, account holder type of a bank account belonging to a [Custom account](#), and optionally sets it as the default for its currency. Other bank account details are not editable by design.

You can re-enable a disabled bank account by performing an update call without providing any arguments or changes.

Parameters

default_for_currency optional

When set to true, this becomes the default external account for its currency.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

More parameters

✓ **account_holder_name** optional

The name of the person or business that owns the bank account.

✓ **account_holder_type** optional

The type of entity that holds the account. This can be either [individual](#) or [company](#).

✓ **account_type** optional

The bank account type. This can only be [checking](#) or [savings](#) in most countries. In Japan, this can only be [futsu](#) or [toza](#).

Returns

Returns the bank account object.

Delete a bank account

You can delete destination bank accounts from a [Custom account](#).

There are restrictions for deleting a bank account with `default_for_currency` set to true.

You cannot delete a bank account if any of the following apply:

- The bank account's `currency` is the same as the [Account \[default_currency\]](#).
- There is another external account (card or bank account) with the same currency as the bank account.

To delete a bank account, you must first replace the default external account by setting `default_for_currency` with another external account in the same currency.

Parameters

No parameters.

Returns

Returns the deleted bank account object.

List all bank accounts

You can see a list of the bank accounts that belong to a [connected account](#). Note that the 10 most recent external accounts are always available by default on the corresponding Stripe object. If you need more than those 10, you can use this API method and the `limit` and `starting_after` parameters to page through additional bank accounts.

Parameters

✓ `ending_before` optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

Returns a list of the bank accounts stored on the account.

The (account) card object

Attributes

id string

Unique identifier for the object.

account string EXPANDABLE CUSTOM CONNECT ONLY

The account this card belongs to. This attribute will not be in the card object if the card belongs to a customer or recipient instead.

address_city string

City/District/Suburb/Town/Village.

address_country string

Billing address country, if provided when creating card.

address_line1 string

Address line 1 (Street address/PO Box/Company name).

address_line2 string

Address line 2 (Apartment/Suite/Unit/Building).

address_state string

State/County/Province/Region.

address_zip string

ZIP or postal code.

address_zip_check string

If `address_zip` was provided, results of the check: `pass`, `fail`, `unavailable`, or `unchecked`.

brand string

Card brand. Can be `American Express`, `Diners Club`, `Discover`, `JCB`, `MasterCard`, `UnionPay`, `Visa`, or `Unknown`.

country string

Two-letter ISO code representing the country of the card. You could use this attribute to get a sense of the international breakdown of cards you've collected.

currency currency CUSTOM CONNECT ONLY

Three-letter [ISO code for currency](#). Only applicable on accounts (not customers or recipients). The card can be used as a transfer destination for funds in this currency.

cvc_check string

If a CVC was provided, results of the check: `pass`, `fail`, `unavailable`, or `unchecked`. A result of `unchecked` indicates that CVC was provided but hasn't been checked yet. Checks are typically performed when attaching a card to a Customer object, or when creating a charge. For more details, see [Check if a card is valid without a charge](#).

default_for_currency boolean CUSTOM CONNECT ONLY

Whether this card is the default external account for its currency.

exp_month integer

Two-digit number representing the card's expiration month.

exp_year integer

Four-digit number representing the card's expiration year.

fingerprint string

Uniquely identifies this particular card number. You can use this attribute to check whether two customers who've signed up with you are using the same card number, for example. For payment methods that tokenize card information (Apple Pay, Google Pay), the tokenized number might be provided instead of the underlying card number.

Starting May 1, 2021, card fingerprint in India for Connect will change to allow two fingerprints for the same card — one for India and one for the rest of the world.

funding string

Card funding type. Can be `credit`, `debit`, `prepaid`, or `unknown`.

last4 string

The last four digits of the card.

metadata hash

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

name string

Cardholder name.

status string

For external accounts, possible values are `new` and `errored`. If a transfer fails, the status is set to `errored` and transfers are stopped until account details are updated.

More attributes

✓ **object** string, value is "card"

String representing the object's type. Objects of the same type share the same value.

✓ **address_line1_check** string

If `address_line1` was provided, results of the check: `pass`, `fail`, `unavailable`, or `unchecked`.

✓ **available_payout_methods** array

A set of available payout methods for this card. Only values from this set should be passed as the `method` when creating a payout.

✓ **customer** string EXPANDABLE

The customer that this card belongs to. This attribute will not be in the card object if the card belongs to an account or recipient instead.

✓ **dynamic_last4** string

(For tokenized numbers only.) The last four digits of the device account number.

✓ **recipient** string EXPANDABLE

The recipient that this card belongs to. This attribute will not be in the card object if the card belongs to a customer or account instead.

✓ **tokenization_method** string

If the card number is tokenized, this is the method that was used. Can be `android_pay` (includes Google Pay), `apple_pay`, `masterpass`, `visa_checkout`, or null.

Create a card

When you create a new debit card, you must specify a [Custom account](#) to create it on.

If the account has no default destination card, then the new card will become the default. However, if the owner already has a default then it will not change. To change the default, you should set `default_for_currency` to `true` when creating a card for a Custom account.

Parameters

external_account REQUIRED

A token, like the ones returned by [Stripe.js](#). Stripe will automatically validate the card.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ **default_for_currency** optional

When set to true, or if this is the first external account added in this currency, this account becomes the default external account for its currency.

Returns

Returns the card object.

Retrieve a card

By default, you can see the 10 most recent external accounts stored on a [connected account](#) directly on the object. You can also retrieve details about a specific card stored on the account.

Parameters

No parameters.

Returns

Returns the card object.

Update a card

If you need to update only some card details, like the billing address or expiration date, you can do so without having to re-enter the full card details. Stripe also works directly with card networks so that your customers can [continue using your service](#) without interruption.

Parameters

default_for_currency optional

When set to true, this becomes the default external account for its currency.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ **address_city** optional

City/District/Suburb/Town/Village.

✓ **address_country** optional

Billing address country, if provided when creating card.

✓ **address_line1** optional

Address line 1 (Street address/PO Box/Company name).

✓ **address_line2** optional

Address line 2 (Apartment/Suite/Unit/Building).

✓ **address_state** optional

State/County/Province/Region.

✓ **address_zip** optional

ZIP or postal code.

✓ **exp_month** optional

Two digit number representing the card's expiration month.

✓ **exp_year** optional

Four digit number representing the card's expiration year.

✓ **name** optional

Cardholder name.

Returns

Returns the card object.

Delete a card

You can delete cards from a [Custom account](#).

There are restrictions for deleting a card with `default_for_currency` set to true. You cannot delete a card if any of the following apply:

- The card's `currency` is the same as the [Account \[default_currency\]](#).
- There is another external account (card or bank account) with the same currency as the card.

To delete a card, you must first replace the default external account by setting `default_for_currency` with another external account in the same currency.

Parameters

No parameters.

Returns

Returns the deleted card object.

List all cards

You can see a list of the cards that belong to a [connected account](#). The 10 most recent external accounts are available on the account object. If you need more than 10, you can use this API method and the `limit` and `starting_after` parameters to page through additional cards.

Parameters

✓ `ending_before` optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ `limit` optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ `starting_after` optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

Returns a list of the cards stored on the account.

Person

This is an object representing a person associated with a Stripe account.

A platform cannot access a Standard or Express account's persons after the account starts onboarding, such as after generating an account link for the

account. See the [Standard onboarding](#) or [Express onboarding](#) documentation for information about platform pre-filling and account onboarding steps.

Related guide: [Handling Identity Verification with the API](#).

The person object

Attributes

id string

Unique identifier for the object.

account string

The account the person is associated with.

address hash

The person's address.

dob hash

The person's date of birth.

[Show child attributes](#)

email string

The person's email address.

first_name string

The person's first name.

last_name string

The person's last name.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

phone string

The person's phone number.

relationship hash

Describes the person's relationship to the account.

requirements hash

Information about the requirements for this person, including what information needs to be collected, and by when.

More attributes

✓ **object** string, value is "person"

String representing the object's type. Objects of the same type share the same value.

✓ **address_kana** hash

The Kana variation of the person's address (Japan only).

✓ **address_kanji** hash

The Kanji variation of the person's address (Japan only).

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **first_name_kana** string

The Kana variation of the person's first name (Japan only).

✓ **first_name_kanji** string

The Kanji variation of the person's first name (Japan only).

✓ **full_name_aliases** array containing strings

A list of alternate names or aliases that the person is known by.

✓ **future_requirements** hash

Information about the upcoming new requirements for this person, including what information needs to be collected, and by when.

✓ **gender** string

The person's gender (International regulations require either "male" or "female").

✓ **id_number_provided** boolean

Whether the person's `id_number` was provided.

✓ **last_name_kana** string

The Kana variation of the person's last name (Japan only).

✓ **last_name_kanji** string

The Kanji variation of the person's last name (Japan only).

✓ **maiden_name** string

The person's maiden name.

✓ **nationality** string

The country where the person is a national.

✓ **political_exposure** enum

Indicates if the person or any of their representatives, family members, or other closely related persons, declares that they hold or have held an important public job or function, in any jurisdiction.

Possible enum values

`none`

The person has disclosed that they have no political exposure

`existing`

The person has disclosed that they do have political exposure

✓ **ssn_last_4_provided** boolean

Whether the last four digits of the person's Social Security number have been provided (U.S. only).

✓ **verification** hash

The person's verification status.



Create a person

Creates a new person.

Parameters

address optional dictionary

The person's address.

dob optional dictionary

The person's date of birth.

email optional

The person's email address.

first_name optional

The person's first name.

id_number optional

The person's ID number, as appropriate for their country. For example, a social security number in the U.S., social insurance number in Canada, etc. Instead of the number itself, you can also provide a [PII token provided by Stripe.js](#).

last_name optional

The person's last name.

metadata optional dictionary

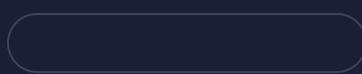
Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

phone optional

The person's phone number.

relationship optional dictionary

The relationship that this person has with the account's legal entity.



ssn_last_4 optional

The last four digits of the person's Social Security number (U.S. only).

[More parameters](#)

✓ **address_kana** optional dictionary

The Kana variation of the person's address (Japan only).

✓ **address_kanji** optional dictionary

The Kanji variation of the person's address (Japan only).

✓ **documents** optional dictionary

Documents that may be submitted to satisfy various informational requests.

✓ **first_name_kana** optional

The Kana variation of the person's first name (Japan only).

✓ **first_name_kanji** optional

The Kanji variation of the person's first name (Japan only).

✓ **full_name_aliases** optional

A list of alternate names or aliases that the person is known by.

✓ **gender** optional

The person's gender (International regulations require either "male" or "female").

✓ **last_name_kana** optional

The Kana variation of the person's last name (Japan only).

✓ **last_name_kanji** optional

The Kanji variation of the person's last name (Japan only).

✓ **maiden_name** optional

The person's maiden name.

✓ **nationality** optional

The country where the person is a national. Two-letter country code ([ISO 3166-1 alpha-2](#)), or "XX" if unavailable.

✓ **person_token** optional

A [person token](#), used to securely provide details to the person.

✓ **political_exposure** optional

Indicates if the person or any of their representatives, family members, or other closely related persons, declares that they hold or have held an important public job or function, in any jurisdiction.

✓ **verification** optional dictionary

The person's verification status.

Returns

Returns a person object.

Retrieve a person

Retrieves an existing person.

Parameters

No parameters.

Returns

Returns a person object.

Update a person

Updates an existing person.

Parameters

address optional dictionary

The person's address.

dob optional dictionary

The person's date of birth.

email optional

The person's email address.

first_name optional

The person's first name.

id_number optional

The person's ID number, as appropriate for their country. For example, a social security number in the U.S., social insurance number in Canada, etc. Instead of the number itself, you can also provide a [PII token provided by Stripe.js](#).

last_name optional

The person's last name.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

phone optional

The person's phone number.

relationship optional dictionary

The relationship that this person has with the account's legal entity.

ssn_last_4 optional

The last four digits of the person's Social Security number (U.S. only).

More parameters

✓ **address_kana** optional dictionary

The Kana variation of the person's address (Japan only).

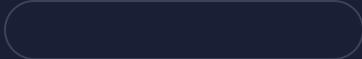
✓ **address_kanji** optional dictionary

The Kanji variation of the person's address (Japan only).



✓ **documents** optional dictionary

Documents that may be submitted to satisfy various informational requests.



✓ **first_name_kana** optional

The Kana variation of the person's first name (Japan only).

✓ **first_name_kanji** optional

The Kanji variation of the person's first name (Japan only).

✓ **full_name_aliases** optional

A list of alternate names or aliases that the person is known by.

✓ **gender** optional

The person's gender (International regulations require either "male" or "female").

✓ **last_name_kana** optional

The Kana variation of the person's last name (Japan only).

✓ **last_name_kanji** optional

The Kanji variation of the person's last name (Japan only).

✓ **maiden_name** optional

The person's maiden name.

✓ **nationality** optional

The country where the person is a national. Two-letter country code ([ISO 3166-1 alpha-2](#)), or "XX" if unavailable.

✓ **person_token** optional

A [person token](#), used to securely provide details to the person.

✓ **political_exposure** optional

Indicates if the person or any of their representatives, family members, or other closely related persons, declares that they hold or have held an important public job or function, in any jurisdiction.

✓ **verification** optional dictionary

The person's verification status.

Returns

Returns a person object.

Delete a person

Deletes an existing person's relationship to the account's legal entity. Any person with a relationship for an account can be deleted through the API, except if the person is the `account_opener`. If your integration is using the `executive` parameter, you cannot delete the only verified `executive` on file.

Parameters

No parameters.

Returns

Returns the deleted person object.

List all persons

Returns a list of people associated with the account's legal entity. The people are returned sorted by creation date, with the most recent people appearing first.

Parameters

relationship optional dictionary

Filters on the list of people returned based on the person's relationship to the account's company.

More parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` people, starting after person `starting_after`. Each entry in the array is a separate person object. If no more people are available, the resulting array will be empty.

Top-ups

To top up your Stripe balance, you create a top-up object. You can retrieve individual top-ups, as well as list all top-ups. Top-ups are identified by a unique, random ID.

Related guide: [Topping Up your Platform Account](#).

The top-up object

Attributes

id string

Unique identifier for the object.

amount integer

Amount transferred.

currency string

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

description string

An arbitrary string attached to the object. Often useful for displaying to users.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

status string

The status of the top-up is either `canceled`, `failed`, `pending`, `reversed`, or `succeeded`.

More attributes

` **object** string, value is "topup"

String representing the object's type. Objects of the same type share the same value.

` **balance_transaction** string [EXPANDABLE](#)

ID of the balance transaction that describes the impact of this top-up on your account balance. May not be specified depending on status of top-up.

` **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

` **expected_availability_date** integer

Date the funds are expected to arrive in your Stripe account for payouts. This factors in delays like weekends or bank holidays. May not be specified depending on status of top-up.

` **failure_code** string

Error code explaining reason for top-up failure if available (see [the errors section](#) for a list of codes).

` **failure_message** string

Message to user further explaining reason for top-up failure if available.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **source** hash, source object

For most Stripe users, the source of every top-up is a bank account. This hash is then the [source object](#) describing that bank account.

✓ **statement_descriptor** string

Extra information about a top-up. This will appear on your source's bank statement. It must contain at least one letter.

✓ **transfer_group** string

A string that identifies this top-up as part of a group.

Create a top-up

Top up the balance of an account

Parameters

amount REQUIRED

A positive integer representing how much to transfer.

currency REQUIRED

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ **source** optional

The ID of a source to transfer funds from. For most users, this should be left unspecified which will use the bank account that was set up in the dashboard for the specified currency. In test mode, this can be a test bank token (see [Testing Top-ups](#)).

✓ **statement_descriptor** optional

Extra information about a top-up for the source's bank statement. Limited to 15 ASCII characters.

✓ **transfer_group** optional

A string that identifies this top-up as part of a group.

Returns

Returns the top-up object.

Retrieve a top-up

Retrieves the details of a top-up that has previously been created. Supply the unique top-up ID that was returned from your previous request, and Stripe will return the corresponding top-up information.

Parameters

No parameters.

Returns

Returns a top-up if a valid identifier was provided, and returns [an error](#) otherwise.

Update a top-up

Updates the metadata of a top-up. Other top-up details are not editable by design.

Parameters

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

The newly updated top-up object if the call succeeded. Otherwise, this call returns an error.

List all top-ups

Returns a list of top-ups.

Parameters**status** optional

Only return top-ups that have the given status. One of `canceled`, `failed`, `pending` or `succeeded`.

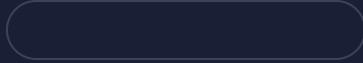
More parameters

▼ **amount** optional dictionary

A filter on the list based on the object `amount` field. The value can be a string with an integer amount, or it can be a dictionary with the following options:

▼ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

▼ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary containing the `data` property, which is an array of separate top-up objects. The number of top-ups in the array is limited to the number designated in `limit`. If no more top-ups are available, the resulting array will be empty. This request should never return an error.

Cancel a top-up

Cancels a top-up. Only pending top-ups can be canceled.

Parameters

No parameters.

Returns

Returns the canceled top-up. If the top-up is already canceled or can't be canceled, an error is returned.

Transfers

A `Transfer` object is created when you move funds between Stripe accounts as part of Connect.

Before April 6, 2017, transfers also represented movement of funds from a Stripe account to a card or bank account. This behavior has since been split out into a [Payout](#) object, with corresponding payout endpoints. For more information, read about the [transfer/payout split](#).

Related guide: [Creating Separate Charges and Transfers](#).

The transfer object

Attributes

id string

Unique identifier for the object.

amount integer

Amount in cents to be transferred.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

description string

An arbitrary string attached to the object. Often useful for displaying to users.

destination string EXPANDABLE

ID of the Stripe account the transfer was sent to.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

More attributes

✓ **object** string, value is "transfer"

String representing the object's type. Objects of the same type share the same value.

✓ **amount_reversed** integer

Amount in cents reversed (can be less than the amount attribute on the transfer if a partial reversal was issued).

✓ **balance_transaction** string [EXPANDABLE](#)

Balance transaction that describes the impact of this transfer on your account balance.

✓ **created** timestamp

Time that this record of the transfer was first created.

✓ **destination_payment** string [EXPANDABLE](#)

If the destination is a Stripe account, this will be the ID of the payment that the destination account received for the transfer.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **reversals** list

A list of reversals that have been applied to the transfer.

✓ **reversed** boolean

Whether the transfer has been fully reversed. If the transfer is only partially reversed, this attribute will still be false.

✓ **source_transaction** string [EXPANDABLE](#)

ID of the charge or payment that was used to fund the transfer. If null, the transfer was funded from the available balance.

✓ **source_type** string

The source balance this transfer came from. One of `card`, `fpx`, or `bank_account`.

✓ **transfer_group** string

A string that identifies this transaction as part of a group. See the [Connect documentation](#) for details.

To send funds from your Stripe account to a connected account, you create a new transfer object. Your [Stripe balance](#) must be able to cover the transfer amount, or you'll receive an "Insufficient Funds" error.

Parameters

amount REQUIRED

A positive integer in cents representing how much to transfer.

currency REQUIRED

3-letter [ISO code for currency](#).

destination REQUIRED

The ID of a connected Stripe account. [See the Connect documentation](#) for details.

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ **source_transaction** optional

You can use this parameter to transfer funds from a charge before they are added to your available balance. A pending balance will transfer immediately but the funds will not become available until the original charge becomes available. [See the Connect documentation](#) for details.

✓ **source_type** optional

The source balance to use for this transfer. One of `bank_account`, `card`, or `fpx`. For most users, this will default to `card`.

✓ **transfer_group** optional

A string that identifies this transaction as part of a group. See the [Connect documentation](#) for details.

Returns

Returns a transfer object if there were no initial errors with the transfer creation (e.g., insufficient funds).

Retrieve a transfer

Retrieves the details of an existing transfer. Supply the unique transfer ID from either a transfer creation request or the transfer list, and Stripe will return the corresponding transfer information.

Parameters

No parameters.

Returns

Returns a transfer object if a valid identifier was provided, and returns [an error](#) otherwise.

Update a transfer

Updates the specified transfer by setting the values of the parameters passed. Any parameters not provided will be left unchanged.

This request accepts only metadata as an argument.

Parameters

description optional

An arbitrary string attached to the object. Often useful for displaying to users.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

Returns

Returns the transfer object if the update succeeded. This call will return an error if update parameters are invalid.

List all transfers

Returns a list of existing transfers sent to connected accounts. The transfers are returned in sorted order, with the most recently created transfers appearing first.

Parameters

destination optional

Only return transfers for the destination specified by this account ID.

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

✓ **transfer_group** optional

Only return transfers with the specified transfer group.

Returns

A dictionary with a `data` property that contains an array of up to `limit` transfers, starting after transfer `starting_after`. Each entry in the array is a separate transfer object. If no more transfers are available, the resulting array will be empty.

Transfer Reversals

Stripe Connect platforms can reverse transfers made to a connected account, either entirely or partially, and can also specify whether to refund any related application fees. Transfer reversals add to the platform's balance and subtract from the destination account's balance.

Reversing a transfer that was made for a `destination_charge` is allowed only up to the amount of the charge. It is possible to reverse a `transfer_group` transfer only if the destination account has enough balance to cover the reversal.

Related guide: [Reversing Transfers](#).

The transfer reversal object

Attributes

id string

Unique identifier for the object.

amount integer

Amount, in cents.

currency currency

Three-letter ISO currency code, in lowercase. Must be a supported currency.

metadata hash

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

transfer string EXPANDABLE

ID of the transfer that was reversed.

More attributes

✓ **object** string, value is "transfer_reversal"

String representing the object's type. Objects of the same type share the same value.

✓ **balance_transaction** string EXPANDABLE

Balance transaction that describes the impact on your account balance.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **destination_payment_refund** string EXPANDABLE

Linked payment refund for the transfer reversal.

✓ **source_refund** string EXPANDABLE

ID of the refund responsible for the transfer reversal.

Create a transfer reversal

When you create a new reversal, you must specify a transfer to create it on.

When reversing transfers, you can optionally reverse part of the transfer. You can do so as many times as you wish until the entire transfer has been reversed.

Once entirely reversed, a transfer can't be reversed again. This method will return an error when called on an already-reversed transfer, or when trying to reverse more money than is left on a transfer.

Parameters

amount optional

A positive integer in cents representing how much of this transfer to reverse. Can only reverse up to the unreversed amount remaining of the transfer. Partial transfer reversals are

only allowed for transfers to Stripe Accounts. Defaults to the entire transfer amount.

description optional

An arbitrary string which you can attach to a reversal object. It is displayed alongside the reversal in the Dashboard. This will be unset if you POST an empty value.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

More parameters

✓ **refund_application_fee** optional

Boolean indicating whether the application fee should be refunded when reversing this transfer. If a full transfer reversal is given, the full application fee will be refunded. Otherwise, the application fee will be refunded with an amount proportional to the amount of the transfer reversed.

Returns

Returns a transfer reversal object if the reversal succeeded. Returns [an error](#) if the transfer has already been reversed or an invalid transfer identifier was provided.

Retrieve a reversal

By default, you can see the 10 most recent reversals stored directly on the transfer object, but you can also retrieve details about a specific reversal stored on the transfer.

Parameters

No parameters.

Returns

Returns the reversal object.

Update a reversal

Updates the specified reversal by setting the values of the parameters passed. Any parameters not provided will be left unchanged.

This request only accepts metadata and description as arguments.

Parameters

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to [metadata](#).

Returns

Returns the reversal object if the update succeeded. This call will return [an error](#) if update parameters are invalid.

List all reversals

You can see a list of the reversals belonging to a specific transfer. Note that the 10 most recent reversals are always available by default on the transfer object. If you need more than those 10, you can use this API method and the [limit](#) and [starting_after](#) parameters to page through additional reversals.

Parameters

✓ **ending_before** optional

A cursor for use in pagination. [ending_before](#) is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with [obj_bar](#), your subsequent call can include [ending_before=obj_bar](#) in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` reversals, starting after reversal `starting_after`. Each entry in the array is a separate reversal object. If no more reversals are available, the resulting array will be empty. If you provide a non-existent transfer ID, this call returns [an error](#).

Early Fraud Warning

An early fraud warning indicates that the card issuer has notified us that a charge may be fraudulent.

Related guide: [Early Fraud Warnings](#).

The early fraud warning object

Attributes

id string

Unique identifier for the object.

object string, value is "radar.early_fraud_warning"

String representing the object's type. Objects of the same type share the same value.

actionable boolean

An EFW is actionable if it has not received a dispute and has not been fully refunded. You may wish to proactively refund a charge that receives an EFW, in order to avoid receiving a dispute later.

charge string EXPANDABLE

ID of the charge this early fraud warning is for, optionally expanded.

created timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

fraud_type string

The type of fraud labelled by the issuer. One of `card_never_received`, `fraudulent_card_application`, `made_with_counterfeit_card`, `made_with_lost_card`, `made_with_stolen_card`, `misc`, `unauthorized_use_of_card`.

livemode boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

payment_intent string EXPANDABLE

ID of the Payment Intent this early fraud warning is for, optionally expanded.

Retrieve an early fraud warning

Retrieves the details of an early fraud warning that has previously been created.

Please refer to the [early fraud warning](#) object reference for more details.

Parameters

No parameters.

Returns

Returns an EarlyFraudWarning if a valid identifier was provided.

List all early fraud warnings

Returns a list of early fraud warnings.

Parameters

charge optional

Only return early fraud warnings for the charge specified by this charge ID.

payment_intent optional

Only return early fraud warnings for charges that were created by the PaymentIntent specified by this PaymentIntent ID.

More parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` EarlyFraudWarnings, starting after EarlyFraudWarnings `starting_after`. Each entry in the array is a separate EarlyFraudWarning object. If no more EarlyFraudWarnings are available, the resulting array will be empty. This request should never return an error.

Reviews

Reviews can be used to supplement automated fraud detection with human expertise.

Learn more about [Radar](#) and reviewing payments [here](#).

The review object

Attributes

id string

Unique identifier for the object.

charge string [EXPANDABLE](#)

The charge associated with this review.

open boolean

If `true`, the review needs action.

payment_intent string [EXPANDABLE](#)

The PaymentIntent ID associated with this review, if one exists.

reason string

The reason the review is currently open or closed. One of `rule`, `manual`, `approved`, `refunded`, `refunded_as_fraud`, `disputed`, or `redacted`.

More attributes

✓ **object** string, value is "review"

String representing the object's type. Objects of the same type share the same value.

✓ **billing_zip** string

The ZIP or postal code of the card used, if applicable.

✓ **closed_reason** string

The reason the review was closed, or null if it has not yet been closed. One of `approved`, `refunded`, `refunded_as_fraud`, `disputed`, or `redacted`.

✓ **created** timestamp

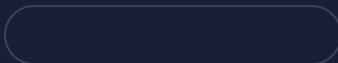
Time at which the object was created. Measured in seconds since the Unix epoch.

ip_address string

The IP address where the payment originated.

✓ **ip_address_location** hash

Information related to the location of the payment. Note that this information is an approximation and attempts to locate the nearest population center - it should not be used to determine a specific address.



✓ **livemode** boolean

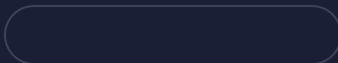
Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **opened_reason** string

The reason the review was opened. One of `rule` or `manual`.

✓ **session** hash

Information related to the browsing session of the user who initiated the payment.



Approve a review

Approves a `Review` object, closing it and removing it from the list of reviews.

Parameters

No parameters.

Returns

Returns the approved `Review` object.

Retrieve a review

Retrieves a `Review` object.

Parameters

No parameters.

Returns

Returns a `Review` object if a valid identifier was provided.

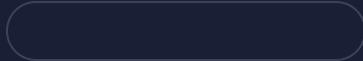
List all open reviews

Returns a list of `Review` objects that have `open` set to `true`. The objects are sorted in descending order by creation date, with the most recently created object appearing first.

Parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` reviews, starting after review `starting_after`. Each entry in the array is a separate `Review` object. If no more reviews are available, the resulting array will be empty.

Value Lists

Value lists allow you to group values together which can then be referenced in rules.

Related guide: [Default Stripe Lists](#).

The value list object

Attributes

`id` string

Unique identifier for the object.

`alias` string

The name of the value list for use in rules.

`item_type` string

The type of items in the value list. One of `card_fingerprint`, `card_bin`, `email`, `ip_address`, `country`, `string`, `case_sensitive_string`, or `customer_id`.

`list_items` list

List of items contained within this value list.



`metadata` hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

`name` string

The name of the value list.

More attributes

✓ `object` string, value is "radar.value_list"

String representing the object's type. Objects of the same type share the same value.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **created_by** string

The name or email address of the user who created this value list.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

Create a value list

Creates a new `ValueList` object, which can then be referenced in rules.

Parameters

alias REQUIRED

The name of the value list for use in rules.

name REQUIRED

The human-readable name of the value list.

item_type optional

Type of the items in the value list. One of `card_fingerprint`, `card_bin`, `email`, `ip_address`, `country`, `string`, `case_sensitive_string`, or `customer_id`. Use `string` if the item type is unknown or mixed.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns a `ValueList` object if creation succeeds.

Retrieve a value list

Retrieves a `ValueList` object.

Parameters

No parameters.

Returns

Returns a `ValueList` object if a valid identifier was provided.

Update a value list

Updates a `ValueList` object by setting the values of the parameters passed. Any parameters not provided will be left unchanged. Note that `item_type` is immutable.

Parameters

alias optional

The name of the value list for use in rules.

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

name optional

The human-readable name of the value list.

Returns

Returns an updated `ValueList` object if a valid identifier was provided.

Delete a value list

Deletes a `ValueList` object, also deleting any items contained within the value list. To be deleted, a value list must not be referenced in any rules.

Parameters

No parameters.

Returns

Returns an object with the deleted `ValueList` object's ID and a `deleted` parameter on success. Otherwise, this call returns [an error](#).

List all value lists

Returns a list of `ValueList` objects. The objects are sorted in descending order by creation date, with the most recently created object appearing first.

Parameters

alias optional

The alias used to reference the value list when writing rules.

More parameters

✓ **contains** optional

A value contained within a value list - returns all value lists containing this value.

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` lists, starting after list `starting_after`. Each entry in the array is a separate `ValueList` object. If no more lists are available, the resulting array will be empty.

Value List Items

Value list items allow you to add specific values to a given Radar value list, which can then be used in rules.

Related guide: [Managing List Items](#).

The value list item object

Attributes

id string

Unique identifier for the object.

value string

The value of the item.

value_list string

The identifier of the value list this item belongs to.

More attributes

✓ **object** string, value is "radar.value_list_item"

String representing the object's type. Objects of the same type share the same value.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **created_by** string

The name or email address of the user who added this item to the value list.

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

Create a value list item

Creates a new `ValueListItem` object, which is added to the specified parent value list.

Parameters

value REQUIRED

The value of the item (whose type must match the type of the parent value list).

value_list REQUIRED

The identifier of the value list which the created item will be added to.

Returns

Returns a `ValueListItem` object if creation succeeds.

Retrieve a value list item

Retrieves a `ValueListItem` object.

Parameters

No parameters.

Returns

Returns a `ValueListItem` object if a valid identifier was provided.

Delete a value list item

Deletes a `ValueListItem` object, removing it from its parent value list.

Parameters

No parameters.

Returns

Returns an object with the deleted `ValueListItem` object's ID and a `deleted` parameter on success. Otherwise, this call returns [an error](#).

List all value list items

Returns a list of `ValueListItem` objects. The objects are sorted in descending order by creation date, with the most recently created object appearing first.

Parameters

`value_list` REQUIRED

Identifier for the parent value list this item belongs to.

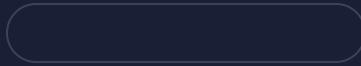
value optional

Return items belonging to the parent list whose value matches the specified value (using an “is like” match).

More parameters

✓ **created** optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` items, starting after item `starting_after`. Each entry in the array is a separate `ValueListItem` object. If no more items are available, the resulting array will be empty.

Authorizations

When an [issued card](#) is used to make a purchase, an Issuing [Authorization](#) object is created. Authorizations must be approved for the purchase to be completed successfully.

Related guide: [Issued Card Authorizations](#).

The Authorization object

Attributes

id string

Unique identifier for the object.

amount integer

The total amount that was authorized or rejected. This amount is in the card's currency and in the [smallest currency unit](#).

approved boolean

Whether the authorization has been approved.

card hash, issuing.card object

Card associated with this authorization.

cardholder string [EXPANDABLE](#)

The cardholder to whom this authorization belongs.

currency currency

Three-letter [ISO currency code](#), in lowercase. Must be a [supported currency](#).

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

status enum

The current status of the authorization in its lifecycle.

Possible enum values

`pending`

The authorization was created and is awaiting approval or was approved and is awaiting [capture](#).

closed

The authorization was declined or [captured](#) through one or more [transactions](#).

reversed

The authorization was reversed by the merchant or expired without capture.

More attributes

- > **object** string, value is "issuing.authorization"
- > **amount_details** hash
- > **authorization_method** enum
- > **balance_transactions** array, contains: [balance_transaction](#) object
- > **created** timestamp
- > **livemode** boolean
- > **merchant_amount** integer
- > **merchant_currency** currency
- > **merchant_data** hash
- > **pending_request** hash
- > **request_history** array of hashes
- > **transactions** array, contains: [issuing.transaction](#) object
- > **verification_data** hash
- > **wallet** string

Retrieve an authorization

Retrieves an Issuing [Authorization](#) object.

Parameters

No parameters.

Returns

Returns an Issuing `Authorization` object if a valid identifier was provided.

Update an authorization

Updates the specified Issuing `Authorization` object by setting the values of the parameters passed. Any parameters not provided will be left unchanged.

Parameters

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns an updated Issuing `Authorization` object if a valid identifier was provided.

Approve an authorization

Approves a pending Issuing `Authorization` object. This request should be made within the timeout window of the **real-time authorization** flow.

Parameters

amount optional

If the authorization's `pending_request.is_amount_controllable` property is `true`, you may provide this value to control how much to hold for the authorization. Must be positive (use `decline` to decline an authorization request).

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns an approved Issuing `Authorization` object.

Decline an authorization

Declines a pending Issuing `Authorization` object. This request should be made within the timeout window of the **real time authorization** flow.

Parameters

metadata optional dictionary

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

Returns

Returns a declined Issuing `Authorization` object.

List all authorizations

Returns a list of Issuing `Authorization` objects. The objects are sorted in descending order by creation date, with the most recently created object appearing first.

Parameters

`card` optional

Only return authorizations that belong to the given card.

`cardholder` optional

Only return authorizations that belong to the given cardholder.

`status` optional

Only return authorizations with the given status. One of `pending`, `closed`, or `reversed`.

More parameters

> `created` optional dictionary

> `ending_before` optional

> `limit` optional

> `starting_after` optional

Returns

A dictionary with a `data` property that contains an array of up to `limit` authorizations, starting after authorization `starting_after`. Each entry in the array is a separate Issuing `Authorization` object. If no more authorizations are available, the resulting array will be empty.

Report Runs

The Report Run object represents an instance of a report type generated with specific run parameters. Once the object is created, Stripe begins processing the report. When the report has finished running, it will give you a reference to a file where you can retrieve your results. For an overview, see [API Access to Reports](#).

Note that certain report types can only be run based on your live-mode data (not test-mode data), and will error when queried without a [live-mode API key](#).

The Report Run object

Attributes

id string

Unique identifier for the object.

parameters hash

Parameters of this report run.

report_type string

The ID of the [report type](#) to run, such as `"balance.summary.1"`.

result hash

The file object representing the result of the report run (populated when `status=succeeded`).

status string

Status of this report run. This will be `pending` when the run is initially created. When the run finishes, this will be set to `succeeded` and the `result` field will be populated. Rarely, we may encounter an error, at which point this will be set to `failed` and the `error` field will be populated.

More attributes

✓ **object** string, value is "reporting.report_run"

String representing the object's type. Objects of the same type share the same value.

✓ **created** timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

✓ **error** string

If something should go wrong during the run, a message about the failure (populated when `status=failed`).

✓ **livemode** boolean

`true` if the report is run on live mode data and `false` if it is run on test mode data.

✓ **succeeded_at** timestamp

Timestamp at which this run successfully finished (populated when `status=succeeded`).

Measured in seconds since the Unix epoch.

Create a Report Run

Creates a new object and begin running the report. (Certain report types require a [live-mode API key](#).)

Parameters

report_type REQUIRED

The ID of the [report type](#) to run, such as `"balance.summary.1"`.

parameters optional dictionary

Parameters specifying how the report should be run. Different Report Types have different required and optional parameters, listed in the [API Access to Reports](#) documentation.

Returns

Returns the new `ReportRun` object.

Retrieve a Report Run

Retrieves the details of an existing Report Run.

Parameters

No parameters.

Returns

Returns the specified `ReportRun` object if found, and returns [an error](#) otherwise.

List all Report Runs

Returns a list of Report Runs, with the most recent appearing first.

Parameters

created optional dictionary

A filter on the list based on the object `created` field. The value can be a string with an integer Unix timestamp, or it can be a dictionary with the following options:



More parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` Report Runs, starting after the argument `starting_after` if it is provided. Each entry in the array is a

separate `ReportRun` object. If no more Report Runs are available, the resulting array will be empty.

Report Types

The Report Type resource corresponds to a particular type of report, such as the "Activity summary" or "Itemized payouts" reports. These objects are identified by an ID belonging to a set of enumerated values. See [API Access to Reports documentation](#) for those Report Type IDs, along with required and optional parameters.

Note that certain report types can only be run based on your live-mode data (not test-mode data), and will error when queried without a [live-mode API key](#).

The Report Type object

Attributes

`id` string

The ID of the Report Type, such as `balance.summary.1`.

`data_available_end` timestamp

Most recent time for which this Report Type is available. Measured in seconds since the Unix epoch.

`data_available_start` timestamp

Earliest time for which this Report Type is available. Measured in seconds since the Unix epoch.

`name` string

Human-readable name of the Report Type

More attributes

✓ `object` string, value is "reporting.report_type"

String representing the object's type. Objects of the same type share the same value.

✓ **default_columns** array containing strings

List of column names that are included by default when this Report Type gets run. (If the Report Type doesn't support the `columns` parameter, this will be null.)

✓ **livemode** boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

✓ **updated** timestamp

When this Report Type was latest updated. Measured in seconds since the Unix epoch.

✓ **version** integer

Version of the Report Type. Different versions report with the same ID will have the same purpose, but may take different run parameters or have different result schemas.

Retrieve a Report Type

Retrieves the details of a Report Type. (Certain report types require a [live-mode API key](#).)

Parameters

No parameters.

Returns

Returns the specified `ReportType` object if found, and returns an error otherwise.

List all Report Types

Returns a full list of Report Types.

Parameters

No parameters.

Returns

A dictionary with a `data` property that contains an array of Report Types. Each entry is a separate `ReportType` object. This request should never return an error.

Webhook Endpoints

You can configure [webhook endpoints](#) via the API to be notified about events that happen in your Stripe account or connected accounts.

Most users configure webhooks from [the dashboard](#), which provides a user interface for registering and testing your webhook endpoints.

Related guide: [Setting up Webhooks](#).

The webhook endpoint object

Attributes

id string

Unique identifier for the object.

api_version string

The API version events are rendered as for this webhook endpoint.

description string

An optional description of what the webhook is used for.

enabled_events array containing strings

The list of events to enable for this endpoint. `['*']` indicates that all events are enabled, except those that require explicit selection.

metadata hash

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format.

secret string

The endpoint's secret, used to generate [webhook signatures](#). Only returned at creation.

status string

The status of the webhook. It can be `enabled` or `disabled`.

url string

The URL of the webhook endpoint.

More attributes

object string, value is "webhook_endpoint"

String representing the object's type. Objects of the same type share the same value.

application string

The ID of the associated Connect application.

created timestamp

Time at which the object was created. Measured in seconds since the Unix epoch.

livemode boolean

Has the value `true` if the object exists in live mode or the value `false` if the object exists in test mode.

Create a webhook endpoint

A webhook endpoint must have a `url` and a list of `enabled_events`. You may optionally specify the Boolean `connect` parameter. If set to true, then a Connect webhook endpoint that notifies the specified `url` about events from all connected accounts is created; otherwise an account webhook endpoint that notifies the specified `url` only about events from your account is created. You can also create webhook endpoints in the [webhooks settings](#) section of the Dashboard.

Parameters

enabled_events REQUIRED

The list of events to enable for this endpoint. You may specify `['*']` to enable all events, except those that require explicit selection.

account.updated

Occurs whenever an account status or property has changed.

account.application.authorized

Occurs whenever a user authorizes an application. Sent to the related application only.

account.application.deauthorized

Occurs whenever a user deauthorizes an application. Sent to the related application only.

account.external_account.created

Occurs whenever an external account is created.

account.external_account.deleted

Occurs whenever an external account is deleted.

account.external_account.updated

Occurs whenever an external account is updated.

application_fee.created

Occurs whenever an application fee is created on a charge.

[Show 182 more](#)

url REQUIRED

The URL of the webhook endpoint.

api_version optional

Events sent to this endpoint will be generated with this Stripe Version instead of your account's default Stripe Version.

description optional

An optional description of what the webhook is used for.

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to **metadata**.

More parameters

✗ **connect** optional

Whether this endpoint should receive events from connected accounts (`true`), or from your account (`false`). Defaults to `false`.

Returns

Returns the webhook endpoint object with the `secret` field populated.

Retrieve a webhook endpoint

Retrieves the webhook endpoint with the given ID.

Parameters

No parameters.

Returns

Returns a webhook endpoint if a valid webhook endpoint ID was provided. Returns [an error](#) otherwise.

Update a webhook endpoint

Updates the webhook endpoint. You may edit the `url`, the list of `enabled_events`, and the status of your endpoint.

Parameters

description optional

An optional description of what the webhook is used for.

enabled_events optional enum

The list of events to enable for this endpoint. You may specify `['*']` to enable all events, except those that require explicit selection.

Possible enum values

account.updated

Occurs whenever an account status or property has changed.

account.application.authorized

Occurs whenever a user authorizes an application. Sent to the related application only.

account.application.deauthorized

Occurs whenever a user deauthorizes an application. Sent to the related application only.

account.external_account.created

Occurs whenever an external account is created.

account.external_account.deleted

Occurs whenever an external account is deleted.

account.external_account.updated

Occurs whenever an external account is updated.

application_fee.created

Occurs whenever an application fee is created on a charge.

[Show 182 more](#)

metadata optional dictionary

Set of [key-value pairs](#) that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to

metadata.**url** optional

The URL of the webhook endpoint.

More parameters

disabled optional

Disable the webhook endpoint if set to true.

Returns

The updated webhook endpoint object if successful. Otherwise, this call returns [an error](#).

List all webhook endpoints

Returns a list of your webhook endpoints.

Parameters

✓ **ending_before** optional

A cursor for use in pagination. `ending_before` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, starting with `obj_bar`, your subsequent call can include `ending_before=obj_bar` in order to fetch the previous page of the list.

✓ **limit** optional

A limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 10.

✓ **starting_after** optional

A cursor for use in pagination. `starting_after` is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with `obj_foo`, your subsequent call can include `starting_after=obj_foo` in order to fetch the next page of the list.

Returns

A dictionary with a `data` property that contains an array of up to `limit` webhook endpoints, starting after webhook endpoint `starting_after`. Each entry in the array is a separate webhook endpoint object. If no more webhook endpoints are available, the resulting array will be empty. This request should never return an error.

Delete a webhook endpoint

You can also delete webhook endpoints via the [webhook endpoint management](#) page of the Stripe dashboard.

Parameters

No parameters.

Returns

An object with the deleted webhook endpoints's ID. Otherwise, this call returns [an error](#), such as if the webhook endpoint has already been deleted.