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 doesn't support bulk updates. You can work on only one object per request.

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

Just getting started?

Check out our development quickstart guide.

Not a developer?

Use Stripe's no-code options or apps from our partners to get started with Stripe and to do more with your Stripe account—no code required.

BASE URL

```
https://api.stripe.com
```

CLIENT LIBRARIES

Ruby Python PHP Java Node.js Go .NET

By default, the Stripe API Docs demonstrate using curl to interact with the API over HTTP. Select one of our official client libraries to see examples in code.

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_4eC39Hq...arjtT1zdp7dc"` instead of `-u sk_test_4eC39Hq...arjtT1zdp7dc`.

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

AUTHENTICATED REQUEST

cURL

```
curl https://api.stripe.com/v1/charges \
  -u sk_test_4eC39Hq...arjtT1zdp7dc:
# The colon prevents curl from asking for a password.
```

YOUR API KEY

A sample test API key is included in all the examples here, so you can test any example right away. Do not submit any personally identifiable information in requests made with this key.

To test requests using your account, replace the sample API key with your actual API key or sign in.

Connected Accounts

To act as connected accounts, clients can issue requests using the `Stripe-Account` special header. Make sure that this header contains a Stripe account ID, which usually starts with the `acct_` prefix.

The value is set per-request as shown in the adjacent code sample. Methods on the returned object reuse the same account ID.

Related guide: Making API calls for connected accounts

PER-REQUEST ACCOUNT

cURL

```
curl https://api.stripe.com/v1/charges/ch_3LmjFA2eZvKYlo2C09TLIsrw \
  -u sk_test_4eC39Hq...arjtT1zdp7dc: \
  -H "Stripe-Account: acct_1032D82eZvKYlo2C" \
  -G
```

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` enum

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

Possible enum values
`api_error`
`card_error`
`idempotency_error`
`invalid_request_error`

`code` nullable string

For some errors that could be handled programmatically, a short string indicating the error code reported.

`decline_code` nullable 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 nullable string

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

param nullable 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 nullable dictionary

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

More

Expand all

- **charge** nullable string
- **payment_method_type** nullable string
- **doc_url** nullable string
- **request_log_url** nullable string
- **setup_intent** nullable dictionary
- **source** nullable dictionary
- **payment_method** nullable dictionary

HTTP STATUS CODE SUMMARY		
200	OK	Everything worked as expected.
400	Bad Request	The request was unacceptable, often due to missing a required parameter.
401	Unauthorized	No valid API key provided.
402	Request Failed	The parameters were valid but the request failed.
403	Forbidden	The API key doesn't have permissions to perform the request.

404	Not Found	The requested resource doesn't exist.
409	Conflict	The request conflicts with another request (perhaps due to using the same idempotent key).
429	Too Many Requests	Too many requests hit the API too quickly. We recommend an exponential backoff of your requests.
500, 502, 503, 504	Server Errors	Something went wrong on Stripe's end. (These are rare.)

ERROR TYPES

`api_error`	API errors cover any other type of problem (e.g., a temporary problem with Stripe's servers), and are extremely uncommon.
`card_error`	Card errors are the most common type of error you should expect to handle. They result when the user enters a card that can't be charged for some reason.
`idempotency_error`	Idempotency errors occur when an `Idempotency-Key` is re-used on a request that does not match the first request's API endpoint and parameters.
`invalid_request_error`	Invalid request errors arise when your request has invalid parameters.

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

cURL

```
# Select a client library to see examples of
# handling different kinds of errors.
```

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. You can expand responses in two ways.

In many cases, an object contains the ID of a related object in its response properties. For example, a `Charge` might have an associated Customer ID. You can expand these objects in line with the expand request parameter. The `expandable` label in this documentation indicates ID fields that you can expand into objects.

Some available fields aren't included in the responses by default, such as the `number` and `cvc` fields for the Issuing Card object. You can request these fields as an expanded response by using the `expand` request parameter.

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

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

Expansions on list requests start with the `data` property. For example, you can expand `data.customers` on a request to list charges and associated customers. Performing deep expansions on numerous list requests might result in slower processing times.

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

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

Related guide: Expanding responses
Related video: Expand

cURL

```curl
curl https://api.stripe.com/v1/charges/ch_3LmzzQ2eZvKYlo2C0XjzUzJV \
  -u sk_test_4eC39Hq...arjtT1zdp7dc: \
  -d "expand[]"=customer \
  -d "expand[]"="invoice.subscription" \
  -G
```

RESPONSE


```
{
  "id": "ch_3LmzzQ2eZvKYlo2C0XjzUzJV",
  "object": "charge",
  "customer": {
    "id": "cu_14HOpH2eZvKYlo2CxXIM7Pb2",
    "object": "customer",
    // ...
  },
  "invoice": {
```

```
    "id": "in_1LmzzQ2eZvKYlo2CpyWn8szu",
    "object": "invoice",
    "subscription": {
      "id": "su_1LmzoG2eZvKYlo2Cpw6S7dAq",
      "object": "subscription",
      // ...
    },
    // ...
  },
  // ...
}
```


Idempotent requests

The API supports idempotency for safely retrying requests without accidentally performing the same operation twice. When creating or updating an object, use an idempotency key. Then, if a connection error occurs, you can safely repeat the request without risk of creating a second object or performing the update twice.

To perform an idempotent request, provide an additional `IdempotencyKey` element to the request options.

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 succeeds or fails. Subsequent requests with the same key return the same result, including `500` errors.

A client generates an idempotency key, which is a unique key that 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 are up to 255 characters long.

You can remove keys from the system automatically after they're at least 24 hours old. We generate a new request if a key is reused after the original is pruned. The idempotency layer compares incoming parameters to those of the original request and errors if they're the same to prevent accidental misuse.

We save results only after the execution of an endpoint begins. If incoming parameters fail validation, or the request conflicts with another request that's executing concurrently, we don't save the idempotent result because no API endpoint initiates the execution. You can retry these requests. Learn more about when you can retry idempotent requests.

All `POST` requests accept idempotency keys. Don't send idempotency keys in `GET` and `DELETE` requests because it has no effect. These requests are idempotent by definition.

cURL

```
1 curl https://api.stripe.com/v1/customers \
```

```
  -u sk_test_4eC39Hq...arjtT1zdp7dc: \
  -H "Idempotency-Key: KG5LxwFBepaKHyUD" \
  -d description="My First Test Customer (created for API docs at https://doc
```

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.

You can use metadata to store additional, structured information on an object. For example, you could store your user's full name and corresponding unique identifier from your system on a Stripe Customer object. Stripe doesn't use metadata—for example, we don't use it to authorize or decline a charge and it 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-for example, a human-readable description such as `2 shirts for test@example.com`. Unlike `metadata`, `description` is a single string, which your users might see (for example, in email receipts Stripe sends on your behalf).

Don't store any sensitive information (bank account numbers, card details, and so on) as metadata or in the `description` parameter.

Related guide: Metadata

Sample metadata use cases

- **Link IDs**: Attach your system's unique IDs to a Stripe object to simplify 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 the reason for a refund and the individual responsible for its creation.
- **Customer details**: Annotate a customer by storing an internal ID for your future use.

```
POST /v1/customers                                                cURL
```

```
curl https://api.stripe.com/v1/customers \
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:" \
  -d "metadata[order_id]"=6735
```


```
{
  "id": "cus_123456789",
  "object": "customer",
  "address": {
    "city": "city",
    "country": "US",
    "line1": "line 1",
    "line2": "line 2",
    "postal_code": "90210",
    "state": "CA"
  },
  "balance": 0,
  "created": 1483565364,
  "currency": null,
  "default_source": null,
  "delinquent": false,
  "description": null,
  "discount": null,
  "email": null,
  "invoice_prefix": "C11F7E1",
  "invoice_settings": {
    "custom_fields": null,
    "default_payment_method": null,
    "footer": null,
    "rendering_options": null
  },
  "livemode": false,
  "metadata": {
    "order_id": "6735"
  },
  "name": null,
  "next_invoice_sequence": 1,
  "phone": null,
  "preferred_locales": [],
  "shipping": null,
  "tax_exempt": "none"
}
```


Pagination

All top-level API resources have support for bulk fetches through “list” API methods. For example, you can list charges, list customers, and list invoices. These list API methods share a common structure and

accept, at a minimum, the following three parameters: `limit`, `starting_after`, and `ending_before`.

Stripe's list API methods use cursor-based pagination through the `starting_after` and `ending_before` parameters. Both parameters accept 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. You can use either the `starting_after` or `ending_before` parameter, but not both simultaneously.

Our client libraries offer auto-pagination helpers to traverse all pages of a list.

Parameters

`limit` optional, default is 10

This specifies a limit on the number of objects to return, ranging between 1 and 100.

`starting_after` optional object ID

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

`ending_before` optional object ID

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

List Response Format

`object` string, value is "list"

A string that provides a description of the object type that returns.

`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` url

The URL for accessing this list.

RESPONSE


```
{
  "object": "list",
  "url": "/v1/customers",
  "has_more": false,
  "data": [
    {
      "id": "cus_4QFJOjw2pOmAGJ",
      "object": "customer",
      "address": null,
      "balance": 0,
      "created": 1405641735,
      "currency": "usd",
      "default_source": "card_14HOpG2eZvKYlo2Cz4u5AJG5",
      "delinquent": false,
      "description": "New customer",
      "discount": null,
      "email": null,
      "invoice_prefix": "7D11B54",
      "invoice_settings": {
        "custom_fields": null,
        "default_payment_method": null,
        "footer": null,
        "rendering_options": null
      },
      "livemode": false,
      "metadata": {
        "order_id": "6735"
      },
      "name": "cus_4QFJOjw2pOmAGJ",
      "next_invoice_sequence": 25,
      "phone": null,
      "preferred_locales": [],
      "shipping": null,
      "tax_exempt": "none",
      "test_clock": null
    },
    {...},
    {...}
  ]
}
```


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 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

The total number of objects that match the query, only accurate up to 10,000. This field isn't included by default. To include it in the response, expand the `total_count` field.

RESPONSE


```
{
  "object": "search_result",
  "url": "/v1/customers/search",
  "has_more": false,
  "data": [
    {
      "id": "cus_4QFJOjw2pOmAGJ",
      "object": "customer",
      "address": null,
      "balance": 0,
      "created": 1405641735,
      "currency": "usd",
      "default_source": "card_14HOpG2eZvKYlo2Cz4u5AJG5",
      "delinquent": false,
      "description": "someone@example.com for Coderwall",
      "discount": null,
      "email": null,
      "invoice_prefix": "7D11B54",
      "invoice_settings": {
        "custom_fields": null,
        "default_payment_method": null,
        "footer": null,
        "rendering_options": null
      },
      "livemode": false,
      "metadata": {
        "foo": "bar"
      },
      "name": "fakename",
      "next_invoice_sequence": 25,
      "phone": null,
      "preferred_locales": [],
      "shipping": null,
      "tax_exempt": "none",
      "test_clock": null
    },
    {...},
    {...}
  ]
}
```

Auto-pagination

Our libraries support auto-pagination. This feature allows you to easily iterate through large lists of resources without having to manually perform the requests to fetch subsequent pages.

Since curl simply emits raw HTTP requests, it doesn't support auto-pagination.

cURL

```
# The auto-pagination feature is specific to Stripe's
# libraries and cannot be used directly with curl.
```

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.

To expedite the resolution process, provide the request identifier when you contact us about a specific request.

cURL

```
curl https://api.stripe.com/v1/customers \
  -u sk_test_4eC39Hq...arjtT1zdp7dc: \
  -D "-" \
  -X POST
```

Versioning

Each major release, such as Acacia, includes backwards-incompatible changes. Each monthly release adds new backwards-compatible changes and uses the same name as the last major release, indicating that it's safe to upgrade without adopting any backwards-incompatible changes. The current version is

2024-09-30.acacia. Learn more about API upgrades and backwards compatibility For information on all API updates, view our API changelog.

By default, requests made with curl use your Stripe account's default API version (controlled in Workbench) unless you override it by setting the `Stripe-Version` header.

Webhook events also use your account's API version by default, unless you set an API version during endpoint creation.

You can upgrade your API version in Workbench. As a precaution, use API versioning to test a new API version before committing to an upgrade.

cURL

```
curl https://api.stripe.com/v1/charges \
  -u sk_test_4eC39Hq...arjtT1zdp7dc: \
  -H "Stripe-Version: 2024-09-30.acacia"
```

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.

Related guide: Understanding Connect account balances

ENDPOINTS

`GET /v1/balance`

SHOW

Balance Transactions

Balance transactions represent funds moving through your Stripe account. Stripe creates them for every type of transaction that enters or leaves your Stripe account balance.

Related guide: Balance transaction types

ENDPOINTS

```
GET /v1/balance_transactions/:id
GET /v1/balance_transactions
```

SHOW

Charges

The `Charge` object represents a single attempt to move money into your Stripe account. PaymentIntent confirmation is the most common way to create Charges, but transferring money to a different Stripe account through Connect also creates Charges. Some legacy payment flows create Charges directly, which is not recommended for new integrations.

ENDPOINTS

```
POST /v1/charges
POST /v1/charges/:id
 GET /v1/charges/:id
 GET /v1/charges
POST /v1/charges/:id/capture
 GET /v1/charges/search
```

SHOW

Customers

This object represents a customer of your business. Use it to create recurring charges, save payment and contact information, and track payments that belong to the same customer.

ENDPOINTS

```
  POST /v1/customers
  POST /v1/customers/:id
   GET /v1/customers/:id
   GET /v1/customers
DELETE /v1/customers/:id
   GET /v1/customers/search
```

SHOW

Customer Session

A Customer Session allows you to grant Stripe's frontend SDKs (like Stripe.js) client-side access control over a Customer.

Related guides: Customer Session with the Payment Element, Customer Session with the Pricing Table, Customer Session with the Buy Button.

ENDPOINTS

```
  POST /v1/customer_sessions
```

SHOW

Disputes

A dispute occurs when a customer questions your charge with their card issuer. When this happens, you have the opportunity to respond to the dispute with evidence that shows that the charge is legitimate.

Related guide: Disputes and fraud

ENDPOINTS

```
POST /v1/disputes/:id
 GET /v1/disputes/:id
 GET /v1/disputes
POST /v1/disputes/:id/close
```

SHOW

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. Certain API requests might create multiple events. For example, if you create a new subscription for a customer, you receive both a `customer.subscription.created` event and a `charge.succeeded` event.

Events occur when the state of another API resource changes. The event's data field embeds the resource's state at the time of the change. For example, a `charge.succeeded` event contains a charge, and an `invoice.payment_failed` event contains 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. You can manage webhooks in your account settings. Learn how to listen for events so that your integration can automatically trigger reactions.

When using Connect, you can also receive event notifications that occur in connected accounts. For these events, there's an additional `account` attribute in the received `Event` object.

We only guarantee access to events through the Retrieve Event API for 30 days.

ENDPOINTS

```
GET /v1/events/:id
GET /v1/events
```

SHOW

Events v2

Learn more about calling API v2 endpoints. >

Events are generated to keep you informed of activity in your business account. APIs in the /v2 namespace generate thin events which have small, unversioned payloads that include a reference to the ID of the object that has changed. The Events v2 API returns these new thin events. Retrieve the event object for additional data about the event. Use the related object ID in the event payload to fetch the API resource of the object associated with the event. Comparatively, events generated by most API v1 include a versioned snapshot of an API object in their payload.

ENDPOINTS

```
GET /v2/core/events/:id
GET /v2/core/events
```

SHOW

Files

This object represents files hosted on Stripe's servers. You can upload files with the create file request (for example, when uploading dispute evidence). Stripe also creates files independently (for example, the results of a Sigma scheduled query).

Related guide: File upload guide

ENDPOINTS

```
POST /v1/files
 GET /v1/files/:id
 GET /v1/files
```

SHOW

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 you can use to retrieve the contents of the file without authentication.

ENDPOINTS

```
POST /v1/file_links
POST /v1/file_links/:id
 GET /v1/file_links/:id
 GET /v1/file_links
```

SHOW

Mandates

A Mandate is a record of the permission that your customer gives you to debit their payment method.

ENDPOINTS

```
 GET /v1/mandates/:id
```

SHOW

Payment Intents

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

ENDPOINTS

```
POST /v1/payment_intents
POST /v1/payment_intents/:id
 GET /v1/payment_intents/:id
 GET /v1/payment_intents
POST /v1/payment_intents/:id/cancel
POST /v1/payment_intents/:id/capture
POST /v1/payment_intents/:id/confirm
POST /v1/payment_intents/:id/increment_authorization
POST /v1/payment_intents/:id/apply_customer_balance
 GET /v1/payment_intents/search
POST /v1/payment_intents/:id/verify_microdeposits
```

SHOW

Setup Intents

A SetupIntent guides you through the process of setting up and saving a customer's payment credentials for future payments. For example, you can 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 when you're ready to collect your customer's payment credentials. Don't maintain long-lived, unconfirmed SetupIntents because they might not be valid. The SetupIntent 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 might need to be run through Strong Customer Authentication during payment method collection to streamline later off-session payments. If you use the SetupIntent with a Customer, it automatically attaches the resulting payment method to that Customer after successful setup. We recommend using SetupIntents or setup_future_usage on PaymentIntents to save payment methods to prevent saving invalid or unoptimized payment methods.

By using SetupIntents, you can reduce friction for your customers, even as regulations change over time.

Related guide: Setup Intents API

ENDPOINTS

```
POST /v1/setup_intents
POST /v1/setup_intents/:id
 GET /v1/setup_intents/:id
 GET /v1/setup_intents
POST /v1/setup_intents/:id/cancel
POST /v1/setup_intents/:id/confirm
POST /v1/setup_intents/:id/verify_microdeposits
```

SHOW

Setup Attempts

A SetupAttempt describes one attempted confirmation of a SetupIntent, whether that confirmation is successful or unsuccessful. You can use SetupAttempts to inspect details of a specific attempt at setting up a payment method using a SetupIntent.

ENDPOINTS

```
GET /v1/setup_attempts
```

SHOW

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, and list all payouts. Payouts are made on varying schedules, depending on your country and industry.

Related guide: Receiving payouts

ENDPOINTS

```
POST /v1/payouts
POST /v1/payouts/:id
 GET /v1/payouts/:id
 GET /v1/payouts
POST /v1/payouts/:id/cancel
POST /v1/payouts/:id/reverse
```

SHOW

Refunds

Refund objects allow you to refund a previously created charge that isn't refunded yet. Funds are refunded to the credit or debit card that's initially charged.

Related guide: Refunds

ENDPOINTS

```
POST /v1/refunds
POST /v1/refunds/:id
 GET /v1/refunds/:id
 GET /v1/refunds
POST /v1/refunds/:id/cancel
```

SHOW

Confirmation Token

ConfirmationTokens help transport client side data collected by Stripe JS over to your server for confirming a PaymentIntent or SetupIntent. If the confirmation is successful, values present on the ConfirmationToken are written onto the Intent.

To learn more about how to use ConfirmationToken, visit the related guides:

- Finalize payments on the server
- Build two-step confirmation.

ENDPOINTS

```
 GET /v1/confirmation_tokens/:id
POST /v1/test_helpers/confirmation_tokens
```

SHOW

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. Use our recommended payments integrations to perform this process on the client-side. This guarantees that no sensitive card data touches your server, and allows your integration to operate in a PCI-compliant way.

If you can't use client-side tokenization, you can also create tokens using the API with either your publishable or secret API key. If your integration uses this method, you're responsible for any PCI compliance that it might require, and you must keep your secret API key safe. Unlike with client-side tokenization, your customer's information isn't sent directly to Stripe, so we can't determine how it's handled or stored.

You can't store or use tokens more than once. To store card or bank account information for later use, create Customer objects or External accounts. Radar, our integrated solution for automatic fraud protection, performs best with integrations that use client-side tokenization.

ENDPOINTS

```
POST /v1/tokens
POST /v1/tokens
POST /v1/tokens
POST /v1/tokens
POST /v1/tokens
POST /v1/tokens
 GET /v1/tokens/:id
```

SHOW

Payment Methods

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.

Related guides: Payment Methods and More Payment Scenarios.

ENDPOINTS

```
POST /v1/payment_methods
POST /v1/payment_methods/:id
 GET /v1/customers/:id/payment_methods/:id
 GET /v1/payment_methods/:id
 GET /v1/customers/:id/payment_methods
 GET /v1/payment_methods
POST /v1/payment_methods/:id/attach
POST /v1/payment_methods/:id/detach
```

SHOW

Payment Method Configurations

PaymentMethodConfigurations control which payment methods are displayed to your customers when you don't explicitly specify payment method types. You can have multiple configurations with different sets of payment methods for different scenarios.

There are two types of PaymentMethodConfigurations. Which is used depends on the charge type:

Direct configurations apply to payments created on your account, including Connect destination charges, Connect separate charges and transfers, and payments not involving Connect.

Child configurations apply to payments created on your connected accounts using direct charges, and charges with the on_behalf_of parameter.

Child configurations have a `parent` that sets default values and controls which settings connected accounts may override. You can specify a parent ID at payment time, and Stripe will automatically resolve the connected account's associated child configuration. Parent configurations are managed in the dashboard and are not available in this API.

Related guides:

- Payment Method Configurations API
- Multiple configurations on dynamic payment methods

- Multiple configurations for your Connect accounts

ENDPOINTS

```
POST /v1/payment_method_configurations
POST /v1/payment_method_configurations/:id
 GET /v1/payment_method_configurations/:id
 GET /v1/payment_method_configurations
```

SHOW

Payment Method Domains

A payment method domain represents a web domain that you have registered with Stripe. Stripe Elements use registered payment method domains to control where certain payment methods are shown.

Related guide: Payment method domains.

ENDPOINTS

```
POST /v1/payment_method_domains
POST /v1/payment_method_domains/:id
 GET /v1/payment_method_domains/:id
 GET /v1/payment_method_domains
POST /v1/payment_method_domains/:id/validate
```

SHOW

Bank Accounts

These bank accounts are payment methods on `Customer` objects.

On the other hand External Accounts are transfer destinations on `Account` objects for connected accounts. They can be bank accounts or debit cards as well, and are documented in the links above.

Related guide: Bank debits and transfers

ENDPOINTS

```
  POST /v1/customers/:id/sources
  POST /v1/customers/:id/sources/:id
   GET /v1/customers/:id/bank_accounts/:id
   GET /v1/customers/:id/bank_accounts
DELETE /v1/customers/:id/sources/:id
  POST /v1/customers/:id/sources/:id/verify
```

SHOW

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.

ENDPOINTS

```
  POST /v1/customers/:id/cash_balance
   GET /v1/customers/:id/cash_balance
```

SHOW

Cash Balance Transaction

Customers with certain payments enabled have a cash balance, representing funds that were paid by the customer to a merchant, but have not yet been allocated to a payment. Cash Balance Transactions represent when funds are moved into or out of this balance. This includes funding by the customer, allocation to payments, and refunds to the customer.

ENDPOINTS

```
  GET /v1/customers/:id/cash_balance_transactions/:id
  GET /v1/customers/:id/cash_balance_transactions
 POST /v1/test_helpers/customers/:id/fund_cash_balance
```

SHOW

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

ENDPOINTS

```
  POST /v1/customers/:id/sources
  POST /v1/customers/:id/sources/:id
   GET /v1/customers/:id/cards/:id
   GET /v1/customers/:id/cards
DELETE /v1/customers/:id/sources/:id
```

SHOW

Sources Deprecated

`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.

Stripe doesn't recommend using the deprecated Sources API. We recommend that you adopt the PaymentMethods API. This newer API provides access to our latest features and payment method types.

Related guides: Sources API and Sources & Customers.

ENDPOINTS

```
  POST /v1/sources
  POST /v1/sources/:id
   GET /v1/sources/:id
  POST /v1/customers/:id/sources
DELETE /v1/customers/:id/sources/:id
```

SHOW

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

ENDPOINTS

```
  POST /v1/products
  POST /v1/products/:id
   GET /v1/products/:id
   GET /v1/products
DELETE /v1/products/:id
   GET /v1/products/search
```

SHOW

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.

ENDPOINTS

```
  POST /v1/prices
  POST /v1/prices/:id
   GET /v1/prices/:id
   GET /v1/prices
```

```
GET /v1/prices/search
```

SHOW

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 subscriptions, invoices, checkout sessions, quotes, and more. Coupons do not work with conventional one-off charges or payment intents.

ENDPOINTS

```
  POST /v1/coupons
  POST /v1/coupons/:id
   GET /v1/coupons/:id
   GET /v1/coupons
DELETE /v1/coupons/:id
```

SHOW

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.

ENDPOINTS

```
POST /v1/promotion_codes
POST /v1/promotion_codes/:id
 GET /v1/promotion_codes/:id
 GET /v1/promotion_codes
```

SHOW

Discounts

A discount represents the actual application of a coupon or promotion code. It contains information about when the discount began, when it will end, and what it is applied to.

Related guide: Applying discounts to subscriptions

ENDPOINTS

```
DELETE /v1/customers/:id/discount
DELETE /v1/subscriptions/:id/discount
```

SHOW

Tax Code

Tax codes classify goods and services for tax purposes.

ENDPOINTS

```
GET /v1/tax_codes/:id
GET /v1/tax_codes
```

SHOW

Tax Rate

Tax rates can be applied to invoices, subscriptions and Checkout Sessions to collect tax.

Related guide: Tax rates

ENDPOINTS

```
POST /v1/tax_rates
POST /v1/tax_rates/:id
 GET /v1/tax_rates/:id
 GET /v1/tax_rates
```

SHOW

Shipping Rates

Shipping rates describe the price of shipping presented to your customers and applied to a purchase. For more information, see Charge for shipping.

ENDPOINTS

```
POST /v1/shipping_rates
POST /v1/shipping_rates/:id
 GET /v1/shipping_rates/:id
 GET /v1/shipping_rates
```

SHOW

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 redirect to its URL to begin Checkout.

Related guide: Checkout quickstart

ENDPOINTS

```
POST /v1/checkout/sessions
POST /v1/checkout/sessions/:id
 GET /v1/checkout/sessions/:id
 GET /v1/checkout/sessions/:id/line_items
 GET /v1/checkout/sessions
POST /v1/checkout/sessions/:id/expire
```

SHOW

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

ENDPOINTS

```
POST /v1/payment_links
POST /v1/payment_links/:id
 GET /v1/payment_links/:id/line_items
 GET /v1/payment_links/:id
 GET /v1/payment_links
```

SHOW

Credit Note

Issue a credit note to adjust an invoice's amount after the invoice is finalized.

Related guide: Credit notes

ENDPOINTS

```
POST /v1/credit_notes
POST /v1/credit_notes/:id
 GET /v1/credit_notes/:id/lines
 GET /v1/credit_notes/preview/lines
 GET /v1/credit_notes/:id
 GET /v1/credit_notes
 GET /v1/credit_notes/preview
```

```
POST /v1/credit_notes/:id/void
```

SHOW

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

ENDPOINTS

```
POST /v1/customers/:id/balance_transactions
POST /v1/customers/:id/balance_transactions/:id
 GET /v1/customers/:id/balance_transactions/:id
 GET /v1/customers/:id/balance_transactions
```

SHOW

Customer Portal Session

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.

Related guide: Customer management

ENDPOINTS

```
POST /v1/billing_portal/sessions
```

SHOW

Customer Portal Configuration

A portal configuration describes the functionality and behavior of a portal session.

ENDPOINTS

```
POST /v1/billing_portal/configurations
POST /v1/billing_portal/configurations/:id
 GET /v1/billing_portal/configurations/:id
 GET /v1/billing_portal/configurations
```

SHOW

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

ENDPOINTS

```
  POST /v1/invoices
  POST /v1/invoices/create_preview
  POST /v1/invoices/:id
   GET /v1/invoices/:id
   GET /v1/invoices/upcoming
   GET /v1/invoices
DELETE /v1/invoices/:id
  POST /v1/invoices/:id/finalize
  POST /v1/invoices/:id/mark_uncollectible
  POST /v1/invoices/:id/pay
   GET /v1/invoices/search
  POST /v1/invoices/:id/send
  POST /v1/invoices/:id/void
```

The Invoice object

Attributes

id string

Unique identifier for the object. This property is always present unless the invoice is an upcoming invoice. See Retrieve an upcoming invoice for more details.

`auto_advance` boolean

Controls whether Stripe performs automatic collection of the invoice. If `false`, the invoice's state doesn't automatically advance without an explicit action.

`charge` nullable string Expandable

ID of the latest charge generated for this invoice, if any.

`collection_method` enum

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.

Possible enum values
`charge_automatically` Attempt payment using the default source attached to the customer.
`send_invoice` Email payment instructions to the customer.

`currency` enum

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` nullable string

An arbitrary string attached to the object. Often useful for displaying to users. Referenced as 'memo' in the Dashboard.

`hosted_invoice_url` nullable 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` dictionary

The individual line items that make up the invoice. `lines` is sorted as follows: (1) pending invoice items (including prorations) in reverse chronological order, (2) subscription items in reverse chronological order, and (3) invoice items added after invoice creation in chronological order.

+ Show child attributes

metadata nullable 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.

payment_intent nullable 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. This looks back one period for a subscription invoice. Use the line item period to get the service period for each price.

period_start timestamp

Start of the usage period during which invoice items were added to this invoice. This looks back one period for a subscription invoice. Use the line item period to get the service period for each price.

status nullable enum

The status of the invoice, one of `draft`, `open`, `paid`, `uncollectible`, or `void`. Learn more

subscription nullable string Expandable

The subscription that this invoice was prepared for, if any.

total integer

Total after discounts and taxes.

More attributes

Expand all

object string

account_country nullable string

- **account_name** nullable string
- **account_tax_ids** nullable array of strings Expandable
- **amount_due** integer
- **amount_paid** integer
- **amount_remaining** integer
- **amount_shipping** integer
- **application** nullable string Expandable Connect only
- **application_fee_amount** nullable integer Connect only
- **attempt_count** integer
- **attempted** boolean
- **automatic_tax** dictionary
- **automatically_finalizes_at** nullable timestamp
- **billing_reason** nullable enum
- **created** timestamp
- **custom_fields** nullable array of dictionaries
- **customer_address** nullable dictionary
- **customer_email** nullable string
- **customer_name** nullable string

- **customer_phone** nullable string
- **customer_shipping** nullable dictionary
- **customer_tax_exempt** nullable enum
- **customer_tax_ids** nullable array of dictionaries
- **default_payment_method** nullable string Expandable
- **default_source** nullable string Expandable
- **default_tax_rates** array of dictionaries
- **discount** nullable dictionary Deprecated
- **discounts** array of strings Expandable
- **due_date** nullable timestamp
- **effective_at** nullable timestamp
- **ending_balance** nullable integer
- **footer** nullable string
- **from_invoice** nullable dictionary
- **invoice_pdf** nullable string
- **issuer** dictionary Connect only
- **last_finalization_error** nullable dictionary
- **latest_revision** nullable string Expandable
- **livemode** boolean

- **next_payment_attempt** nullable timestamp
- **number** nullable string
- **on_behalf_of** nullable string Expandable Connect only
- **paid** boolean
- **paid_out_of_band** boolean
- **payment_settings** dictionary
- **post_payment_credit_notes_amount** integer
- **pre_payment_credit_notes_amount** integer
- **quote** nullable string Expandable
- **receipt_number** nullable string
- **rendering** nullable dictionary
- **shipping_cost** nullable dictionary
- **shipping_details** nullable dictionary
- **starting_balance** integer
- **statement_descriptor** nullable string
- **status_transitions** dictionary
- **subscription_details** nullable dictionary
- **subscription_proration_date** nullable integer

- **subtotal** integer
- **subtotal_excluding_tax** nullable integer
- **tax** nullable integer
- **test_clock** nullable string Expandable
- **threshold_reason** nullable dictionary
- **total_discount_amounts** nullable array of dictionaries
- **total_excluding_tax** nullable integer
- **total_tax_amounts** array of dictionaries
- **transfer_data** nullable dictionary Connect only
- **webhooks_delivered_at** nullable timestamp

THE INVOICE OBJECT


```
{
  "id": "in_1MtHbELkdIwHu7ixl4OzzPMv",
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": false,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "charge_automatically",
```

```
  "created": 1680644467,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZwdNtLEOXuvB",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null,
  "default_source": null,
  "default_tax_rates": [],
  "description": null,
  "discount": null,
  "discounts": [],
  "due_date": null,
  "ending_balance": null,
  "footer": null,
  "from_invoice": null,
  "hosted_invoice_url": null,
  "invoice_pdf": null,
  "issuer": {
```

Create an invoice

This endpoint creates a draft invoice for a given customer. 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` boolean

Controls whether Stripe performs automatic collection of the invoice. If `false`, the invoice's state doesn't automatically advance without an explicit action.

`collection_method` enum

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`.

Possible enum values

`charge_automatically`

`send_invoice`

`customer` string

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.

`metadata` 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` string

The ID of the subscription to invoice, if any. If set, the created invoice will only include pending invoice items for that subscription. The subscription’s billing cycle and regular subscription events won’t be affected.

More parameters

Expand all

- **`account_tax_ids`** array of strings
- **`application_fee_amount`** integer Connect only
- **`automatic_tax`** dictionary
- **`automatically_finalizes_at`** timestamp Preview feature
- **`currency`** enum
- **`custom_fields`** array of dictionaries
- **`days_until_due`** integer

- **default_payment_method** string
- **default_source** string
- **default_tax_rates** array of strings
- **discounts** array of dictionaries
- **due_date** timestamp
- **effective_at** timestamp
- **footer** string
- **from_invoice** dictionary
- **issuer** dictionary Connect only
- **number** string
- **on_behalf_of** string Connect only
- **payment_settings** dictionary
- **pending_invoice_items_behavior** enum
- **rendering** dictionary
- **shipping_cost** dictionary
- **shipping_details** dictionary
- **statement_descriptor** string
- **transfer_data** dictionary Connect only

Returns

Returns the invoice object. Returns an error if the customer ID provided is invalid.

POST /v1/invoices

cURL

```
curl https://api.stripe.com/v1/invoices \
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:" \
  -d customer=cus_NeZwdNtLEOXuvB
```

RESPONSE


```
{
  "id": "in_1MtHbELkdIwHu7ixl4OzzPMv",
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": false,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "charge_automatically",
  "created": 1680644467,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZwdNtLEOXuvB",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null,
  "default_source": null,
  "default_tax_rates": [],
```

```
  "description": null,
  "discount": null,
  "discounts": []
```

Create a preview 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 the `subscription_details.proration_date` parameter when doing the actual subscription update. 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_details.proration_date` value passed in the request.

Note: Currency conversion calculations use the latest exchange rates. Exchange rates may vary between the time of the preview and the time of the actual invoice creation. Learn more

Parameters

`customer` string

The identifier of the customer whose upcoming invoice you'd like to retrieve. If `automatic_tax` is enabled then one of `customer`, `customer_details`, `subscription`, or `schedule` must be set.

`subscription` string

The identifier of the subscription for which you'd like to retrieve the upcoming invoice. If not provided, but a `subscription_details.items` is provided, you will preview creating a subscription with those items. If neither `subscription` nor `subscription_details.items` is provided, you will retrieve the next upcoming invoice from among the customer's subscriptions.

More parameters

Expand all

- **automatic_tax** dictionary
- **coupon** string Deprecated
- **currency** enum
- **customer_details** dictionary
- **discounts** array of dictionaries
- **invoice_items** array of dictionaries
- **issuer** dictionary Connect only
- **on_behalf_of** string Connect only
- **preview_mode** enum
- **schedule** string
- **schedule_details** dictionary
- **subscription_details** dictionary

Returns

Returns an invoice if valid customer information is provided. Returns an error otherwise.

POST /v1/invoices/create_preview

cURL

```
curl https://api.stripe.com/v1/invoices/create_preview \
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:" \
  -d customer=cus_NeZwdNtLEOXuvB
```

RESPONSE


```
{
  "id": "upcoming_in_1MtHbELkdIwHu7ixl4OzzPMv",
  "object": "invoice",
  "account_country": "US",
```

```
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": false,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "charge_automatically",
  "created": 1680644467,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZwdNtLEOXuvB",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null,
  "default_source": null,
  "default_tax_rates": [],
  "description": null,
  "discount": null,
  "discounts": [],
  "due_date": 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 boolean

Controls whether Stripe performs automatic collection of the invoice.

collection_method enum

Either `charge_automatically` or `send_invoice`. This field can be updated only on `draft` invoices.

Possible enum values
`charge_automatically`
`send_invoice`

description string

An arbitrary string attached to the object. Often useful for displaying to users. Referenced as 'memo' in the Dashboard.

metadata 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

Expand all

- **account_tax_ids** array of strings
- **application_fee_amount** integer Connect only
- **automatic_tax** dictionary
- **automatically_finalizes_at** timestamp Preview feature
- **custom_fields** array of dictionaries
- **days_until_due** integer
- **default_payment_method** string

- **default_source** string
- **default_tax_rates** array of strings
- **discounts** array of dictionaries
- **due_date** timestamp
- **effective_at** timestamp
- **footer** string
- **issuer** dictionary Connect only
- **number** string
- **on_behalf_of** string Connect only
- **payment_settings** dictionary
- **rendering** dictionary
- **shipping_cost** dictionary
- **shipping_details** dictionary
- **statement_descriptor** string
- **transfer_data** dictionary Connect only

Returns

Returns the invoice object.

POST /v1/invoices/:id

cURL

```
curl https://api.stripe.com/v1/invoices/in_1MtHbELkdIwHu7ixl4OzzPMv \
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:" \
```

```
3    -d "metadata[order_id]"=6735
```

RESPONSE


```
{
  "id": "in_1MtHbELkdIwHu7ixl4OzzPMv",
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": false,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "charge_automatically",
  "created": 1680644467,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZwdNtLEOXuvB",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null,
  "default_source": null,
  "default_tax_rates": [],
  "description": null,
  "discount": null,
  "discounts": [],
```


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.

```
GET /v1/invoices/:id                                                cURL
1 curl https://api.stripe.com/v1/invoices/in_1MtHbELkdIwHu7ixl4OzzPMv \
2   -u "sk_test_4eC39Hq...arjtT1zdp7dc:"
```

RESPONSE


```
{
  "id": "in_1MtHbELkdIwHu7ixl4OzzPMv",
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
```

```
  "attempted": false,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "charge_automatically",
  "created": 1680644467,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZwdNtLEOXuvB",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null,
  "default_source": null,
  "default_tax_rates": [],
  "description": null,
  "discount": null,
  "discounts": [],
```


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 the `subscription_details.proration_date` parameter when doing the actual subscription update. 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_details.proration_date` value passed in the request.

Note: Currency conversion calculations use the latest exchange rates. Exchange rates may vary between the time of the preview and the time of the actual invoice creation. Learn more

Parameters

customer string

The identifier of the customer whose upcoming invoice you'd like to retrieve. If `automatic_tax` is enabled then one of `customer`, `customer_details`, `subscription`, or `schedule` must be set.

subscription string

The identifier of the subscription for which you'd like to retrieve the upcoming invoice. If not provided, but a `subscription_details.items` is provided, you will preview creating a subscription with those items. If neither `subscription` nor `subscription_details.items` is provided, you will retrieve the next upcoming invoice from among the customer's subscriptions.

More parameters

Expand all

- **automatic_tax** dictionary
- **coupon** string Deprecated
- **currency** enum
- **customer_details** dictionary
- **discounts** array of dictionaries
- **invoice_items** array of dictionaries
- **issuer** dictionary Connect only
- **on_behalf_of** string Connect only
- **preview_mode** enum
- **schedule** string

- **schedule_details** dictionary
- **subscription_billing_cycle_anchor** string | timestamp Deprecated
- **subscription_cancel_at** timestamp Deprecated
- **subscription_cancel_at_period_end** boolean Deprecated
- **subscription_cancel_now** boolean Deprecated
- **subscription_default_tax_rates** array of strings Deprecated
- **subscription_details** dictionary
- **subscription_items** array of dictionaries Deprecated
- **subscription_proration_behavior** enum Deprecated
- **subscription_proration_date** timestamp Deprecated
- **subscription_resume_at** string Deprecated
- **subscription_start_date** timestamp Deprecated
- **subscription_trial_end** string | timestamp Deprecated

Returns

Returns an invoice if valid customer information is provided. Returns an error otherwise.

GET /v1/invoices/upcoming

cURL

```
curl -G https://api.stripe.com/v1/invoices/upcoming \
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:" \
  -d customer=cus_NeZwdNtLEOXuvB
```

RESPONSE

```
{
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": false,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "charge_automatically",
  "created": 1680644467,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZwdNtLEOXuvB",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null,
  "default_source": null,
  "default_tax_rates": [],
  "description": null,
  "discount": null,
  "discounts": [],
```

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 string

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

status enum

The status of the invoice, one of `draft`, `open`, `paid`, `uncollectible`, or `void`. Learn more

subscription string

Only return invoices for the subscription specified by this subscription ID.

More parameters

Expand all

> **collection_method** enum

> **created** dictionary

> **ending_before** string

> **limit** integer

> **starting_after** string

Returns

A dictionary with a `data` property that contains an array invoice attachments,

GET /v1/invoices

cURL

```
curl -G https://api.stripe.com/v1/invoices \
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:" \
  -d limit=3
```

RESPONSE


```
{
  "object": "list",
  "url": "/v1/invoices",
  "has_more": false,
  "data": [
    {
      "id": "in_1MtHbELkdIwHu7ixl4OzzPMv",
```

```
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": false,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "charge_automatically",
  "created": 1680644467,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZwdNtLEOXuvB",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null
```


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.

```
DELETE /v1/invoices/:id                                   cURL
1 curl -X DELETE https://api.stripe.com/v1/invoices/in_1MtHbELkdIwHu7ixl4OzzPMv
2   -u "sk_test_4eC39Hq...arjtT1zdp7dc:"
```

RESPONSE

```
{
  "id": "in_1MtHbELkdIwHu7ixl4OzzPMv",
  "object": "invoice",
  "deleted": true
}
```

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 boolean

Controls whether Stripe performs automatic collection of the invoice. If `false`, the invoice's state doesn't automatically advance without an explicit action.

Returns

Returns an invoice object with `status=open`.

```
POST /v1/invoices/:id/finalize                            cURL
1 curl -X POST https://api.stripe.com/v1/invoices/in_1MtGmCLkdIwHu7ix6PgS6g8S/f
2   -u "sk_test_4eC39Hq...arjtT1zdp7dc:"
```

RESPONSE

```
{
  "id": "in_1MtGmCLkdIwHu7ix6PgS6g8S",
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": true,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "send_invoice",
  "created": 1680641304,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZw0zvTyquTfF",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null,
  "default_source": null,
  "default_tax_rates": [],
  "description": null,
  "discount": null,
  "discounts": [],
```


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.

```
POST /v1/invoices/:id/mark_uncollectible                    cURL
```

```
curl -X POST https://api.stripe.com/v1/invoices/in_1MtG0nLkdIwHu7ixAaUw3Cb4/m
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:"
```

RESPONSE


```
{
  "id": "in_1MtG0nLkdIwHu7ixAaUw3Cb4",
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 599,
  "amount_paid": 0,
  "amount_remaining": 599,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": false,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "charge_automatically",
  "created": 1680638365,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZw0zvTyquTfF",
  "customer_address": null,
```

```
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [
    {
      "type": "eu_vat",
      "value": "DE123456789"
    },
    {
      "type": "eu_vat",
```


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

No parameters.

More parameters

Expand all

- **forgive** boolean
- **mandate** string
- **off_session** boolean
- **paid_out_of_band** boolean
- **payment_method** string
- **source** string

Returns

Returns the invoice object.

```
POST /v1/invoices/:id/pay                                    cURL
```

```
curl -X POST https://api.stripe.com/v1/invoices/in_1MtGmCLkdIwHu7ix6PgS6g8S/p
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:"
```

RESPONSE


```
{
  "id": "in_1MtGmCLkdIwHu7ix6PgS6g8S",
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": true,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "send_invoice",
  "created": 1680641304,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZw0zvTyquTfF",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null,
  "default_source": null,
  "default_tax_rates": [],
  "description": null,
  "discount": null,
  "discounts": [],
  "due_date": 1681246104
```

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 string Required

The search query string. See search query language and the list of supported query fields for invoices.

limit integer

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

page string

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.

GET /v1/invoices/search

cURL

```
curl -G https://api.stripe.com/v1/invoices/search \
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:" \
  -d query="total<1"
```

RESPONSE

```
{
  "object": "search_result",
  "url": "/v1/invoices/search",
  "has_more": false,
  "data": [
```

```
{
  "id": "in_1MtHbELkdIwHu7ixl4OzzPMv",
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": false,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "charge_automatically",
  "created": 1680644467,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZwdNtLEOXuvB",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null
```


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.

```
POST /v1/invoices/:id/send                                   cURL
```

```
curl -X POST https://api.stripe.com/v1/invoices/in_1MtGmCLkdIwHu7ixJlveR2DO/s
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:"
```

RESPONSE


```
{
  "id": "in_1MtGmCLkdIwHu7ixJlveR2DO",
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
  "attempted": true,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "send_invoice",
  "created": 1680641304,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZwvqcz9Sh2uw",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null,
  "default_source": null,
```

```
  "default_tax_rates": [],
  "description": null,
  "discount": null,
  "discounts": [],
  "due_date": 1681246104
```


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.

Consult with local regulations to determine whether and how an invoice might be amended, canceled, or voided in the jurisdiction you're doing business in. You might need to issue another invoice or credit note instead. Stripe recommends that you consult with your legal counsel for advice specific to your business.

Parameters

No parameters.

Returns

Returns the voided invoice object.

`POST /v1/invoices/:id/void` cURL

```
curl -X POST https://api.stripe.com/v1/invoices/in_1MtGmCLkdIwHu7ix6PgS6g8S/v
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:"
```

RESPONSE


```
{
  "id": "in_1MtGmCLkdIwHu7ix6PgS6g8S",
  "object": "invoice",
  "account_country": "US",
  "account_name": "Stripe Docs",
  "account_tax_ids": null,
  "amount_due": 0,
  "amount_paid": 0,
  "amount_remaining": 0,
  "amount_shipping": 0,
  "application": null,
  "application_fee_amount": null,
  "attempt_count": 0,
```

```
  "attempted": false,
  "auto_advance": false,
  "automatic_tax": {
    "enabled": false,
    "liability": null,
    "status": null
  },
  "billing_reason": "manual",
  "charge": null,
  "collection_method": "charge_automatically",
  "created": 1680644467,
  "currency": "usd",
  "custom_fields": null,
  "customer": "cus_NeZwdNtLEOXuvB",
  "customer_address": null,
  "customer_email": "jennyrosen@example.com",
  "customer_name": "Jenny Rosen",
  "customer_phone": null,
  "customer_shipping": null,
  "customer_tax_exempt": "none",
  "customer_tax_ids": [],
  "default_payment_method": null,
  "default_source": null,
  "default_tax_rates": [],
  "description": null,
  "discount": null,
  "discounts": [],
  "due_date": null,
```


Invoice Items

Invoice Items represent the component lines of an invoice. An invoice item is added to an invoice by creating or updating it with an `invoice` field, at which point it will be included as an invoice line item within invoice.lines.

Invoice Items can be created before you are ready to actually send the invoice. This can be particularly useful when combined with a subscription. 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 guides: Integrate with the Invoicing API, Subscription Invoices.

ENDPOINTS

```
POST /v1/invoiceitems
POST /v1/invoiceitems/:id
```

```
   GET /v1/invoiceitems/:id
   GET /v1/invoiceitems
DELETE /v1/invoiceitems/:id
```

SHOW

Invoice Line Item

Invoice Line Items represent the individual lines within an invoice and only exist within the context of an invoice.

Each line item is backed by either an invoice item or a subscription item.

ENDPOINTS

```
  POST /v1/invoices/:id/lines/:id
   GET /v1/invoices/:id/lines
   GET /v1/invoices/upcoming/lines
  POST /v1/invoices/:id/add_lines
  POST /v1/invoices/:id/remove_lines
  POST /v1/invoices/:id/update_lines
```

SHOW

Invoice Rendering Templates

Invoice Rendering Templates are used to configure how invoices are rendered on surfaces like the PDF. Invoice Rendering Templates can be created from within the Dashboard, and they can be used over the API when creating invoices.

ENDPOINTS

```
 GET /v1/invoice_rendering_templates/:id
 GET /v1/invoice_rendering_templates
POST /v1/invoice_rendering_templates/:id/archive
POST /v1/invoice_rendering_templates/:id/unarchive
```

SHOW

Alerts

A billing alert is a resource that notifies you when a certain usage threshold on a meter is crossed. For example, you might create a billing alert to notify you when a certain user made 100 API requests.

ENDPOINTS

```
POST /v1/billing/alerts
 GET /v1/billing/alerts/:id
 GET /v1/billing/alerts
POST /v1/billing/alerts/:id/activate
POST /v1/billing/alerts/:id/archive
POST /v1/billing/alerts/:id/deactivate
```

SHOW

Meters

A billing meter is a resource that allows you to track usage of a particular event. For example, you might create a billing meter to track the number of API calls made by a particular user. You can then attach the billing meter to a price and attach the price to a subscription to charge the user for the number of API calls they make.

Related guide: Usage based billing

ENDPOINTS

```
POST /v1/billing/meters
POST /v1/billing/meters/:id
 GET /v1/billing/meters/:id
 GET /v1/billing/meters
POST /v1/billing/meters/:id/deactivate
POST /v1/billing/meters/:id/reactivate
```

SHOW

Meter Events

A billing meter event represents a customer's usage of a product. Meter events are used to bill a customer based on their usage. Meter events are associated with billing meters, which define the shape of the event's payload and how those events are aggregated for billing.

ENDPOINTS

```
POST /v1/billing/meter_events
```

SHOW

Meter Events v2

Learn more about calling API v2 endpoints. >

Meter events are used to report customer usage of your product or service. Meter events are associated with billing meters, which define the shape of the event's payload and how those events are aggregated. Meter events are processed asynchronously, so they may not be immediately reflected in aggregates or on upcoming invoices.

ENDPOINTS

`POST /v2/billing/meter_events`

SHOW

Meter Event Adjustment

A billing meter event adjustment is a resource that allows you to cancel a meter event. For example, you might create a billing meter event adjustment to cancel a meter event that was created in error or attached to the wrong customer.

ENDPOINTS

`POST /v1/billing/meter_event_adjustments`

SHOW

Meter Event Adjustment v2

Learn more about calling API v2 endpoints. >

A billing meter event adjustment is a resource that allows you to cancel a meter event. For example, you might create a billing meter event adjustment to cancel a meter event that was created in error or attached to the wrong customer.

ENDPOINTS

`POST /v2/billing/meter_event_adjustments`

SHOW

Meter Event Stream v2

Learn more about calling API v2 endpoints. >

You can send a higher-throughput of meter events using meter event streams. For this flow, you must first create a meter event session, which will provide you with a session token. You can then create meter events through the meter event stream endpoint, using the session token for authentication. The session tokens are short-lived and you will need to create a new meter event session when the token expires.

ENDPOINTS

```
POST /v2/billing/meter_event_session
POST /v2/billing/meter_event_stream
```

SHOW

Meter Event Summary

A billing meter event summary represents an aggregated view of a customer's billing meter events within a specified timeframe. It indicates how much usage was accrued by a customer for that period.

ENDPOINTS

```
GET /v1/billing/meters/:id/event_summaries
```

SHOW

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.

Related guides: Set up a subscription and more about products and prices.

ENDPOINTS

```
  POST /v1/plans
  POST /v1/plans/:id
   GET /v1/plans/:id
   GET /v1/plans
DELETE /v1/plans/:id
```

SHOW

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.

ENDPOINTS

```
  POST /v1/quotes
  POST /v1/quotes/:id
   GET /v1/quotes/:id/line_items
   GET /v1/quotes/:id/computed_upfront_line_items
```

```
 GET /v1/quotes/:id
 GET /v1/quotes
POST /v1/quotes/:id/accept
POST /v1/quotes/:id/cancel
 GET /v1/quotes/:id/pdf
POST /v1/quotes/:id/finalize
```

SHOW

Subscriptions

Subscriptions allow you to charge a customer on a recurring basis.

Related guide: Creating subscriptions

ENDPOINTS

```
  POST /v1/subscriptions
  POST /v1/subscriptions/:id
   GET /v1/subscriptions/:id
   GET /v1/subscriptions
DELETE /v1/subscriptions/:id
  POST /v1/subscriptions/:id/resume
   GET /v1/subscriptions/search
```

SHOW

Subscription Items

Subscription items allow you to create customer subscriptions with more than one plan, making it easy to represent complex billing relationships.

ENDPOINTS

```
  POST /v1/subscription_items
  POST /v1/subscription_items/:id
   GET /v1/subscription_items/:id
   GET /v1/subscription_items
DELETE /v1/subscription_items/:id
```

SHOW

Subscription Schedule

A subscription schedule allows you to create and manage the lifecycle of a subscription by predefining expected changes.

Related guide: Subscription schedules

ENDPOINTS

```
  POST /v1/subscription_schedules
  POST /v1/subscription_schedules/:id
   GET /v1/subscription_schedules/:id
   GET /v1/subscription_schedules
  POST /v1/subscription_schedules/:id/cancel
  POST /v1/subscription_schedules/:id/release
```

SHOW

Tax IDs

You can add one or multiple tax IDs to a customer or account. Customer and account tax IDs get displayed on related invoices and credit notes.

Related guides: Customer tax identification numbers, Account tax IDs

ENDPOINTS

```
  POST /v1/customers/:id/tax_ids
  POST /v1/tax_ids
   GET /v1/customers/:id/tax_ids/:id
   GET /v1/tax_ids/:id
   GET /v1/customers/:id/tax_ids
   GET /v1/tax_ids
DELETE /v1/customers/:id/tax_ids/:id
DELETE /v1/tax_ids/:id
```

SHOW

Test Clocks Test helper

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.

ENDPOINTS

```
  POST /v1/test_helpers/test_clocks
   GET /v1/test_helpers/test_clocks/:id
   GET /v1/test_helpers/test_clocks
DELETE /v1/test_helpers/test_clocks/:id
  POST /v1/test_helpers/test_clocks/:id/advance
```

SHOW

Usage Records

Usage records allow you to report customer usage and metrics to Stripe for metered billing of subscription prices.

Related guide: Metered billing

This is our legacy usage-based billing API. See the updated usage-based billing docs.

ENDPOINTS

```
POST /v1/subscription_items/:id/usage_records
```

SHOW

Usage Record Summary

A usage record summary represents an aggregated view of how much usage was accrued for a subscription item within a subscription billing period.

ENDPOINTS

```
GET /v1/subscription_items/:id/usage_record_summaries
```

SHOW

Accounts

This is an object representing a Stripe account. You can retrieve it to see properties on the account like its current requirements or if the account is enabled to make live charges or receive payouts.

For accounts where controller.requirement_collection is `application`, which includes Custom accounts, the properties below are always returned.

For accounts where controller.requirement_collection is `stripe`, which includes Standard and Express accounts, some properties are only returned until you create an Account Link or Account Session to start Connect Onboarding. Learn about the differences between accounts.

ENDPOINTS

```
  POST /v1/accounts
  POST /v1/accounts/:id
   GET /v1/accounts/:id
   GET /v1/accounts
DELETE /v1/accounts/:id
  POST /v1/accounts/:id/reject
```

SHOW

Login Links

Login Links are single-use URLs for a connected account to access the Express Dashboard. The connected account's account.controller.stripe_dashboard.type must be `express` to have access to the Express Dashboard.

ENDPOINTS

```
  POST /v1/accounts/:id/login_links
```

SHOW

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

ENDPOINTS
`POST /v1/account_links`

SHOW

Account Session

An AccountSession allows a Connect platform to grant access to a connected account in Connect embedded components.

We recommend that you create an AccountSession each time you need to display an embedded component to your user. Do not save AccountSessions to your database as they expire relatively quickly, and cannot be used more than once.

Related guide: Connect embedded components

ENDPOINTS
`POST /v1/account_sessions`

SHOW

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

ENDPOINTS

```
GET /v1/application_fees/:id
GET /v1/application_fees
```

SHOW

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

ENDPOINTS

```
POST /v1/application_fees/:id/refunds
POST /v1/application_fees/:id/refunds/:id
 GET /v1/application_fees/:id/refunds/:id
 GET /v1/application_fees/:id/refunds
```

SHOW

Capabilities

This is an object representing a capability for a Stripe account.

Related guide: Account capabilities

ENDPOINTS

```
POST /v1/accounts/:id/capabilities/:id
 GET /v1/accounts/:id/capabilities/:id
 GET /v1/accounts/:id/capabilities
```

SHOW

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.

ENDPOINTS

```
GET /v1/country_specs/:id
GET /v1/country_specs
```

SHOW

External Bank Accounts

External bank accounts are financial accounts associated with a Stripe platform's connected accounts for the purpose of transferring funds to or from the connected account's Stripe balance.

ENDPOINTS

```
  POST /v1/accounts/:id/external_accounts
  POST /v1/accounts/:id/external_accounts/:id
   GET /v1/accounts/:id/external_accounts/:id
   GET /v1/accounts/:id/external_accounts
DELETE /v1/accounts/:id/external_accounts/:id
```

SHOW

External Account Cards

External account cards are debit cards associated with a Stripe platform's connected accounts for the purpose of transferring funds to or from the connected accounts Stripe balance.

ENDPOINTS

```
  POST /v1/accounts/:id/external_accounts
  POST /v1/accounts/:id/external_accounts/:id
   GET /v1/accounts/:id/external_accounts/:id
   GET /v1/accounts/:id/external_accounts
DELETE /v1/accounts/:id/external_accounts/:id
```

SHOW

Person

This is an object representing a person associated with a Stripe account.

A platform cannot access a person for an account where account.controller.requirement_collection is `stripe`, which includes Standard and Express accounts, after creating an Account Link or Account Session to start Connect onboarding.

See the Standard onboarding or Express onboarding documentation for information about prefilling information and account onboarding steps. Learn more about handling identity verification with the API.

ENDPOINTS

```
  POST /v1/accounts/:id/persons
  POST /v1/accounts/:id/persons/:id
   GET /v1/accounts/:id/persons/:id
   GET /v1/accounts/:id/persons
DELETE /v1/accounts/:id/persons/:id
```

SHOW

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

ENDPOINTS

```
  POST /v1/topups
  POST /v1/topups/:id
   GET /v1/topups/:id
   GET /v1/topups
  POST /v1/topups/:id/cancel
```

SHOW

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

ENDPOINTS

```
POST /v1/transfers
POST /v1/transfers/:id
 GET /v1/transfers/:id
 GET /v1/transfers
```

SHOW

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: Reverse transfers

ENDPOINTS

```
POST /v1/transfers/:id/reversals
POST /v1/transfers/:id/reversals/:id
 GET /v1/transfers/:id/reversals/:id
```

```
GET /v1/transfers/:id/reversals
```

SHOW

Secrets

Secret Store is an API that allows Stripe Apps developers to securely persist secrets for use by UI Extensions and app backends.

The primary resource in Secret Store is a `secret`. Other apps can't view secrets created by an app. Additionally, secrets are scoped to provide further permission control.

All Dashboard users and the app backend share `account` scoped secrets. Use the `account` scope for secrets that don't change per-user, like a third-party API key.

A `user` scoped secret is accessible by the app backend and one specific Dashboard user. Use the `user` scope for per-user secrets like per-user OAuth tokens, where different users might have different permissions.

Related guide: Store data between page reloads

ENDPOINTS

```
 GET /v1/apps/secrets
POST /v1/apps/secrets/delete
 GET /v1/apps/secrets/find
POST /v1/apps/secrets
```

SHOW

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

ENDPOINTS

```
GET /v1/radar/early_fraud_warnings/:id
GET /v1/radar/early_fraud_warnings
```

SHOW

Reviews

Reviews can be used to supplement automated fraud detection with human expertise.

Learn more about Radar and reviewing payments here.

ENDPOINTS

```
GET /v1/reviews/:id
GET /v1/reviews
POST /v1/reviews/:id/approve
```

SHOW

Value Lists

Value lists allow you to group values together which can then be referenced in rules.

Related guide: Default Stripe lists

ENDPOINTS

```
POST /v1/radar/value_lists
```

```
  POST /v1/radar/value_lists/:id
   GET /v1/radar/value_lists/:id
   GET /v1/radar/value_lists
DELETE /v1/radar/value_lists/:id
```

SHOW

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

ENDPOINTS

```
  POST /v1/radar/value_list_items
   GET /v1/radar/value_list_items/:id
   GET /v1/radar/value_list_items
DELETE /v1/radar/value_list_items/:id
```

SHOW

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

ENDPOINTS

```
POST /v1/issuing/authorizations/:id
 GET /v1/issuing/authorizations/:id
 GET /v1/issuing/authorizations
POST /v1/issuing/authorizations/:id/approve
POST /v1/issuing/authorizations/:id/decline
POST /v1/test_helpers/issuing/authorizations
POST /v1/test_helpers/issuing/authorizations/:id/capture
POST /v1/test_helpers/issuing/authorizations/:id/expire
POST /v1/test_helpers/issuing/authorizations/:id/finalize_amount
POST /v1/test_helpers/issuing/authorizations/:id/increment
POST /v1/test_helpers/issuing/authorizations/:id/fraud_challenges/respond
POST /v1/test_helpers/issuing/authorizations/:id/reverse
```

SHOW

Cardholders

An Issuing `Cardholder` object represents an individual or business entity who is issued cards.

Related guide: How to create a cardholder

ENDPOINTS

```
POST /v1/issuing/cardholders
POST /v1/issuing/cardholders/:id
 GET /v1/issuing/cardholders/:id
 GET /v1/issuing/cardholders
```

SHOW

Cards

You can create physical or virtual cards that are issued to cardholders.

ENDPOINTS

```
POST /v1/issuing/cards
POST /v1/issuing/cards/:id
 GET /v1/issuing/cards/:id
 GET /v1/issuing/cards
POST /v1/test_helpers/issuing/cards/:id/shipping/deliver
POST /v1/test_helpers/issuing/cards/:id/shipping/fail
POST /v1/test_helpers/issuing/cards/:id/shipping/return
POST /v1/test_helpers/issuing/cards/:id/shipping/ship
```

SHOW

Disputes

As a card issuer, you can dispute transactions that the cardholder does not recognize, suspects to be fraudulent, or has other issues with.

Related guide: Issuing disputes

ENDPOINTS

```
POST /v1/issuing/disputes
POST /v1/issuing/disputes/:id
 GET /v1/issuing/disputes/:id
 GET /v1/issuing/disputes
POST /v1/issuing/disputes/:id/submit
```

SHOW

Funding Instructions

Funding Instructions contain reusable bank account and routing information. Push funds to these addresses via bank transfer to top up Issuing Balances.

ENDPOINTS

```
POST /v1/issuing/funding_instructions
 GET /v1/issuing/funding_instructions
POST /v1/test_helpers/issuing/fund_balance
```

SHOW

Personalization Designs

A Personalization Design is a logical grouping of a Physical Bundle, card logo, and carrier text that represents a product line.

ENDPOINTS

```
POST /v1/issuing/personalization_designs
POST /v1/issuing/personalization_designs/:id
 GET /v1/issuing/personalization_designs/:id
 GET /v1/issuing/personalization_designs
POST /v1/test_helpers/issuing/personalization_designs/:id/activate
POST /v1/test_helpers/issuing/personalization_designs/:id/deactivate
POST /v1/test_helpers/issuing/personalization_designs/:id/reject
```

SHOW

Physical Bundles

A Physical Bundle represents the bundle of physical items - card stock, carrier letter, and envelope - that is shipped to a cardholder when you create a physical card.

ENDPOINTS

```
GET /v1/issuing/physical_bundles/:id
GET /v1/issuing/physical_bundles
```

SHOW

Tokens Preview feature

An issuing token object is created when an issued card is added to a digital wallet. As a card issuer, you can view and manage these tokens through Stripe.

ENDPOINTS

```
POST /v1/issuing/tokens/:id
 GET /v1/issuing/tokens/:id
 GET /v1/issuing/tokens
```

SHOW

Transactions

Any use of an issued card that results in funds entering or leaving your Stripe account, such as a completed purchase or refund, is represented by an Issuing `Transaction` object.

Related guide: Issued card transactions

ENDPOINTS

```
POST /v1/issuing/transactions/:id
 GET /v1/issuing/transactions/:id
 GET /v1/issuing/transactions
POST /v1/test_helpers/issuing/transactions/create_force_capture
POST /v1/test_helpers/issuing/transactions/create_unlinked_refund
POST /v1/test_helpers/issuing/transactions/:id/refund
```

SHOW

Connection Token

A Connection Token is used by the Stripe Terminal SDK to connect to a reader.

Related guide: Fleet management

ENDPOINTS

```
POST /v1/terminal/connection_tokens
```

SHOW

Location

A Location represents a grouping of readers.

Related guide: Fleet management

ENDPOINTS

```
  POST /v1/terminal/locations
  POST /v1/terminal/locations/:id
   GET /v1/terminal/locations/:id
   GET /v1/terminal/locations
DELETE /v1/terminal/locations/:id
```

SHOW

Reader

A Reader represents a physical device for accepting payment details.

Related guide: Connecting to a reader

ENDPOINTS

```
  POST /v1/terminal/readers
  POST /v1/terminal/readers/:id
   GET /v1/terminal/readers/:id
   GET /v1/terminal/readers
DELETE /v1/terminal/readers/:id
  POST /v1/terminal/readers/:id/cancel_action
  POST /v1/terminal/readers/:id/collect_inputs
  POST /v1/terminal/readers/:id/confirm_payment_intent
  POST /v1/terminal/readers/:id/collect_payment_method
  POST /v1/terminal/readers/:id/process_payment_intent
  POST /v1/terminal/readers/:id/process_setup_intent
  POST /v1/terminal/readers/:id/refund_payment
  POST /v1/terminal/readers/:id/set_reader_display
  POST /v1/test_helpers/terminal/readers/:id/present_payment_method
```

SHOW

Terminal Hardware Order

Preview feature

A TerminalHardwareOrder represents an order for Terminal hardware, containing information such as the price, shipping information and the items ordered.

ENDPOINTS

```
POST /v1/terminal/hardware_orders
 GET /v1/terminal/hardware_orders/:id
 GET /v1/terminal/hardware_orders
POST /v1/terminal/hardware_orders/:id/cancel
 GET /v1/terminal/hardware_orders/preview
POST /v1/test_helpers/terminal/hardware_orders/:id/mark_ready_to_ship
POST /v1/test_helpers/terminal/hardware_orders/:id/deliver
POST /v1/test_helpers/terminal/hardware_orders/:id/ship
POST /v1/test_helpers/terminal/hardware_orders/:id/mark_undeliverable
```

SHOW

Terminal Hardware Product

Preview feature

A TerminalHardwareProduct is a category of hardware devices that are generally similar, but may have variations depending on the country it's shipped to.

TerminalHardwareSKUs represent variations within the same Product (for example, a country specific device). For example, WisePOS E is a TerminalHardwareProduct and a WisePOS E - US and WisePOS E - UK are TerminalHardwareSKUs.

ENDPOINTS

```
 GET /v1/terminal/hardware_products/:id
 GET /v1/terminal/hardware_products
```

SHOW

Terminal Hardware SKU

Preview feature

A TerminalHardwareSKU represents a SKU for Terminal hardware. A SKU is a representation of a product available for purchase, containing information such as the name, price, and images.

ENDPOINTS

```
GET /v1/terminal/hardware_skus/:id
GET /v1/terminal/hardware_skus
```

SHOW

Terminal Hardware Shipping Method

Preview feature

A TerminalHardwareShipping represents a Shipping Method for Terminal hardware. A Shipping Method is a country-specific representation of a way to ship hardware, containing information such as the country, name, and expected delivery date.

ENDPOINTS

```
GET /v1/terminal/hardware_shipping_methods/:id
GET /v1/terminal/hardware_shipping_methods
```

SHOW

Configuration

A Configurations object represents how features should be configured for terminal readers.

ENDPOINTS

```
  POST /v1/terminal/configurations
  POST /v1/terminal/configurations/:id
   GET /v1/terminal/configurations/:id
   GET /v1/terminal/configurations
DELETE /v1/terminal/configurations/:id
```

SHOW

Financial Accounts

Stripe Treasury provides users with a container for money called a FinancialAccount that is separate from their Payments balance. FinancialAccounts serve as the source and destination of Treasury's money movement APIs.

ENDPOINTS

```
  POST /v1/treasury/financial_accounts
  POST /v1/treasury/financial_accounts/:id
   GET /v1/treasury/financial_accounts/:id
   GET /v1/treasury/financial_accounts
```

SHOW

Financial Account Features

Encodes whether a FinancialAccount has access to a particular Feature, with a `status` enum and associated `status_details`. Stripe or the platform can control Features via the requested field.

ENDPOINTS

```
POST /v1/treasury/financial_accounts/:id/features
 GET /v1/treasury/financial_accounts/:id/features
```

SHOW

Transactions

Transactions represent changes to a FinancialAccount's balance.

ENDPOINTS

```
 GET /v1/treasury/transactions/:id
 GET /v1/treasury/transactions
```

SHOW

Transaction Entries

TransactionEntries represent individual units of money movements within a single Transaction.

ENDPOINTS

```
GET /v1/treasury/transaction_entries/:id
GET /v1/treasury/transaction_entries
```

SHOW

Outbound Transfers

Use OutboundTransfers to transfer funds from a FinancialAccount to a PaymentMethod belonging to the same entity. To send funds to a different party, use OutboundPayments instead. You can send funds over ACH rails or through a domestic wire transfer to a user's own external bank account.

Simulate OutboundTransfer state changes with the `/v1/test_helpers/treasury/outbound_transfers` endpoints. These methods can only be called on test mode objects.

Related guide: Moving money with Treasury using OutboundTransfer objects

ENDPOINTS

```
POST /v1/treasury/outbound_transfers
 GET /v1/treasury/outbound_transfers/:id
 GET /v1/treasury/outbound_transfers
POST /v1/treasury/outbound_transfers/:id/cancel
POST /v1/test_helpers/treasury/outbound_transfers/:id/fail
POST /v1/test_helpers/treasury/outbound_transfers/:id/post
POST /v1/test_helpers/treasury/outbound_transfers/:id/return
POST /v1/test_helpers/treasury/outbound_transfers/:id
```

SHOW

Outbound Payments

Use OutboundPayments to send funds to another party's external bank account or FinancialAccount. To send money to an account belonging to the same user, use an OutboundTransfer.

Simulate OutboundPayment state changes with the `/v1/test_helpers/treasury/outbound_payments` endpoints. These methods can only be called on test mode objects.

Related guide: Moving money with Treasury using OutboundPayment objects

ENDPOINTS

```
POST /v1/treasury/outbound_payments
 GET /v1/treasury/outbound_payments/:id
 GET /v1/treasury/outbound_payments
POST /v1/treasury/outbound_payments/:id/cancel
POST /v1/test_helpers/treasury/outbound_payments/:id/fail
POST /v1/test_helpers/treasury/outbound_payments/:id/post
POST /v1/test_helpers/treasury/outbound_payments/:id/return
POST /v1/test_helpers/treasury/outbound_payments/:id
```

SHOW

Inbound Transfers

Use InboundTransfers to add funds to your FinancialAccount via a PaymentMethod that is owned by you. The funds will be transferred via an ACH debit.

Related guide: Moving money with Treasury using InboundTransfer objects

ENDPOINTS

```
POST /v1/treasury/inbound_transfers
 GET /v1/treasury/inbound_transfers/:id
 GET /v1/treasury/inbound_transfers
POST /v1/treasury/inbound_transfers/:id/cancel
POST /v1/test_helpers/treasury/inbound_transfers/:id/fail
POST /v1/test_helpers/treasury/inbound_transfers/:id/return
POST /v1/test_helpers/treasury/inbound_transfers/:id/succeed
```

SHOW

Received Credits

ReceivedCredits represent funds sent to a FinancialAccount (for example, via ACH or wire). These money movements are not initiated from the FinancialAccount.

ENDPOINTS

```
 GET /v1/treasury/received_credits/:id
 GET /v1/treasury/received_credits
POST /v1/test_helpers/treasury/received_credits
```

SHOW

Received Debits

ReceivedDebits represent funds pulled from a FinancialAccount. These are not initiated from the FinancialAccount.

ENDPOINTS

```
 GET /v1/treasury/received_debits/:id
 GET /v1/treasury/received_debits
POST /v1/test_helpers/treasury/received_debits
```

SHOW

Credit Reversals

You can reverse some ReceivedCredits depending on their network and source flow. Reversing a ReceivedCredit leads to the creation of a new object known as a CreditReversal.

ENDPOINTS

```
POST /v1/treasury/credit_reversals
 GET /v1/treasury/credit_reversals/:id
 GET /v1/treasury/credit_reversals
```

SHOW

Debit Reversals

You can reverse some ReceivedDebits depending on their network and source flow. Reversing a ReceivedDebit leads to the creation of a new object known as a DebitReversal.

ENDPOINTS

```
POST /v1/treasury/debit_reversals
 GET /v1/treasury/debit_reversals/:id
 GET /v1/treasury/debit_reversals
```

SHOW

Feature

A feature represents a monetizable ability or functionality in your system. Features can be assigned to products, and when those products are purchased, Stripe will create an entitlement to the feature for the purchasing customer.

ENDPOINTS

```
POST /v1/entitlements/features
 GET /v1/entitlements/features
POST /v1/entitlements/features/:id
```

SHOW

Product Feature

A product_feature represents an attachment between a feature and a product. When a product is purchased that has a feature attached, Stripe will create an entitlement to the feature for the purchasing customer.

ENDPOINTS

```
 GET /v1/products/:id/features
POST /v1/products/:id/features
```

```
DELETE /v1/products/:id/features/:id
```

SHOW

Active Entitlement

An active entitlement describes access to a feature for a customer.

ENDPOINTS

```
GET /v1/entitlements/active_entitlements/:id
GET /v1/entitlements/active_entitlements
```

SHOW

Scheduled Queries

If you have scheduled a Sigma query, you'll receive a `sigma.scheduled_query_run.created` webhook each time the query runs. The webhook contains a `ScheduledQueryRun` object, which you can use to retrieve the query results.

ENDPOINTS

```
GET /v1/sigma/scheduled_query_runs/:id
GET /v1/sigma/scheduled_query_runs
```

SHOW

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.

ENDPOINTS

```
POST /v1/reporting/report_runs
 GET /v1/reporting/report_runs/:id
 GET /v1/reporting/report_runs
```

SHOW

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.

ENDPOINTS

```
GET /v1/reporting/report_types/:id
GET /v1/reporting/report_types
```

SHOW

Accounts

A Financial Connections Account represents an account that exists outside of Stripe, to which you have been granted some degree of access.

ENDPOINTS

```
 GET /v1/financial_connections/accounts/:id
 GET /v1/financial_connections/accounts
POST /v1/financial_connections/accounts/:id/disconnect
POST /v1/financial_connections/accounts/:id/refresh
POST /v1/financial_connections/accounts/:id/subscribe
POST /v1/financial_connections/accounts/:id/unsubscribe
```

SHOW

Account Owner

Describes an owner of an account.

ENDPOINTS

```
GET /v1/financial_connections/accounts/:id/owners
```

SHOW

Session

A Financial Connections Session is the secure way to programmatically launch the client-side Stripe.js modal that lets your users link their accounts.

ENDPOINTS

```
POST /v1/financial_connections/sessions
 GET /v1/financial_connections/sessions/:id
```

SHOW

Transactions

A Transaction represents a real transaction that affects a Financial Connections Account balance.

ENDPOINTS

```
 GET /v1/financial_connections/transactions/:id
 GET /v1/financial_connections/transactions
```

SHOW

Tax Calculations

A Tax Calculation allows you to calculate the tax to collect from your customer.

Related guide: Calculate tax in your custom payment flow

ENDPOINTS

```
POST /v1/tax/calculations
 GET /v1/tax/calculations/:id/line_items
```

```
GET /v1/tax/calculations/:id
```

SHOW

Tax Registrations

A Tax `Registration` lets us know that your business is registered to collect tax on payments within a region, enabling you to automatically collect tax.

Stripe doesn't register on your behalf with the relevant authorities when you create a Tax `Registration` object. For more information on how to register to collect tax, see our guide.

Related guide: Using the Registrations API

ENDPOINTS

```
POST /v1/tax/registrations
POST /v1/tax/registrations/:id
 GET /v1/tax/registrations/:id
 GET /v1/tax/registrations
```

SHOW

Tax Transactions

A Tax Transaction records the tax collected from or refunded to your customer.

Related guide: Calculate tax in your custom payment flow

ENDPOINTS

```
POST /v1/tax/transactions/create_reversal
POST /v1/tax/transactions/create_from_calculation
```

```
GET /v1/tax/transactions/:id/line_items
GET /v1/tax/transactions/:id
```

SHOW

Tax Settings

You can use Tax `Settings` to manage configurations used by Stripe Tax calculations.

Related guide: Using the Settings API

ENDPOINTS

```
POST /v1/tax/settings
 GET /v1/tax/settings
```

SHOW

Verification Session

A VerificationSession guides you through the process of collecting and verifying the identities of your users. It contains details about the type of verification, such as what verification check to perform. Only create one VerificationSession for each verification in your system.

A VerificationSession transitions through multiple statuses throughout its lifetime as it progresses through the verification flow. The VerificationSession contains the user's verified data after verification checks are complete.

Related guide: The Verification Sessions API

ENDPOINTS

```
POST /v1/identity/verification_sessions
```

```
POST /v1/identity/verification_sessions/:id
 GET /v1/identity/verification_sessions/:id
 GET /v1/identity/verification_sessions
POST /v1/identity/verification_sessions/:id/cancel
POST /v1/identity/verification_sessions/:id/redact
```

SHOW

Verification Report

A VerificationReport is the result of an attempt to collect and verify data from a user. The collection of verification checks performed is determined from the `type` and `options` parameters used. You can find the result of each verification check performed in the appropriate sub-resource: `document`, `id_number`, `selfie`.

Each VerificationReport contains a copy of any data collected by the user as well as reference IDs which can be used to access collected images through the FileUpload API. To configure and create VerificationReports, use the VerificationSession API.

Related guide: Accessing verification results.

ENDPOINTS

```
GET /v1/identity/verification_reports/:id
GET /v1/identity/verification_reports
```

SHOW

Crypto Onramp Session

A Crypto Onramp Session represents your customer's session as they purchase cryptocurrency through Stripe. Once payment is successful, Stripe will fulfill the delivery of cryptocurrency to your user's wallet

and contain a reference to the crypto transaction ID.

You can create an onramp session on your server and embed the widget on your frontend. Alternatively, you can redirect your users to the standalone hosted onramp.

Related guide: Integrate the onramp

ENDPOINTS

```
POST /v1/crypto/onramp_sessions
 GET /v1/crypto/onramp_sessions/:id
 GET /v1/crypto/onramp_sessions
```