stripe-django Documentation

Release 0.1

Tony Narlock

October 13, 2016

Contents

Explore:

CHAPTER 1

Models Reference

1.1 Account

class stripe_django.models.account.**Account**(**args*, ***kwargs*)

Stripe Account object.

This is an object representing your Stripe account. You can retrieve it to see properties on the account like its current e-mail address or if the account is enabled yet to make live charges.

Some properties, marked as "managed accounts only", are only available to platforms who want to create and manage Stripe accounts.

Parameters

- **id** (*AutoField*) – Id
- **charges_enabled** (*BooleanField*) – Whether or not the account can create live charges
- **country** (*CharField*) – The country of the account
- **currencies_supports** (*JSONField*) – The currencies this account can submit when creating charges
- **default_currency** (*CharField*) – The currency this account has chosen to use as the default
- **details_submitted** (*BooleanField*) – Whether or not account details have been submitted yet. Standalone accounts cannot receive transfers before this is true.
- **transfers_enabled** (*BooleanField*) – Whether or not Stripe will send automatic transfers for this account. This is only false when Stripe is waiting for additional information from the account holder.
- **display_name** (*CharField*) – The display name for this account. This is used on the Stripe dashboard to help you differentiate between accounts.
- **email** (*EmailField*) – The primary user's email address
- **statement_descriptor** (*TextField*) – The text that will appear on credit card statements
- **timezone** (*CharField*) – The timezone used in the Stripe dashboard for this account. A list of possible timezone values is maintained at the IANA Timezone Database.
- **business_name** (*CharField*) – The publicly visible name of the business

- **business_url** (*URLField*) – The publicly visible website of the business
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the charge in a structured format.
- **support_phone** (*CharField*) – The publicly visible support phone number for the business
- **managed** (*BooleanField*) – Whether or not the account is managed by your platform. Returns null if the account was not created by a platform.
- **bank_accounts** (*JSONField*) – (Managed Accounts Only) Bank accounts currently attached to this account.
- **debit_negative_balances** (*BooleanField*) – (Managed Accounts Only) Whether or not Stripe will attempt to reclaim negative account balances from this account's bank account.
- **decline_charge_on** (*JSONField*) – (Managed Accounts Only) Account-level settings to automatically decline certain types of charges regardless of the bank's decision.
- **legal_entity** (*JSONField*) – (Managed Accounts Only) Information regarding the owner of this account, including verification status.
- **product_description** (*TextField*) – (Managed Accounts Only) An internal-only description of the product or service provided. This is used by Stripe in the event the account gets flagged for potential fraud.
- **tos_acceptance** (*JSONField*) – (Managed Accounts Only) Who accepted the Stripe terms of service, and when they accepted it.
- **transfer_schedule** (*JSONField*) – (Managed Accounts Only) When payments collected will be automatically paid out to the account holder's bank account
- **verfication** (*JSONField*) – (Managed Accounts Only) That state of the account's information requests, including what information is needed and by when it must be provided.

1.2 Application Fee

class stripe_django.models.application_fee.**ApplicationFee**(**args*, ***kwargs*)

Stripe Application Fee object.

When you collect a transaction fee on top of a charge made for your user (using Stripe Connect), an application fee object is created in your account. You can list, retrieve, and refund application fees.

For more information on collecting transaction fees, see our documentation.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **account_id** (ForeignKey to *Account*) – ID of the Stripe account this fee was taken from.
- **amount** (*IntegerField*) – Amount earned, in cents.
- **application** (*CharField*) – ID of the Connect Application that earned the fee.

- **balance_transaction_id** (ForeignKey to *BalanceTransaction*) – Balance transaction that describes the impact of this collected application fee on your account balance (not including refunds).
- **charge_id** (ForeignKey to *Charge*) – ID of the charge that the application fee was taken from.
- **created** (*DateTimeField*) – Created
- **currency** (*CharField*) – Three-letter ISO currency code representing the currency of thecharge.
- **refunded** (*BooleanField*) – Whether or not the fee has been fully refunded. If the fee is only partially refunded, this attribute will still be false.
- **amount_refunded** (*PositiveIntegerField*) – Amount refunded

class stripe_django.models.application_fee.**ApplicationFeeRefund**(**args*, ***kwargs*)

Stripe Application Fee Refund object.

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 that the fee was originally collected from.

Parameters

- **id** (*AutoField*) – Id
- **amount** (*IntegerField*) – Amount reversed, in cents.
- **created** (*DateTimeField*) – Created
- **currency** (*CharField*) – Three-letter ISO currency code representing the currency of the reverse.
- **balance_transaction_id** (ForeignKey to *BalanceTransaction*) – Balance transaction that describes the impact of this reversal on your account balance.
- **fee_id** (ForeignKey to *ApplicationFee*) – ID of the application fee that was refunded.
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the charge in a structured format.

1.3 Balance

class stripe_django.models.balance.**Balance**(**args*, ***kwargs*)

Stripe Balance object.

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 a list of the balance history, which contains a full list of transactions that have ever contributed to the balance (charges, refunds, transfers, and so on).

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **available** (*JSONField*) – Funds that are available to be paid out automatically by Stripe or explicitly via the transfers API.

- **pending** (*JSONField*) – Funds that are not available in the balance yet, due to the 7-day rolling pay cycle.

1.4 Balance Transaction

class stripe_django.models.balance_transaction.**BalanceTransaction**(*id*, *amount*, *available_on*, *created*, *currency*, *fee*, *fee_details*, *net*, *status*, *type*, *description*, *source*, *sourced_transfers*)

Parameters

- **id** (*AutoField*) – Id
- **amount** (*IntegerField*) – Gross amount of the transaction, in cents
- **available_on** (*DateTimeField*) – The date the transaction's net funds will become available in the Stripe balance.
- **created** (*DateTimeField*) – Created
- **currency** (*CharField*) – Currency
- **fee** (*IntegerField*) – Fees (in cents) paid for this transaction
- **fee_details** (*JSONField*) – Detailed breakdown of fees (in cents) paid for this transaction
- **net** (*IntegerField*) – Net amount of the transaction, in cents.
- **status** (*CharField*) – If the transaction's net funds are available in the Stripe balance yet. Either `available` or `pending`.
- **type** (*CharField*) – Type of the transaction, one of: `charge`, `refund`, `adjustment`, `application_fee`, `application_fee_refund`, `transfer`, `transfer_cancel` or `transfer_failure`.
- **description** (*CharField*) – Description
- **source** (*JSONField*) – The Stripe object this transaction is related to.
- **sourced_transfers** (*JSONField*) – The transfers (if any) for which source is a source_transaction.

1.5 Bitcon Receiver

class stripe_django.models.bitcoin_receiver.**BitCoinReceiver**(**args*, ***kwargs*)

Stripe Bitcoin Receiver object.

A Bitcoin receiver wraps a Bitcoin address so that a customer can push a payment to you. This guide describes how to use receivers to create Bitcoin payments.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **active** (*BooleanField*) – True when this bitcoin receiver has received a non-zero amount of bitcoin.
- **amount** (*PositiveIntegerField*) – The amount of currency that you are collecting as payment.
- **amount_received** (*PositiveIntegerField*) – The amount of currency to which bitcoin_amount_received has been converted.
- **bitcoin_amount** (*PositiveIntegerField*) – The amount of bitcoin that the customer should send to fill the receiver. The bitcoin_amount is denominated in Satoshi: there are 10^8 Satoshi in one bitcoin.
- **bitcoin_amount_received** (*PositiveIntegerField*) – The amount of bitcoin that has been sent by the customer to this receiver.
- **bitcoin_uri** (*URLField*) – This URI can be displayed to the customer as a clickable link (to activate their bitcoin client) or as a QR code (for mobile wallets).
- **created** (*DateTimeField*) – Created
- **currency** (*CharField*) – Three-letter ISO currency code representing the currency to which the bitcoin will be converted.
- **filled** (*BooleanField*) – This flag is initially false and updates to true when the customer sends the bitcoin_amount to this receiver.
- **inbound_address** (*CharField*) – A bitcoin address that is specific to this receiver. The customer can send bitcoin to this address to fill the receiver.
- **transactions** (*JSONField*) – A list with one entry for each time that the customer sent bitcoin to the receiver. Hidden when viewing the receiver with a publishable key.
- **uncaptured_funds** (*BooleanField*) – This receiver contains uncaptured funds that can be used for a payment or refunded.
- **description** (*CharField*) – Description
- **email** (*EmailField*) – The customer's email address, set by the API call that creates the receiver.
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the charge in a structured format.
- **payment** (*CharField*) – The ID of the payment created from the receiver, if any. Hidden when viewing the receiver with a publishable key.
- **refund_address** (*CharField*) – The refund address for these bitcoin, if communicated by the customer.
- **customer_id** (ForeignKey to *Customer*) – Customer

1.6 Card

class stripe_django.models.card.**Card**(**args*, ***kwargs*)
Stripe Card object.

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.

Parameters

- **id** (*AutoField*) – ID of card (used in conjunction with a customer or recipient ID)
- **brand** (*CharField*) – Card brand. Can be `Visa`, `American Express`, `MasterCard`, `Discover`, `JCB`, `Diners Club`, or `Unknown`.
- **exp_month** (*IntegerField*) – Exp month
- **exp_year** (*IntegerField*) – Exp year
- **funding** (*CharField*) – Funding
- **last4** (*PositiveIntegerField*) – Last4
- **address_city** (*CharField*) – Address city
- **address_country** (*CharField*) – Billing address country, if provided when creating card
- **address_line1** (*CharField*) – Address line1
- **address_line1_check** (*CharField*) – If `address_line1` was provided, results of the check: `pass`, `fail`, `unavailable`, or `unchecked`.
- **address_line2** (*CharField*) – Address line2
- **address_state** (*CharField*) – Address state
- **address_zip** (*CharField*) – Address zip
- **address_zip_check** (*CharField*) – If `address_zip` was provided, results of the check: `pass`, `fail`, `unavailable`, or `unchecked`.
- **country** (*CharField*) – Two-letter ISO code representing the country of the card. You could use this attribute to get a sense of the international breakdown of cards you've collected.
- **customer_id** (ForeignKey to *Customer*) – The customer that this card belongs to. This attribute will not be in the card object if the card belongs to a recipient instead.
- **cvc_check** (*CharField*) – Cvc check
- **dynamic_last4** (*CharField*) – (For Apple Pay integrations only.) The last four digits of the device account number.
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a card object. it can be useful for storing additional information about the card in a structured format.
- **name** (*CharField*) – Cardholder name
- **fingerprint** (*CharField*) – Uniquely identifies this particular card number. You can use this attribute to check whether two customers who've signed up with you are using the same card number, for example.

1.7 Charge

class `stripe_django.models.charge.`**Charge**(**args*, ***kwargs*)
Stripe Charge Object.

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

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **amount** (*IntegerField*) – Amount charged in cents
- **captured** (*BooleanField*) – If the charge was created without capturing, this boolean represents whether or not it is still uncaptured or has since been captured.
- **created** (*DateTimeField*) – Created
- **currency** (*CharField*) – Three-letter ISO currency code representing the currency in which the charge was made.
- **paid** (*BooleanField*) – true if the charge succeeded, or was successfully authorized for later capture.
- **refunded** (*BooleanField*) – Whether or not the charge has been fully refunded. If the charge is only partially refunded, this attribute will still be false.
- **source** (*JSONField*) – For most Stripe users, the source of every charge is a credit or debit card. This hash is then the card object describing that card.
- **status** (*CharField*) – The status of the payment is either `succeeded` or `failed`.
- **amount_refunded** (*PositiveIntegerField*) – Amount in cents refunded (can be less than the amount attribute on the charge if a partial refund was issued).
- **balance_transaction** (*CharField*) – ID of the balance transaction that describes the impact of this charge on your account balance (not including refunds or disputes).
- **customer_id** (ForeignKey to *Customer*) – ID of the customer this charge is for if one exists.
- **description** (*CharField*) – Description
- **dispute_id** (ForeignKey to *Dispute*) – Details about the dispute if the charge has been disputed.
- **failure_code** (*CharField*) – Error code explaining reason for charge failure if available (see the errors section for a list of codes).
- **failure_message** (*CharField*) – Message to user further explaining reason for charge failure if available.
- **invoice_id** (ForeignKey to *Invoice*) – ID of the invoice this charge is for if one exists.
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the charge in a structured format.
- **receipt_email** (*EmailField*) – This is the email address that the receipt for this charge was sent to.
- **receipt_number** (*CharField*) – This is the transaction number that appears on email receipts sent for this charge.
- **application_fee** (*CharField*) – The application fee (if any) for the charge. See the Connect documentation for details.
- **destination** (*CharField*) – The account (if any) the charge was made on behalf of. See the Connect documentation for details.

- **fraud_details** (*JSONField*) – Hash with information on fraud assessments for the charge. Assessments reported by you have the key `user_report` and, if set, possible values of `safe` and `fraudulent`. Assessments from Stripe have the key `stripe_report` and, if set, the value fraudulent.
- **shipping** (*JSONField*) – Shipping information for the charge.
- **transfer** (*CharField*) – ID of the transfer to the `destination` account (only applicable if the charge was created using the `destination` parameter).

1.8 Coupon

class `stripe_django.models.coupon.`**Coupon**(**args*, ***kwargs*)

Stipe Coupon object.

A coupon contains information about a percent-off or amount-off discount you might want to apply to a customer. Coupons only apply to invoices; they do not apply to one-off charges.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **created** (*DateTimeField*) – Created
- **duration** (*CharField*) – One of `forever`, `once`, and `repeating`. Describes how long a customer who applies this coupon will get the discount.
- **amount_off** (*PositiveIntegerField*) – Amount (in the `currency` specified) that will be taken off the subtotal of any invoices for this customer.
- **currency** (*CharField*) – If `amount_off` has been set, the currency of the amount to take off.
- **duration_in_months** (*PositiveIntegerField*) – If `duration` is `repeating`, the number of months the coupon applies. Null if coupon `duration` is `forever``or ``once`.
- **max_redemptions** (*PositiveIntegerField*) – Maximum number of times this coupon can be redeemed, in total, before it is no longer valid.
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the coupon in a structured format.
- **percent_off** (*PositiveIntegerField*) – Percent that will be taken off the subtotal of any invoices for this customer for the duration of the coupon. For example, a coupon with percent_off of 50 will make a $100 invoice $50 instead.
- **redeem_by** (*DateTimeField*) – Date after which the coupon can no longer be redeemed
- **times_redeemed** (*PositiveIntegerField*) – Number of times this coupon has been applied to a customer.
- **valid** (*BooleanField*) – Taking account of the above properties, whether this coupon can still be applied to a customer

1.9 Customer

class stripe_django.models.customer.**Customer**(**args*, ***kwargs*)
Stripe Customer object.

Customer objects allow you to perform recurring charges and track multiple charges that are associated with the same customer. The API allows you to create, delete, and update your customers. You can retrieve individual customers as well as a list of all your customers.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **created** (*DateTimeField*) – Created
- **account_balance** (*DateTimeField*) – Current balance, if any, being stored on the customer's account. If negative, the customer has credit to apply to the next invoice. If positive, the customer has an amount owed that will be added to the next invoice. The balance does not refer to any unpaid invoices; it solely takes into account amounts that have yet to be successfully applied to any invoice. This balance is only taken into account for recurring charges.
- **currency** (*CharField*) – The currency the customer can be charged in for recurring billing purposes (subscriptions, invoices, invoice items).
- **default_source** (*CharField*) – ID of the default source attached to this customer.
- **delinquent** (*CharField*) – Whether or not the latest charge for the customer's latest invoice has failed
- **description** (*CharField*) – Description
- **email** (*EmailField*) – Email
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. It can be useful for storing additional information about the customer in a structured format.
- **sources** (*JSONField*) – The customer's payment sources, if any

1.10 Discount

class stripe_django.models.discount.**Discount**(**args*, ***kwargs*)
Stripe Discount object.

A discount represents the actual application of a coupon to a particular customer. It contains information about when the discount began and when it will end.

Parameters

- **id** (*AutoField*) – Id
- **coupon_id** (ForeignKey to *Coupon*) – Hash describing the coupon applied to create this discount
- **customer_id** (ForeignKey to *Customer*) – Customer
- **start** (*DateTimeField*) – Date that the coupon was applied

- **end** (*DateTimeField*) – If the coupon has a duration of `once` or `repeating`, the date that this discount will end. If the coupon used has a forever duration, this attribute will be null.

1.11 Dispute

class `stripe_django.models.dispute.`**Dispute**(**args*, ***kwargs*)

Stripe Dispute object.

A dispute occurs when a customer questions your charge with their bank or credit card company. When a customer disputes your charge, you're given the opportunity to respond to the dispute with evidence that shows the charge is legitimate. You can find more information about the dispute process in our disputes FAQ.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **amount** (*IntegerField*) – Disputed amount. Usually the amount of the charge, but can differ (usually because of currency fluctuation or because only part of the order is disputed).
- **created** (*DateTimeField*) – Date dispute was opened
- **currency** (*CharField*) – Three-letter ISO currency code representing the currency of the amount that was disputed.
- **reason** (*CharField*) – Reason given by cardholder for dispute. Possible values are `duplicate`, `fraudulent`, `subscription_canceled`, `product_unacceptable`, `product_not_received`, `unrecognized`, `credit_not_processed`, `general`. Read more about dispute reasons.
- **status** (*CharField*) – Current status of dispute. Possible values are `warning_needs_response`, `warning_under_review`, `warning_closed`, `needs_response`, `response_disabled`, `under_review`, `charge_refunded`, `won`, `lost`.
- **evidence_id** (ForeignKey to *DisputeEvidence*) – Evidence provided to respond to a dispute. Updating any field in the hash will submit all fields in the hash for review.
- **evidence_details** (*JSONField*) – Information about the evidence submission.
- **is_charge_refundable** (*BooleanField*) – If true, it is still possible to refund the disputed payment. once the payment has been fully refunded, no further funds will be withdrawn from your stripe account as a result of this dispute.
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the dispute in a structured format.
- **balance_transaction** (*ManyToManyField*) – List of zero, one, or two balance transactions that show funds withdrawn and reinstated to your Stripe account as a result of this dispute.

1.12 Dispute Evidence

class `stripe_django.models.dispute_evidence.`**DisputeEvidence**(**args*, ***kwargs*)

Stripe Dispute Evidence object.

DisputeEvidence revserse relations will be prefixed with `dispute_`.

Parameters

- **id** (*AutoField*) – Id
- **access_activity_log** (*TextField*) – Any server or activity logs showing proof that the customer accessed or downloaded the purchased digital product. This information should include IP addresses, corresponding timestamps, and any detailed recorded activity.
- **billing_address** (*TextField*) – The billing addess provided by the customer.
- **cancelling_policy_id** (ForeignKey to *FileUpload*) – (ID of a file upload) Your subscription cancellation policy, as shown to the customer.
- **cancellation_policy_disclosure** (*TextField*) – An explanation of how and when the customer was shown your refund policy prior to purchase.
- **cancellation_rebuttal** (*TextField*) – A justification for why the customer's subscription was not canceled.
- **customer_communication_id** (ForeignKey to *FileUpload*) – (ID of a file upload) Any communication with the customer that you feel is relevant to your case (for example emails proving that they received the product or service, or demonstrating their use of or satisfaction with the product or service)
- **customer_email_address** (*EmailField*) – Customer email address
- **customer_name** (*CharField*) – Customer name
- **customer_purchase_ip** (*CharField*) – Customer purchase ip
- **customer_signature_id** (ForeignKey to *FileUpload*) – (ID of a file upload) A relevant document or contract showing the customer's signature.
- **duplicate_charge_documentation_id** (ForeignKey to *FileUpload*) – ID of a file upload) Documentation for the prior charge that can uniquely identify the charge, such as a receipt, shipping label, work order, etc. This document should be paired with a similar document from the disputed payment that proves the two payments are separate.
- **duplicate_charge_explanation** (*TextField*) – An explanation of the difference between the disputed charge and the prior charge that appears to be a duplicate.
- **duplicate_charge_id_id** (ForeignKey to *Charge*) – The Stripe ID for the prior charge which appears to be a duplicate of the disputed charge.
- **product_description** (*TextField*) – A description of the product or service which was sold.
- **receipt_id** (ForeignKey to *FileUpload*) – (ID of a file upload) Any receipt or message sent to the customer notifying them of the charge.
- **refund_policy_id** (ForeignKey to *FileUpload*) – (ID of a file upload) Your refund policy, as shown to the customer.
- **refund_policy_disclosure** (*TextField*) – Documentation demonstrating that the customer was shown your refund policy prior to purchase.
- **refund_refusal_explanation** (*TextField*) – A justification for why the customer is not entitled to a refund.
- **service_date** (*DateTimeField*) – The date on which the customer received or began receiving the purchased service, in a clear human-readable format.

- **service_documentation_id** (ForeignKey to *FileUpload*) – (ID of a file upload) Documentation showing proof that a service was provided to the customer. This could include a copy of a signed contract, work order, or other form of written agreement.
- **shipping_address** (*TextField*) – The address to which a physical product was shipped. You should try to include as much complete address information as possible.
- **shipping_carrier** (*TextField*) – The delivery service that shipped a physical product, such as Fedex, UPS, USPS, etc. If multiple carriers were used for this purchase, please separate them with commas.
- **shipping_date** (*DateTimeField*) – The date on which a physical product began its route to the shipping address, in a clear human-readable format.
- **shipping_documentation_id** (ForeignKey to *FileUpload*) – (ID of a file upload) Documentation showing proof that a product was shipped to the customer at the same address the customer provided to you. This could include a copy of the shipment receipt, shipping label, etc, and should show the full shipping address of the customer, if possible.
- **shipping_tracking_number** (*TextField*) – The tracking number for a physical product, obtained from the delivery service. If multiple tracking numbers were generated for this purchase, please separate them with commas.
- **uncategorized_file_id** (ForeignKey to *FileUpload*) – (ID of a file upload) Any additional evidence or statements.
- **uncategorized_text** (*TextField*) – Any additional evidence or statements.

1.13 Event

class stripe_django.models.event.**Event**(**args*, ***kwargs*)

Stripe Event object.

Events are our way of letting you know about something interesting that has just happened 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; or, when an invoice can't be paid we create an `invoice.payment_failed` event. Note that many API requests may cause multiple events to be created. For example, if you create a new subscription for a customer, you will receive both a `customer.subscription.created` event and a `charge.succeeded` event.

Like our other API resources, you can retrieve an individual event or a list of events from the API. We also have a system for sending the events directly to your server, called webhooks. Webhooks are managed in your account settings, and our webhook guide will help you get them set up.

NOTE: Right now, we only guarantee access to events through the Retrieve Event API for 30 days.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **created** (*DateTimeField*) – Created
- **data** (*JSONField*) – Hash containing data associated with the event.
- **pending_webhooks** (*PositiveIntegerField*) – Number of webhooks yet to be delivered successfully (return a 20x response) to the URLs you've specified.
- **type** (*CharField*) – Description of the event: e.g. invoice.created, charge.refunded, etc.

- **api_version** (*CharField*) – The Stripe API version used to render data. Note: this property is populated for events on or after October 31, 2014.
- **request** (*CharField*) – ID of the API request that caused the event. If null, the event was automatic (e.g. Stripe's automatic subscription handling). Request logs are available in the dashboard but currently not in the API. Note: this property is populated for events on or after April 23, 2013.

1.14 File Upload

class stripe_django.models.file_upload.**FileUpload**(*id*, *created*, *purpose*, *size*, *type*, *url*)

Parameters

- **id** (*AutoField*) – Id
- **created** (*DateTimeField*) – Created
- **purpose** (*CharField*) – The purpose of the uploaded file. Possible values are identity_document, dispute_evidence.
- **size** (*IntegerField*) – The size in bytes of the file upload object.
- **type** (*CharField*) – The type of the file returned. Returns one of the following: pdf, jpg, png.
- **url** (*URLField*) – A read-only URL where the uploaded file can be accessed. Will be nil unless the uploaded file has one of the following purposes: dispute_evidence. Also nil if retrieved with the publishable API key.

1.15 Invoice

class stripe_django.models.invoice.**Invoice**(**args*, ***kwargs*)

Stripe Invoice object.

Invoices are statements of what a customer owes for a particular billing period, including subscriptions, invoice items, and any automatic proration adjustments if necessary.

Once an invoice is created, payment is automatically attempted. Note that the payment, while automatic, does not happen exactly at the time of invoice creation. If you have configured webhooks, the invoice will wait until one hour after the last webhook is successfully sent (or the last webhook times out after failing).

Any customer credit on the account is applied before determining how much is due for that invoice (the amount that will be actually charged). If the amount due for the invoice is less than 50 cents (the minimum for a charge), We add the amount to the customer's running account balance to be added to the next invoice. If this amount is negative, it will act as a credit to offset the next invoice. Note that the customer account balance does not include unpaid invoices; it only includes balances that need to be taken into account when calculating the amount due for the next invoice.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **amount_due** (*IntegerField*) – Final amount due at this time for this invoice. If the invoice's total is smaller than the minimum charge amount, for example, or if there is account credit that can be applied to the invoice, the amount_due may be 0. If there is a positive

starting_balance for the invoice (the customer owes money), the amount_due will also take that into account. The charge that gets generated for the invoice will be for the amount specified in amount_due.

- **attempt_count** (*PositiveIntegerField*) – Number of payment attempts made for this invoice, from the perspective of the payment retry schedule. Any payment attempt counts as the first attempt, and subsequently only automatic retries increment the attempt count. In other words, manual payment attempts after the first attempt do not affect the retry schedule.
- **attempted** (*BooleanField*) – Whether or not an attempt has been made to pay the invoice. An invoice is not attempted until 1 hour after the invoice.created webhook, for example, so you might not want to display that invoice as unpaid to your users.
- **closed** (*BooleanField*) – Whether or not the invoice is still trying to collect payment. An invoice is closed if it's either paid or it has been marked closed. A closed invoice will no longer attempt to collect payment.
- **currency** (*CharField*) – Currency
- **customer_id** (ForeignKey to *Customer*) – Customer
- **date** (*DateTimeField*) – Date
- **forgiven** (*BooleanField*) – Whether or not the invoice has been forgiven. Forgiving an invoice instructs us to update the subscription status as if the invoice were succcessfully paid. Once an invoice has been forgiven, it cannot be unforgiven or reopened
- **lines** (*JSONField*) – The individual line items that make up the invoice. lines is sorted as follows: invoice items in reverse chronological order, followed by the subscription, if any.
- **paid** (*BooleanField*) – Whether or not payment was successfully collected for this invoice. An invoice can be paid (most commonly) with a charge or with credit from the customer's account balance.
- **period_end** (*DateTimeField*) – End of the usage period during which invoice items were added to this invoice
- **period_start** (*DateTimeField*) – Start of the usage period during which invoice items were added to this invoice
- **starting_balance** (*IntegerField*) – Starting customer balance before attempting to pay invoice. If the invoice has not been attempted yet, this will be the current customer balance.
- **subtotal** (*IntegerField*) – Total of all subscriptions, invoice items, and prorations on the invoice before any discount is applied
- **total** (*IntegerField*) – Total after discount
- **application_fee** (*IntegerField*) – The fee in cents that will be applied to the invoice and transferred to the application owner's Stripe account when the invoice is paid.
- **description** (*CharField*) – Description
- **discount_id** (ForeignKey to *Discount*) – Discount
- **ending_balance** (*IntegerField*) – Ending customer balance after attempting to pay invoice. If the invoice has not been attempted yet, this will be null.
- **next_payment_attempt** (*DateTimeField*) – The time at which payment will next be attempted.

- **receipt_number** (*CharField*) – This is the transaction number that appears on email receipts sent for this invoice.
- **statement_descriptor** (*CharField*) – Extra information about an invoice for the customer's credit card statement.
- **subscription_id** (ForeignKey to *Subscription*) – The subscription that this invoice was prepared for, if any.
- **webhooks_delivered_at** (*DateTimeField*) – The time at which webhooks for this invoice were successfully delivered (if the invoice had no webhooks to deliver, this will match `date`). Invoice payment is delayed until webhooks are delivered, or until all webhook delivery attempts have been exhausted.
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the invoice in a structured format.
- **subscription_proration_date** (*IntegerField*) – Only set for upcoming invoices that preview prorations. The time used to calculate prorations.
- **tax** (*IntegerField*) – The amount of tax included in the total, calculated from `tax_percent` and the subtotal. If no `tax_percent` is defined, this value will be null.
- **tax_percent** (*DecimalField*) – This percentage of the subtotal has been added to the total amount of the invoice, including invoice line items and discounts. This field is inherited from the subscription's `tax_percent` field, but can be changed before the invoice is paid. This field defaults to null.

1.16 Invoice Item

class `stripe_django.models.invoice_item.`**InvoiceItem**(**args*, ***kwargs*)

Stripe Invoice Item object.

Sometimes you want to add a charge or credit to a customer but only actually charge the customer's card at the end of a regular billing cycle. This is useful for combining several charges to minimize per-transaction fees or having Stripe tabulate your usage-based billing totals.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **amount** (*IntegerField*) – Amount
- **currency** (*CharField*) – Currency
- **customer_id** (ForeignKey to *Customer*) – Customer
- **date** (*DateTimeField*) – Date
- **discountable** (*BooleanField*) – If true, discounts will apply to this invoice item. Always false for prorations.
- **proration** (*BooleanField*) – Whether or not the invoice item was created automatically as a proration adjustment when the customer switched plans
- **description** (*CharField*) – Description
- **invoice** (*CharField*) – Invoice

- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the invoice item in a structured format.
- **period** (*JSONField*) – Period
- **plan_id** (ForeignKey to *Plan*) – If the invoice item is a proration, the plan of the subscription that the proration was computed for.
- **quantity** (*IntegerField*) – If the invoice item is a proration, the quantity of the subscription that the proration was computed for.
- **subscription_id** (ForeignKey to *Subscription*) – The subscription that this invoice item has been created for, if any.

1.17 Plan

class stripe_django.models.plan.**Plan**(**args*, ***kwargs*)

Stripe Plan object.

A subscription plan contains the pricing information for different products and feature levels on your site. For example, you might have a $10/month plan for basic features and a different $20/month plan for premium features.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **amount** (*PositiveIntegerField*) – The amount in cents to be charged on the interval specified
- **created** (*DateTimeField*) – Created
- **currency** (*CharField*) – Currency in which the subscription will be charged
- **interval** (*CharField*) – One of `day`, `week`, `month` or `year`. The frequency with which a subscription should be billed.
- **interval_count** (*PositiveIntegerField*) – The number of intervals (specified in the `interval` property) between each subscription billing. For example, `interval=month` and interval_count=3 bills every 3 months.
- **name** (*CharField*) – Display name of the plan
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the plan in a structured format.
- **trial_period_days** (*PositiveIntegerField*) – Number of trial period days granted when subscribing a customer to this plan. Null if the plan has no trial period.
- **statement_descriptor** (*CharField*) – Extra information about a charge for the customer's credit card statement.

1.18 Recipient

class stripe_django.models.recipient.**Recipient**(**args*, ***kwargs*)

Stripe Recipient object.

With recipient objects, you can transfer money from your Stripe account to a third party bank account or debit card. The API allows you to create, delete, and update your recipients. You can retrieve individual recipients as well as a list of all your recipients.

Recipient objects have been deprecated in favor of Connect, specifically the much more powerful account objects. Please use them instead. If you are already using recipients, please see our migration guide for more information.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **created** (*DateTimeField*) – Created
- **type** (*CharField*) – Type of the recipient, one of `individual` or `corporation`.
- **active_account** (*JSONField*) – Hash describing the current account on the recipient, if there is one.
- **description** (*TextField*) – Description
- **email** (*EmailField*) – Email
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the recipient in a structured format.
- **name** (*CharField*) – Full, legal name of the recipient.
- **default_card_id** (ForeignKey to *Card*) – The default card to use for creating transfers to this recipient.
- **migrated_to** (*CharField*) – Migrated to
- **cards** (*ManyToManyField*) – Cards

1.19 Refund

class `stripe_django.models.refund.`**Refund**(**args*, ***kwargs*)

Stripe Refund objects.

Refund objects allow you to refund a charge that has previously been created but not yet refunded. Funds will be refunded to the credit or debit card that was originally charged. The fees you were originally charged are also refunded.

Parameters

- **id** (*AutoField*) – Id
- **amount** (*IntegerField*) – Amount reversed, in cents.
- **created** (*DateTimeField*) – Created
- **currency** (*IntegerField*) – Three-letter ISO code representing the currency of the reversal.
- **balance_transaction** (*CharField*) – Balance transaction that describes the impact of this reversal on your account balance.
- **charge** (*CharField*) – ID of the charge that was
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. It can be useful for storing additional information about the refund in a structured format.

- **reason** (*CharField*) – Reason for the refund. If set, possible values are duplicate, fraudulent, and requested_by_customer.
- **receipt_number** (*CharField*) – This is the transaction number that appears on email receipts sent for this refund.
- **description** (*CharField*) – Description

1.20 Subscription

class stripe_django.models.subscription.**Subscription**(**args*, ***kwargs*)

Stripe subscription object.

Subscriptions allow you to charge a customer's card on a recurring basis. A subscription ties a customer to a particular plan you've created.

Parameters

- **id** (*AutoField*) – Id
- **cancel_at_period_end** (*BooleanField*) – If the subscription has been canceled with the at_period_end''flag set to ''true, cancel_at_period_end on the subscription will be true. You can use this attribute to determine whether a subscription that has a status of active is scheduled to be canceled at the end of the current period.
- **customer_id** (ForeignKey to *Customer*) – Customer
- **plan_id** (ForeignKey to *Plan*) – Hash describing the plan the customer is subscribed to
- **quantity** (*PositiveIntegerField*) – Quantity
- **start** (*DateTimeField*) – Date the subscription started
- **status** (*CharField*) – Possible values are trialing, active, past_due, canceled, or unpaid. A subscription still in its trial period is trialing and moves to active when the trial period is over. When payment to renew the subscription fails, the subscription becomes past_due. After Stripe has exhausted all payment retry attempts, the subscription ends up with a status of either canceled or unpaid depending on your retry settings. Note that when a subscription has a status of unpaid, no subsequent invoices will be attempted (invoices will be created, but then immediately automatically closed. Additionally, updating customer card details will not lead to Stripe retrying the latest invoice.). After receiving updated card details from a customer, you may choose to reopen and pay their closed invoices.
- **application_fee_percent** (*CharField*) – A positive decimal that represents the fee percentage of the subscription invoice amount that will be transferred to the application owner's Stripe account each billing period.
- **canceled_at** (*DateTimeField*) – If the subscription has been canceled, the date of that cancellation. If the subscription was canceled with cancel_at_period_end, canceled_at will still reflect the date of the initial cancellation request, not the end of the subscription period when the subscription is automatically moved to a canceled state.
- **current_period_start** (*DateTimeField*) – End of the current period that the subscription has been invoiced for. At the end of this period, a new invoice will be created.
- **discount_id** (ForeignKey to *Discount*) – Describes the current discount applied to this subscription, if there is one. When billing, a discount applied to a subscription overrides a discount applied on a customer-wide basis.

- **ended_at** (*DateTimeField*) – If the subscription has ended (either because it was canceled or because the customer was switched to a subscription to a new plan), the date the subscription ended
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the subscription in a structured format.
- **trial_end** (*DateTimeField*) – If the subscription has a trial, the end of that trial.
- **trial_start** (*DateTimeField*) – If the subscription has a trial, the beginning of that trial.
- **tax_percent** (*DecimalField*) – If provided, each invoice created by this subscription will apply the tax rate, increasing the amount billed to the customer.

1.21 Token

class stripe_django.models.token.**Token**(**args*, ***kwargs*)

Stripe Token object.

Often you want to be able to charge credit cards or send payments to bank accounts without having to hold sensitive card information on your own servers. Stripe.js makes this easy in the browser, but you can use the same technique in other environments with our token API.

Tokens can be created with your publishable API key, which can safely be embedded in downloadable applications like iPhone and Android apps. You can then use a token anywhere in our API that a card or bank account is accepted. Note that tokens are not meant to be stored or used more than once—to store these details for use later, you should create Customer or Recipient objects.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **created** (*DateTimeField*) – Created
- **type** (*CharField*) – Type of the token: `card` or `bank_account`
- **used** (*BooleanField*) – Whether or not this token has already been used (tokens can be used only once)
- **bank_account** (*JSONField*) – Hash describing the bank account
- **card** (*JSONField*) – Hash describing the bank account
- **client_ip** (*CharField*) – IP address of the client that generated the token

1.22 Transfer

class stripe_django.models.transfer.**Transfer**(**args*, ***kwargs*)

Stripe Transfer object.

When Stripe sends you money or you initiate a transfer to a bank account, debit card, or connected Stripe account, a transfer object will be created. You can retrieve individual transfers as well as list all transfers.

View the *documentation* on creating transfers via the API.

Parameters

- **id** (*AutoField*) – Id
- **livemode** (*BooleanField*) – Livemode
- **amount** (*IntegerField*) – Amount (in cents) to be transferred to your bank account
- **created** (*DateTimeField*) – Time that this record of the transfer was first created.
- **currency** (*CharField*) – Three-letter ISO code representing the currency of the transfer.
- **date** (*DateTimeField*) – Date the transfer is scheduled to arrive in the bank. This doesn't factor in delays like weekends or bank holidays.
- **reversals** (*JSONField*) – A list of reversals that have been applied to the transfer.
- **reversed** (*BooleanField*) – Whether or not the transfer has been fully reversed. If the transfer is only partially reversed, this attribute will still be false.
- **status** (*CharField*) – Current status of the transfer (`paid`, `pending`, `canceled` or `failed`). A transfer will be `pending` until it is submitted, at which point it becomes `paid`. If it does not go through successfully, its status will change to `failed` or `canceled`.
- **type** (*CharField*) – The type of this type of this transfer. Can be `card`, `bank_account`, or `stripe_account`.
- **amount_reversed** (*IntegerField*) – Amount in cents reversed (can be less than the amount attribute on the transfer if a partial reversal was issued).
- **balance_transaction_id** (ForeignKey to *BalanceTransaction*) – Balance transaction that describes the impact of this transfer on your account balance.
- **description** (*TextField*) – Internal-only description of the transfer
- **failure_code** (*CharField*) – Error code explaining reason for transfer failure if available. See Types of transfer failures for a list of failure codes.
- **failure_message** (*TextField*) – Message to user further explaining reason for transfer failure if available.
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the transfer in a structured format.
- **application_fee** (*CharField*) – Application fee
- **destination** (*CharField*) – ID of the bank account, card, or Stripe account the transfer was sent to.
- **destination_payment** (*CharField*) – If the destination is a Stripe account, this will be the ID of the payment that the destination account received for the transfer.
- **source_transaction** (*CharField*) – ID of the charge (or other transaction) that was used to fund the transfer. If null, the transfer was funded from the available balance.
- **statement_descriptor** (*CharField*) – Extra information about a transfer to be displayed on the user's bank statement.

1.23 Transfer Reversal

class `stripe_django.models.transfer_reversal.`**TransferReversal**(**args*, ***kwargs*)
Stripe Transfer Reversal object.

A previously created transfer can be reversed if it has not yet been paid out. Funds will be refunded to your available balance, and the fees you were originally charged on the transfer will be refunded. You may not reverse automatic Stripe transfers.

Parameters

- **id** (*AutoField*) – Id
- **amount** (*IntegerField*) – Amount reversed, in cents.
- **created** (*DateTimeField*) – Created
- **currency** (*CharField*) – Three-letter ISO code representing the currency of the reversal.
- **balance_transaction_id** (ForeignKey to *BalanceTransaction*) – Balance transaction that describes the impact of this reversal on your account balance.
- **metadata** (*JSONField*) – A set of key/value pairs that you can attach to a charge object. it can be useful for storing additional information about the transfer reversal in a structured format.
- **transfer_id** (ForeignKey to *Transfer*) – ID of the transfer that was reversed.

CHAPTER 2

History

CHAPTER 3

Developing and Testing

Our tests are inside `tests/`. Tests are implemented using pytest.

`make test` will create a tmux server on a separate `socket_name` using `$ tmux -L test_case`.

3.1 Install the latest code from git

To begin developing, check out the code from github:

```
$ git clone git@github.com:tony/stripe_django.git
$ cd stripe_django
```

Now create a virtualenv, if you don't know how to, you can create a virtualenv with:

```
$ virtualenv .venv
```

Then activate it to your current tty / terminal session with:

```
$ source .venv/bin/activate
```

Good! Now let's run this:

```
$ pip install -e .
```

This has `pip`, a python package manager install the python package in the current directory. `-e` means `--editable`, which means you can adjust the code and the installed software will reflect the changes.

```
$ stripe_django
```

3.2 Test Runner

As you seen above, the `stripe_django` command will now be available to you, since you are in the virtual environment, your *PATH* environment was updated to include a special version of `python` inside your `.venv` folder with its own packages.

```
$ make test
```

You probably didn't see anything but tests scroll by.

If you found a problem or are trying to write a test, you can file an issue on github.

3.2.1 Test runner options

```
$ py.test tests/test_common.py
```

will test the `tests/test_common.py` tests.

```
$ py.test tests/test_common.py::test_ignores_letter_versions
```

tests `test_ignore_letter_versions()` `tests/test_common.py`.

Multiple can be separated by spaces:

```
$ py.test tests/test_{window,pane}.py \
  tests/test_common.py::test_ignores_letter_versions
```

3.2.2 Visual testing

You can watch tmux testsuite build sessions visually by keeping a client open in a separate terminal.

Create two terminals:

- Terminal 1: `$ tmux -L test_case`
- Terminal 2: `$ cd` into the stripe_django project and into the `virtualenv` if you are using one, see details on installing the dev version of stripe_django above. Then:

```
    $ py.test tests/test_workspacebuilder.py
```

Terminal 1 should have flickered and built the session before your eyes. stripe_django hides this building from normal users.

3.3 Run tests on save

You can re-run tests automatically on file edit.

Note: This requires `entr(1)`.

Install entr. Packages are available available on most Linux and BSD variants, including Debian, Ubuntu, FreeBSD, OS X.

To run all tests upon editing any `.py` file:

```
$ make watch_test
```

Rebuild the documentation when an `.rst` file is edited:

```
$ cd doc
$ make watch
```

3.4 developer workflow

After you *Install the latest code from git*, when inside the stripe_django checkout:

```
$ stripe_django load .
```

this will load the `.tmuxp.yaml` in the root of the project.

```
session_name: stripe-django
start_directory: ./ # load session relative to config location (project root).
before_script: ./bootstrap_env.py # ./ to load relative to project root.
windows:
- window_name: stripe-django
  focus: True
  layout: main-horizontal
  options:
    main-pane-height: 35
  shell_command_before:
    - '[ -d .venv -a -f .venv/bin/activate ] && source .venv/bin/activate'
  panes:
  - focus: true
  - pane
  - make watch_flake8
  - make watch_test
- window_name: docs
  layout: main-horizontal
  options:
    main-pane-height: 35
  start_directory: docs/
  shell_command_before:
    - '[ -d ../.venv -a -f ../.venv/bin/activate ] && source ../.venv/bin/activate'
  panes:
  - focus: true
  - pane
  - make serve
  - make watch
```

3.4.1 Travis CI

stripe_django uses travis-ci for continuous integration / automatic unit testing.

stripe_django is tested against tmux 1.8 and the latest git source. Interpretters tested are 2.6, 2.7 and 3.3. The travis build site uses this .travis.yml configuration:

```
language: python

sudo: false
python:
  - 2.7
  - 3.4
  - 3.5
before_install:
  - export PIP_USE_MIRRORS=true
  - pip install --upgrade pytest  # https://github.com/travis-ci/travis-ci/issues/4873
  - pip install --upgrade pip wheel virtualenv setuptools
  - pip install coveralls
install:
  - pip install -e .
script: coverage run --source=stripe_django setup.py test
after_success:
  - bash <(curl -s https://codecov.io/bash)
```

l

s