Stripe REST API Reference

Stripe · https://namazustudios.com/docs/add-ons/commerce/stripe/stripe-rest-api-reference/

This page documents every REST endpoint exposed by the Stripe Element. All paths are relative to the configured RS root, `dev.getelements.elements.element.rs.root` (default `/element/stripe/api`) — so `/stripe/payment-intent` below means `/element/stripe/api/stripe/payment-intent` on a default deployment. Every endpoint except the webhook receiver requires the `session_secret` security scheme. The full OpenAPI document is served at `/api/rest/openapi.json` once the Element is running.

Webhook

Method	Path	Auth	Description
`POST`	`/stripe/webhook`	None (signature-verified)	Receives and verifies a Stripe webhook, then publishes typed and raw internal events. See Stripe Webhooks and the Typed Event Bus.

Customers

Method	Path	Request	Response	De
`PATCH`	`/stripe/customer/{customerId}`	`UpdateCustomerRequest`	`204 No Content`	Up cu er di na fie le un

Method	Path	Request	Response	De
`GET`	`/stripe/customers/search?key=&value=`	—	`CreateCustomerResponse`	Fi cu a ke pa Re 4(m k fo cr
`POST`	`/stripe/customer/{customerId}/portal-session?returnUrl=`	—	`CreatePortalSessionResponse`	Cr si St Cu Po se

`UpdateCustomerRequest(String email, String name)`. `CreateCustomerResponse(String customerId)`. `CreatePortalSessionResponse(String url)`.

Payments

Method	Path	Request	Response	Description
`POST`	`/stripe/payment-intent`	`CreatePaymentIntentRequest`	`CreatePaymentIntentResponse`	Creates a PaymentIntent for a one-off charge and returns its client secret.

`CreatePaymentIntentRequest(long amount, String currency, String customerId, String description, Map<String,String> metadata, Boolean automaticPaymentMethods, String setupFutureUsage, String idempotencyKey)` — `amount` is in the currency's smallest unit (e.g. cents); `customerId` may be `null` for guest checkouts; `setupFutureUsage` set to `"off_session"` saves the payment method for later charges. `CreatePaymentIntentResponse(String paymentIntentId, String clientSecret)`.

Subscriptions

Method	Path	Request
`POST`	`/stripe/customer/{customerId}/subscription`	`CreateSubscriptionR`
`GET`	`/stripe/subscription/{subscriptionId}`	—

Method	Path	Request
`DELETE`	`/stripe/subscription/{subscriptionId}`	—
`GET`	`/stripe/customer/{customerId}/subscriptions?status=&limit=10&startingAfter=`	—

`CreateSubscriptionRequest(String priceId, String description, Map<String,String> metadata, String idempotencyKey)` — factory `of(priceId)` covers the common case. `SubscriptionStatusResponse(String subscriptionId, String status, String currentPeriodEnd)`.
`SubscriptionListResponse(List<SubscriptionStatusResponse> subscriptions, boolean hasMore, String nextCursor)` — pass `nextCursor` back as `startingAfter` to page.

Checkout and Invoices

Method	Path	Request	
`POST`	`/stripe/checkout-session`	`CreateCheckoutSessionRequest`	
`GET`	`/stripe/customer/{customerId}/invoices?limit=10&startingAfter=`	—	

`CreateCheckoutSessionRequest(String customerId, String priceId, String successUrl, String cancelUrl, String mode, String idempotencyKey, Map<String,String> metadata)` — `mode` defaults to `subscription` if `null`; use `payment` for a one-off charge. `CreateCheckoutSessionResponse(String sessionId, String url)`. `InvoiceSummary(String id, String subscriptionId, Long amountPaid, String currency, String status, String createdAt)` — `status` is one of `draft`, `open`, `paid`, `uncollectible`, `void`.

Catalogue: Products and Prices

Method	Path	Response	Description
`GET`	`/stripe/products?active=true&limit=100`	`List<ProductSummary>`	Lists products from the St
`GET`	`/stripe/prices?productId=&active=true&limit=100`	`List<PriceSummary>`	Lists prices, optionally filte memory (TTL: `dev.getelements.element`).

Method	Path	Response	Description
`GET`	`/stripe/prices/{priceId}`	`PriceSummary`	Retrieves a single price by its product ID.

`ProductSummary(String id, String name, String description, boolean active, PriceSummary defaultPrice)` — `active: false` means archived. `PriceSummary(String id, String productId, String nickname, Long unitAmount, String currency, String type, String interval)` — `unitAmount` is `null` for usage-based prices; `type` is `one_time` or `recurring`; `interval` is `day`, `week`, `month`, or `year` for recurring prices, otherwise `null`.

Billing Meters

Method	Path	Request	Response	Description
`POST`	`/stripe/meter-event`	`RecordMeterEventRequest`	`204 No Content`	Reports a usage event to a Stripe Billing Meter. Throws `NoSuchMeterException` (surfaced as an error response) if Stripe has no active meter for the given event name.

`RecordMeterEventRequest(String customerId, String eventName, BigDecimal value, String idempotencyKey)` — `value` must be greater than zero; a `long`-value convenience constructor exists for whole-unit usage. The `idempotencyKey` is used both as Stripe's own meter-event deduplication identifier and as the HTTP idempotency key, so retrying with the same key never double-reports usage.

Note

`listMeters` and both overloads of `resolvePriceForMeterEventName` — the meter-to-price join described in Stripe → Products, Prices, and Billing Meters — are `StripeService` methods only. There is no REST endpoint for them; call them from an Element that depends on the Stripe `api` module.

Configuration and Event Log (Superuser)

Method	Path	Request	Response	Description
`GET`	`/stripe/config`	—	`StripeConfig`	Returns the effective Stripe credentials with values masked (e.g. `••••1234`).
`PUT`	`/stripe/config`	`StripeConfig`	`{"saved":true}`	Persists Stripe credentials to the database, overriding the deployment's attributes.
`GET`	`/stripe/events?type=&limit=20&offset=0`	—	`StripeEventLogResponse`	Lists received webhook events, newest first, with type filtering and offset pagination.

`StripeConfig(String apiKey, String webhookSecret)`. `StripeEventLogResponse(List<StripeEventLogEntry> events, long total, boolean hasMore)`, where `StripeEventLogEntry(String stripeEventId, String eventType, String receivedAt)`. See Configuring the Stripe Element for credential precedence.

Service-Only Methods

Not every `StripeService` capability is exposed over REST. The following are reachable only by injecting `StripeService` from another Element that depends on Stripe's `api` module — there is deliberately no HTTP surface for them, either because they're building blocks for server-side flows (customer/payment-method setup ahead of a charge) or because they're only meaningful to trusted server code (recording a receipt from inside the webhook handler):

- `createCustomer(email, name, orgId)`
- `createSetupIntent(customerId)`
- `listPaymentMethods(customerId)` / `hasPaymentMethod(customerId)`
- `getProduct(productId)`
- `listMeters(activeOnly, limit)`
- `resolvePriceForMeterEventName(eventName)` and `resolvePriceForMeterEventName(eventName, subscriptionId)`
- `recordPaymentReceipt(transactionId, amount, currency, userId)` — called internally by the webhook handler; not intended to be called from arbitrary server code

---

Related Pages

- Stripe — Element overview and core concepts
- Configuring the Stripe Element — attributes and credential precedence
- Stripe Webhooks and the Typed Event Bus — the webhook endpoint and typed events in detail