Configuring the Stripe Element

Stripe · https://namazustudios.com/docs/add-ons/commerce/stripe/configuring-the-stripe-element/

The Stripe Element is configured through standard Elements deployment attributes, with two of them — the API key and webhook signing secret — overridable at runtime from a superuser admin panel without a rebuild or redeploy. This page covers the available attributes, how credential precedence works, running multiple environments safely, and the Maven coordinates other Elements need to depend on it.

Attributes

Attribute	Default	Description
`dev.getelements.elements.stripe.api.key`	*none — required*	Stripe secret API key (`sk_live_...` or `sk_test_...`). Marked sensitive.
`dev.getelements.elements.stripe.webhook.secret`	*none — required*	Stripe webhook signing secret (`whsec_...`), shown by Stripe when you register a webhook endpoint. Marked sensitive.
`dev.getelements.elements.element.rs.root`	`/element/stripe/api`	REST API root path for every endpoint this Element exposes.
`dev.getelements.elements.auth.enabled`	`true`	Enables the Elements session auth filter for this Element's endpoints.

Attribute	Default	Description
`dev.getelements.elements.stripe.price.cache.ttl.ms`	`300000` (5 minutes)	How long `listPrices` and the catalogue-wide `resolvePriceForMeterEventName` cache their results in memory before re-querying Stripe.

Setting Attributes

Deployed instances — set attributes from the admin panel under Element Management. Select the deployment and edit its attributes directly; no rebuild is required.

Local development — set attributes as system properties when launching the `debug` module:

```
mvn -pl debug exec:java
  -Ddev.getelements.elements.stripe.api.key=sk_test_YOUR_KEY
  -Ddev.getelements.elements.stripe.webhook.secret=whsec_YOUR_SECRET
```

Defaults can also be baked into `element/src/main/elm/dev.getelements.element.attributes.properties` so they're packaged into the `.elm` archive itself — these are still overridden by anything set at the deployment level.

Credential Precedence and the Admin Panel

The API key and webhook secret can also be set from the "Stripe" tab in the superuser admin dashboard, or directly via `PUT /stripe/config`. Values saved this way are persisted to a MongoDB document and take precedence over the deployment attributes whenever they're non-blank — `GET /stripe/config` returns the effective values with everything but the last four characters masked (e.g. `••••1234`), so the full secret is never re-displayed once saved.

This lets an operator rotate a compromised key or point a deployment at a different Stripe account without touching deployment attributes or redeploying. If no override has been saved, the Element falls back to the injected attribute values above.

Multi-Environment Deployments

Run separate environments (e.g. `MyGame_Dev`, `MyGame_Staging`, `MyGame_Prod`) by deploying a distinct instance of this Element per environment, each with:

- Its own Stripe API key and webhook secret, configured independently through the Stripe → Configuration admin tab
- Its own MongoDB database (or at minimum a separate MongoDB instance), so configuration and event-log data are naturally isolated with no extra namespacing required

In the Elements platform this maps directly to creating one application deployment per environment. Because each deployment connects to its own database, the DAO layer needs no special configuration to achieve isolation.

Warning

Do not share one MongoDB database between multiple deployments of this Element. The configuration and webhook event-log collections use fixed, unnamespaced names (`stripe_config` and `stripe_event_log`), so sharing a database lets one deployment's credentials and event log bleed into another's.

Build and Run

```
# Build everything (requires Java 21 + Maven)
mvn install

# Start local MongoDB
docker compose -f services-dev/docker-compose.yml up -d

# Run locally
mvn -pl debug exec:java
```

The admin UI panel is only built under the `build-ui` Maven profile (`mvn install -Pbuild-ui`); its React bundle is copied into the `.elm` archive's `ui/` directory and registers a "Stripe" tab (icon: `CreditCard`, route: `stripe`) in the superuser dashboard.

Maven Coordinates

Deploy the `.elm` archive built by the `element` module. Other Elements that call `StripeService` directly (rather than only consuming its events or REST API) should also depend on the `api` module, provided at compile time:

```
<!-- .elm archive (for deployment) -->
<dependency>
    <groupId>dev.getelements.elements.stripe</groupId>
    <artifactId>element</artifactId>
    <version>1.0.3</version>
    <type>elm</type>
</dependency>

<!-- API interfaces (for other Elements that depend on this one) -->
<dependency>
    <groupId>dev.getelements.elements.stripe</groupId>
    <artifactId>api</artifactId>
    <version>1.0.3</version>
    <classifier>dev.getelements.elements.stripe.api</classifier>
    <scope>provided</scope>
</dependency>
```

Integration Tests

Integration tests exercise the real Stripe API in test mode. Credentials come from environment variables so CI can inject them as secrets:

Env var	Maven property	Purpose
`STRIPE_TEST_API_KEY`	`stripe.test.apiKey`	Stripe test-mode secret key
`STRIPE_TEST_CUSTOMER_ID`	`stripe.test.customerId`	Existing test-mode customer to reuse
`STRIPE_TEST_PRICE_ID`	`stripe.test.priceId`	Existing test-mode price to reuse, instead of creating a new Product + Price every run

```
export STRIPE_TEST_API_KEY=sk_test_YOUR_KEY
mvn verify -pl integration-test
```

Each variable can also be overridden per run with the matching `-Dstripe.test.*` system property. Subscription tests create and tear down a real subscription automatically. Webhook tests generate and verify their own HMAC signatures against a fixed, non-secret test key, and run without any network connection or Stripe credentials.

Related Pages

- Stripe — Element overview and core concepts
- Stripe Webhooks and the Typed Event Bus — webhook Dashboard setup and the typed event list
- Stripe REST API Reference — full REST endpoint reference