

Mapping chargeback webhooks

Forter works with many payment providers (e.g., Stripe, PayPal, Adyen). Each one sends chargeback webhooks in different formats. To automate onboarding and minimize manual mapping, we want to give our merchants a way to transform those webhooks into Forter's normalized chargeback schema using a declarative mapping engine (e.g., JSONata).

You'll prototype a webhook receiver that transforms raw webhook payloads into Forter's internal chargeback format. Your first example will only map a single webhook, **but the goal is to design a component that will be able to map multiple types of webhooks and will be easily extensible**.

What you'll build

Choose **one** payment provider that sends webhooks for new chargebacks. Create a service that maps the webhook into Forter's chargeback object that can be used for downstreaming. For the sake of simplicity, you can simply return the mapped object as the API response.

Endpoint and Request Body

POST /webhook

```
{
  "payload": { /* raw webhook from a payment provider */ },
}
```

Response

```
{
  "result": { /* transformed payload */ }
}
```

Sample Forter Chargeback Schema (simplified)

```
{
  "type": "object",
  "required": ["transaction_id", "reason", "currency", "amount"],
  "properties": {
    "transaction_id": { "type": "string" },
    "reason": { "type": "string" },
  }
}
```

```
"currency": { "type": "string" },
"amount": { "type": "number" },
"provider": { "type": "string" }
}
}
```

Requirements

1. A working `/webhook` endpoint that applies a JSONata (or other) expression to the payload.
2. JSON validation of the result against a provided Forter chargeback schema, as described above.
3. One test case showing a Stripe or PayPal chargeback webhook mapped into the Forter format.

Design Prompts (Answer in README or design doc)

We're hiring a **Staff Engineer**, so beyond the code, we'd love to see how you think about:

1. **Extensibility:** How would you design this so we can onboard new providers (Square, Braintree, etc.) with minimal engineering effort?
2. **Developer Experience:** If a merchant needed to test their mapping before going live, what tooling would you expose?
3. **Safety and Maintainability:** How would you sandbox mapping logic, manage versions, or detect breaking changes?
4. **GTM Integration:** How could this system help shorten merchant onboarding time? What key metrics would you track?
5. **Future enhancements:** What will be the next milestones of this project? How will you decide what would be the next steps after the initial version of it?

Time Expectations

We don't want you to over-invest.

- Total time: ~3 hours
 - ~1.5h: functional prototype
 - ~1h: design doc or README
 - ~30m: polish & review

Please upload it to a public/private GitHub account and provide a simple README on how to run the server and steps to e2e test the API. Please let us know if you have any questions along the way.