SENTINAL

Digital Tax Compliance Platform

FUNCTIONAL API SPECIFICATION

Payment Processor Integration Reference

Version 1.5.8 — August 2026

Ghana Revenue Authority (GRA) Implementation

Audience & scope

This document is intended for Payment Processors integrating with the Sentinal platform in Ghana. It describes the authenticated API surface a Payment Processor consumes: the access token, merchant registration, the transactional lifecycle, VAT rules retrieval, and the webhook events delivered to your platform.

PRIVATE AND CONFIDENTIAL

Document Information

Property	Value
Title	Sentinal Functional API Specification
Version	1.5.8
Status	Released
Deployment	Ghana — Ghana Revenue Authority (GRA)
Audience	Payment Processors
Classification	Private and Confidential

Environments

All Sentinal URLs are reachable only over the IPSEC / site-to-site VPN to the GRA network (see Appendix B). They are not exposed on the public internet.

Environment	API (Backend)	Dashboard (Front End)	IP Address
UAT	https://uat-sentinal-api.gra.gov.gh	https://uat-ecom.gra.gov.gh	10.65.1.5
Production	https://prod-sentinal-api.gra.gov.gh	https://ecom.gra.gov.gh	Not configured yet

> **Production not yet available**
> As of August 2026 (version 1.5.8), the Production environment is not yet implemented. The Production URLs above are provided for reference only. Integrators will be notified separately once Production is available and its IP address is configured. Integrate against UAT in the meantime.

Change Log

Changes introduced in version 1.5.8, a correctness revision: every endpoint, header, field table and signature rule was re-verified line by line against the deployed implementation, and each row below records a place where versions 1.5.6 and 1.5.7 did not match it. The 1.5.7 entries are retained beneath for continuity. Earlier revision history is available on request.

Version	Date	Section	Change
1.5.8	August 2026	3.3, 4	CORRECTION — the RSA access-token signature is submitted as LOWERCASE HEX, not base64. Versions 1.5.6 and 1.5.7 specified base64; the platform decodes X-SIGNATURE as hex and a base64 value fails verification with HTTP 401. This is the most likely cause of a token request that is correctly signed and still rejected.
1.5.8	August 2026	3.4	Clarified that requestPath is the full request path exactly as transmitted, including the /v1 prefix and any query string.
1.5.8	August 2026	3.2.3	Added the delegation outcome table. An unrecognised X-ISSUER-ID and an issuer that has not appointed the agent are distinguishable, returning 404 and 403 respectively, correcting the previous statement that they were not; the 400 and 409 outcomes are documented for the first time.

Version	Date	Section	Change
1.5.8	August 2026	4	CORRECTION — an invalid or expired X-TIMESTAMP on the token endpoint returns HTTP 401, not 400, and every token failure returns the same generic message.
1.5.8	August 2026	5.1	CORRECTION — where a merchant matches neither an MCC fee nor a classification fee, the registration response returns COLLECTED at the country default rate. The previous statement that there is no country-default fallback describes the PSP-side snapshot rule (7.3 rule 5), not this response.
1.5.8	August 2026	5.2.1	CORRECTION — exempt merchants are distributed NATIONALLY: every approved merchant carrying a VAT exemption or a NOT_COLLECTED status appears in every Payment Processor snapshot. Only statusOverrides are Payment-Processor-scoped.
1.5.8	August 2026	5.3 (new)	Added POST /v1/client/statusChanges, a live merchant-status polling endpoint omitted from all previous revisions. It is the supported alternative for Payment Processors that cannot consume the merchant.vat_updated webhook.
1.5.8	August 2026	6.1	Added validation rules that were enforced but undocumented: paymentMethods must sum exactly to totalAmount, the submission window is the transaction calendar day in GMT, and a repeated ref is idempotent rather than an error. DECLINED added to the response status enumeration.
1.5.8	August 2026	6.2	CORRECTION — the response is the full stored transaction record and its identifier field is _id, not transactionId. An unknown transaction currently returns HTTP 500.
1.5.8	August 2026	6.3	Added the response table and error-code mapping. creditTime must carry a numeric offset (+HH:MM / -HH:MM); a Z suffix is rejected.
1.5.8	August 2026	6.4, 6.5	Added the preconditions governing both endpoints: only a COLLECTED transaction may be reversed, and only once. Documented the matching tolerances on the verified* fields and the generated ref format.
1.5.8	August 2026	6.5	CORRECTION — exchangeRateUsed is OMITTED for a GHS request, not returned as 1; the worked example response is corrected. Added a warning that /chargeBack is a different endpoint from /chargeback.
1.5.8	August 2026	6.6	Added the refund response and validation tables, absent from previous revisions.
1.5.8	August 2026	7.1, 7.2	Documented the collectionRules item shape and named the exact fields covered by rulesetHash, without which the integrity check cannot be reproduced.
1.5.8	August 2026	8.2, 8.4	CORRECTION — webhook payloads are delivered inside a success / message / content / metadata envelope, so the merchant.vat_updated fields sit under content, not at the top level. Defined the path segment used in the signature.
1.5.8	August 2026	11.1	CORRECTION — error responses may carry a non-null content object holding a code and a hint.
1.5.8	August 2026	5.2, 5.2.1	Rewritten around the confirmed design decision that approved, known and VAT-exempt merchants are distributed to every Payment Processor snapshot. An exempt merchant is now identifiable before its first transaction, so the section states what the national list already gives you and confines

Version	Date	Section	Change
			the residual discrepancy to status overrides, central matching and stored attributes.
1.5.8	August 2026	Appendices C—F	Each appendix now carries its title in the running page header, and the artwork occupies the full page rather than sitting beneath a repeated heading.
1.5.8	August 2026	3.2.2, 11.2	Documented that an absent authentication header returns 400 on endpoints that declare a header schema and 401 on those that do not, so the status code does not distinguish a missing header from a rejected credential. Confirmed by testing against a running instance.
1.5.8	August 2026	6.1, 11.3, 11.4	Qualified the idempotency guarantee. A repeated ref never creates a second transaction, but an IMMEDIATE retry can be answered with HTTP 500 instead of the original record, because the first write has not yet settled. Confirmed by testing against a running instance. Retry guidance updated to forbid instant resubmission and to state that a 500 here is retryable and must not be answered with a new ref.
1.5.8	August 2026	5.1, 6.1, 7.1, 7.3	CORRECTION — vatRate and defaultVatRate are delivered as PERCENTAGES (20 for 20%), not as decimal fractions. Versions 1.5.6 and 1.5.7 described them as decimals and used 0.2 in worked examples. A processor following the old text over-collects VAT by a factor of one hundred, silently. VAT arithmetic restated as amount x vatRate / 100.
1.5.8	August 2026	Appendix F	Placeholder replaced with a structurally accurate sample of the GET /v1/psp/vatRules response, marked throughout as illustrative with invented values.
1.5.8	August 2026	8.3, 8.6 (new)	Narrowed the webhook section to merchant.vat_updated, the only event in scope for Ghana, and documented the subscription-management endpoints Payment Processors use to create, test, rotate and disable their own subscription. Warned that omitting the events field on subscribe registers EVERY platform event rather than none.
1.5.8	August 2026	6.8 (new)	Added POST /v1/rates, the reference exchange-rate endpoint underpinning MCA VAT determination, previously undocumented. Its response is not wrapped in the standard envelope.
1.5.8	August 2026	Appendix A	Endpoint summary completed, and endpoints outside the Ghana integration named explicitly so they are not built against by mistake.
1.5.8	August 2026	Appendices C—E	Flow diagrams inserted; the appendices are no longer placeholders and are laid out landscape so the sequence diagrams are legible.
1.5.7	August 2026	5.2 (new)	Added Merchant Pre-Registration and whitelisting in the VAT Rules Engine, covering the VAT discrepancy that arises when a merchant already exists in Sentinal, the recommended stabilisation period, and the optional merchant export for populating a local rules engine.
1.5.7	August 2026	2.4 (new)	Added Settlement Agents: the National Collection Agent principal (GhIPSS operates as one) and the Delegated Processor Agent principal (Third Party Issuer Processors), with the endpoints and appointment model applying to each.
1.5.7	August 2026	3.2.2, 3.2.3 (new), 3.4	Added the X-ISSUER-ID delegation header — the target issuer's GIP code, or bank code as fallback — used by both

Version	Date	Section	Change
			agent principals, and documented how it is bound into the HMAC string to sign.
1.5.7	August 2026	6.5	Chargebacks expanded from a summary paragraph to a full specification: request and response field tables, the D+1 / D+12 timing and status matrix, permitted reason codes, and worked request/response examples.
1.5.7	August 2026	6.6	Refund restated: build the integration now, but the refund process remains manual until the GRA working committee finalises it. The endpoint contract is stable and will not change when the automated process is enabled.
1.5.7	August 2026	5.1	Card processors directed to populate registrationNumber with DE 42, and warned that merchantId is not a match key. Confirmed that Ghana merchants are registered on a VAT-exclusive basis.
1.5.7	August 2026	6.1	Clarified that the response totalAmount is the submitted amount echoed back and is NOT VAT-inclusive; added the Ghana amount-arithmetic table and the rule that the cardholder total is totalAmount + vatAmount, computed by the Payment Processor. merchantAmount and vatAmount descriptions corrected.
1.5.7	August 2026	6.1	transactionFee must be OMITTED for Ghana — the validation tests presence, not value, so sending 0 is rejected. penaltyApplied and penaltyAmount marked as not in use for Ghana.
1.5.7	August 2026	6.1	issuingBank documented as free text to a maximum of 8 characters, not validated against a register.
1.5.7	August 2026	6.4, 6.6	Chargeback and refund restated: build them in this cycle, as the contracts will not change when the GRA committee finalises the internal process, so no further code changes will be needed.
1.5.7	August 2026	7.1, 7.4	Snapshot refresh cadence documented per environment — production expires on the calendar month boundary, UAT sooner for testing. Removed the max-age=3600 statement, warned against a hard-coded 30-day cache, and set out the consequence of not consuming the merchant.vat_updated webhook.
1.5.7	August 2026	2.1, 7.3	Added the cross-border precondition: scope is determined before the ruleset is evaluated, which is why localMerchantVatExempt needs no processor-side logic. Added the aggregator exception — a transaction registered against an aggregator's Ghanaian details presents as domestic but is in scope, must still be submitted by the upstream provider, and carries its VAT obligation at the aggregator.
1.5.7	August 2026	7.3	Rule 2 (merchantOverrides.statusOverrides) confirmed as in scope, withdrawing an earlier remark to the contrary. Processors are advised to implement it: an MCC may be out of scope as a whole while individual merchants carrying it remain in scope.
1.5.7	August 2026	7.3	Added rule applicability by processor type: rule 3 (mccRules) is the classification tier for card processors and rule 4 (classificationRules) is out of scope for them; rule 4 remains required for processors that hold merchants against a Type.

Version	Date	Section	Change
1.5.7	August 2026	7.3	Added the collection-status action table: collect only on COLLECTED, and submit POTENTIALLY_COLLECTED transactions to verifyTransaction for reconciliation.
1.5.7	August 2026	Appendix F (new)	Added an appendix for a sample VAT Rules ruleset.
1.5.7	August 2026	Appendices C–E	Flow diagrams updated.
1.5.6	July 2026	—	Previous released version.

Table of Contents

This table updates when opened in Microsoft Word (right-click → Update Field, or press F9). Refresh it before circulating or printing: the entries and page numbers are generated by Word, not by this document.

1. Introduction

1.1 Background

Sentinal is a digital tax compliance platform that assists the Ghana Revenue Authority (GRA) to calculate and collect Value Added Tax (VAT) on qualifying electronic transactions in real time. It integrates with Payment Processors outside of the authorisation flow, without blocking settlement.

> **Sentinal does not interfere with your authorisation process**
> Sentinal sits OUTSIDE the authorisation flow. It is never in the authorisation path: it does not authorise, decline, gate, delay, or modify a transaction, and it never blocks settlement.
> Authorisation is completed entirely by your existing systems and the payment rails. Only after authorisation has completed does the Payment Processor submit the transaction to Sentinal for VAT determination and recording (Section 6.1). A Sentinal outage or slow response can never prevent a transaction from being authorised or settled.

This document is the functional API specification for Payment Processors integrating with the Sentinal platform in Ghana. It covers authentication and request signing, the access-token exchange, merchant registration and pre-registration, the transactional lifecycle (verification, decline, chargeback, refund, and credit notification), VAT rules retrieval, and the webhook events delivered to your platform.

> **Scope — cross-border digital services**
> Sentinal applies to cross-border, digitally-delivered services consumed in Ghana. In scope: a buyer using a credit, debit, or prepaid card issued by a Ghanaian issuer — or a Ghana eWallet / mobile-money account — to purchase a service that is delivered digitally.
> Out of scope: hard or tangible goods, for which VAT is collected through existing Customs processes. Requests reflect this: the issuing instrument and issuer are Ghanaian (issuerCountryCode = GH; countryCode = GH; currency = GHS), and the transaction is for a digitally-delivered service.

1.2 Objective

To provide Payment Processors with a complete and accurate integration reference for the Ghana deployment. Payment Processors should treat this document, together with the live VAT rules snapshot (Section 7), as authoritative for request and response formats, validation rules, and collection-status semantics.

1.3 Terminology and Definitions

Term	Definition
Payment Processor	Used throughout as the umbrella term for any entity that integrates with Sentinal and is mandated to apply VAT and remit it to the GRA — Banks (Issuers), TPPs, Payment Service Providers (PSPs), Fintechs, eWallet providers, and Mobile Money providers (see Section 2.1).
TPP	Third Party Transaction Processor.
PSP	Payment Service Provider — one category of Payment Processor.
Merchant / Client	A business registered under a Payment Processor for VAT determination and collection.
GRA	Ghana Revenue Authority — the tax authority for Ghana.

Term	Definition
Issuer	The institution issuing the payment instrument and authorising the transaction.
Acquirer	The institution managing the merchant relationship and acquiring system.
GHS	Ghanaian Cedi — the local settlement currency (ISO 4217).
GH	ISO 3166-1 alpha-2 country code for Ghana; the only country code accepted by this deployment.
GMT	Greenwich Mean Time (UTC+0) — Ghana's timezone. Timestamps are ISO 8601.
VAT	Value Added Tax applied to qualifying electronic transactions under Ghanaian law.
Collection Status	Whether VAT is collected for a transaction: COLLECTED, NOT_COLLECTED, or POTENTIALLY_COLLECTED.
Snapshot	A point-in-time, signed copy of the active VAT ruleset for a Payment Processor (Section 7).
Snapshot Signature	A snapshot's unique identifier: SNT-GH-YYYYMMDD-{16 hex, uppercase}.
RSA Signature	Asymmetric SHA-256 signature (Payment Processor private key) used for the access-token request.
HMAC	Hash-based Message Authentication Code (HMAC-SHA256) used to sign authenticated requests and outbound webhooks.
Access Token	Short-lived (1 hour) JWT bearer token obtained from the token endpoint.
RRN	Retrieval Reference Number — ISO 8583 field 37.
MCC	Merchant Category Code — ISO 18245 four-digit code.
Type / Classification	A Sentinal merchant category that maps to a VAT rate and default collection status (Section 9).
MCA	Multi-Currency Acquiring — a transaction whose currency differs from the local currency; VAT is determined using Sentinal's reference exchange rate (Section 6.1).
Foreign Merchant	A merchant originating outside Ghana, or acquired via a foreign acquirer (Acquirer Country ≠ GH); the primary subject of VAT determination.
Switching	The institution interconnecting the issuer's and acquirer's systems.
Credit Status	The state of a VAT credit for a settlement: Pending, Settled, or Reversed.
VAT Settlement	Recording and segregating collected VAT into the VAT account, typically on the next business day.
VAT Submission	Reporting and remitting the collected VAT data and funds to the GRA in accordance with applicable regulations.

2. Getting Started

2.1 Who Must Integrate

In this document, "Payment Processor" is the umbrella term for any entity mandated to apply VAT to in-scope transactions and remit it to the GRA. This includes Banks (Issuers), Third Party Transaction Processors (TPPs), Payment Service Providers (PSPs), Fintechs, eWallet providers, and Mobile Money providers.

Obliged to integrate

- Issuers and payment providers whose Ghana-issued instruments (card, eWallet, mobile money) are used to purchase in-scope cross-border digital services.
- Aggregators — entities that take payments locally via cards and mobile money under their OWN business details and settle offshore (foreign) merchants outside the card/payment rail. Because the underlying merchant is not visible to the issuer, the aggregator must report these transactions to Sentinal. See the Aggregator Flow in Appendix E.

Not required to integrate

- Fintechs operating purely as a merchant-side payment gateway that pass the ACTUAL merchant details through to the card schemes. These transactions are captured through the Bank (Issuer) flow and must not be reported again. See the Bank Issuer Flow in Appendix D.

> **Aggregator flows produce transactions that look domestic — submit them anyway**
> Where an aggregator is in the flow, the upstream provider registers the merchant using the AGGREGATOR's Ghanaian details. The transaction therefore presents to that upstream provider as a local, domestic one even though the underlying supply is cross-border.
> Upstream providers must still submit these transactions to Sentinal. Do not filter them out as domestic — see the exception in Section 7.3. The obligation to apply and remit the VAT on the underlying supply rests with the aggregator, which integrates separately for that purpose.

> **Merchant communications**
> The GRA has established a committee overseeing communications to merchants, to ensure merchants make the relevant changes on their side. Coordinate merchant-facing messaging and timelines with this committee.

2.2 Onboarding

To begin onboarding, complete the Sentinal onboarding form:

https://forms.office.com/pages/responsepage.aspx?id=5p-I_-4bwEO2pv6bFJ5dfaf5CpnjR3NJst9sMuBcDCpUMElHOVA3R1VCM05JT0VFQldLRFk4VkNCRi4u&origin=lprLink&route=shorturl

As part of onboarding you must generate a 2048-bit RSA key pair and share the PUBLIC key with Sentinal over a secure channel. Never send keys by email or chat. Acceptable channels:

- A shared item in a 1Password or Bitwarden vault; or
- A SharePoint link restricted so that only the Sentinal Support email address can access it.

Keep your private key secret
Only the PUBLIC key is shared. Never transmit your private key. Sentinal uses your public key to verify your token-request signatures (Section 3.3).

2.3 Integration Overview

A Payment Processor integrates with Sentinal over HTTPS using JSON. The typical lifecycle is: obtain a short-lived access token (RSA-signed), register merchants, retrieve the VAT rules snapshot, and submit transactions and dispute events — each authenticated request carrying an HMAC signature. Sentinal notifies the Payment Processor of merchant and dispute events by signed webhook.

Sentinal operates outside the authorisation flow and never blocks settlement. Once your systems have completed authorisation, the Payment Processor submits the transaction to Sentinal for VAT determination and recording, and Sentinal returns the applicable VAT treatment. Per-operation sequence diagrams are in Appendix C; the full Bank Issuer and Aggregator flows are in Appendices D and E.

At a glance
- Position: OUTSIDE the authorisation flow — Sentinal never authorises, declines, delays, or blocks a transaction or its settlement.
- Transport: HTTPS, JSON request/response bodies.
- Token auth: RSA-SHA256 signature (Section 3.3).
- Request auth: HMAC-SHA256 signature over method, path, token, body hash, and timestamp (Section 3.4).
- Amounts: submitted exclusive of VAT (Ex-VAT) after authorisation; Sentinal applies VAT (Section 6.1).
- VAT determination: apply the per-processor snapshot (Section 7); the snapshot is authoritative.
- Notifications: signed webhooks (Section 8).

2.4 Settlement Agents

Two principal types act on behalf of an issuing Payment Processor rather than on their own account. Both authenticate with their own credentials, using the same access token and HMAC scheme as a direct Payment Processor, and both name the issuer they are acting for in the X-ISSUER-ID header (Section 3.2.3). Sentinal attributes the call to that issuer, so the transaction, the merchant and the VAT position belong to the issuer.

Principal	Acts for	Scope of access
National Collection Agent	Any issuer in the market	Settlement confirmation only (Section 2.4.1).
Delegated Processor Agent	Only issuers that appointed it	The transactional and registration surface (Section 2.4.2).

These are principal types, not a separate API
An agent calls the endpoints documented in Sections 5 and 6 with the same request and response contracts as a direct Payment Processor. The only additions are the X-ISSUER-ID header and the restriction on which endpoints each agent may call. No agent-specific endpoint exists.

2.4.1 National Collection Agent

A National Collection Agent confirms VAT settlement on behalf of the market as a whole. It is nationally scoped: it may act for ANY issuer and requires no per-issuer appointment, because a national collection agent settles

across the market rather than for a named set of institutions. Its reach is constrained by endpoint instead of by appointment.

GhIPSS (Ghana Interbank Payment and Settlement Systems) operates as a National Collection Agent under this deployment.

Property	Detail
Appointment	None required. The agent may name any issuer in X-ISSUER-ID.
Permitted endpoint	POST /v1/transaction/creditInformation (Section 6.3) — settlement confirmation.
All other endpoints	Refused with HTTP 403. Verification, registration, decline, chargeback and refund are not available to this principal.
X-ISSUER-ID	Required on every call. The issuer's GIP code as registered with Sentinal (Section 3.2.3).

2.4.2 Delegated Processor Agent

A Delegated Processor Agent transacts on behalf of the issuers that have appointed it. Unlike a National Collection Agent it is not nationally scoped: it may act only for issuers whose delegated-processor list names it, and a call naming any other issuer is refused before the merchant is resolved.

Third Party Issuer Processors — institutions that operate the issuing or transaction-processing function for one or more banks — fall under this principal type.

Property	Detail
Appointment	Per-issuer. Each issuer that wishes to be processed by the agent must have the agent added to its delegated-processor list by Sentinal.
Permitted endpoints	verifyTransaction, verifyTransactionStatus, declineTransaction, chargeback, refund, creditInformation, registerClient and statusChanges.
Not permitted	Any endpoint outside that list is refused with 403. An issuer that has not appointed the agent is refused with 403; an issuer identifier that resolves to nothing is refused with 404. See the outcome table in Section 3.2.3.
X-ISSUER-ID	Required on every call. The issuer's GIP code as registered with Sentinal (Section 3.2.3).

> **Appointment and revocation**
> Appointment and revocation are administered by Sentinal, not through the partner API. An issuer that revokes an agent takes effect on the agent's next call; there is no partner-side endpoint to allocate or deallocate. Raise appointment changes through your Sentinal technical contact (Appendix B).

3. Authentication & Authorization

3.1 Authentication Model

Sentinal uses a two-layer authentication model:

- Access token request — authenticated with an RSA-SHA256 signature produced with the Payment Processor's private key and verified against the Payment Processor's registered public key. This layer proves the identity of the Payment Processor and issues a short-lived bearer token.
- Authenticated requests — every subsequent request carries the bearer access token and an HMAC-SHA256 signature computed with the Payment Processor secret over a canonical string that binds the method, path, token, request body, and timestamp.

Both layers include an ISO 8601 timestamp that must be within five minutes of server time, preventing replay of captured requests.

3.2 Required Headers

3.2.1 Access Token Endpoint (POST /v1/psp/token)

Header	Required	Description
Content-Type	Yes	application/json
X-PARTNER-ID	Yes	Payment Processor API key
X-TIMESTAMP	Yes	ISO 8601 timestamp (validated within 5 minutes)
X-SIGNATURE	Yes	RSA-SHA256 signature — see 3.3

3.2.2 Authenticated Requests (Registration, Transactional, VAT Rules)

Header	Required	Description
Authorization	Yes	Bearer {access token} from the token endpoint
X-PARTNER-ID	Yes	Payment Processor API key
X-TIMESTAMP	Yes	ISO 8601 timestamp (validated within 5 minutes)
X-SIGNATURE	Yes	HMAC-SHA256 request signature — see 3.4
X-MERCHANT-KEY	Cond.	Merchant API key. Required for verifyTransaction PURCHASE, verifyTransactionStatus, declineTransaction, chargeback, and refund; optional otherwise.
X-ISSUER-ID	Cond.	Target issuer identifier — the issuer's GIP code. Required for National Collection Agents and Delegated Processor Agents (Section 2.4); not sent by an ordinary Payment Processor. See 3.2.3.
X-RULES-SIGNATURE	No	Active snapshot signature, echoed for audit linkage on verifyTransaction (advisory).
Content-Type	Yes	application/json

A missing header and an invalid credential fail differently

An ABSENT required header is a schema failure, not an authentication failure. Most endpoints declare their required headers in their request schema, which is evaluated before any credential is examined, so omitting

Authorization, X-PARTNER-ID, X-TIMESTAMP or X-SIGNATURE on those endpoints returns HTTP 400 with a validation message rather than 401.
Not every endpoint declares that schema. POST /v1/client/registerClient does not, so the same omission there reaches the authentication layer and returns 401. Do not rely on one code or the other to distinguish 'header missing' from 'credential rejected' — the status depends on the endpoint, not on the nature of the fault.
A header that is PRESENT but carries a bad value is always 401: an expired or malformed bearer token, an X-PARTNER-ID that is not yours, or a signature that fails verification. The one exception is X-TIMESTAMP outside the five-minute window, which is 400 on authenticated requests (Section 3.6) and 401 on the token endpoint (Section 4).
In short: treat 400 and 401 together as 'this request was not accepted for authentication reasons' and read the message, rather than branching on the code.

3.2.3 X-ISSUER-ID — Agent Delegation Header

A settlement agent authenticates as itself and names the issuer it is acting for in the X-ISSUER-ID header. Sentinal resolves the header to that issuer and attributes the call to it, so the transaction, the merchant and the VAT position all belong to the issuer rather than to the agent.

Property	Detail
Sent by	National Collection Agents and Delegated Processor Agents only (Section 2.4).
Value — primary	The target issuer's GIP code, as registered with Sentinal when the issuer was onboarded. This is the value to send.
Value — fallback	The target issuer's bank code, where no GIP code was registered for that issuer.
Format	1–32 characters, letters, digits, hyphen and underscore only (regex ^[A-Za-z0-9_-]{1,32}$). A value outside this shape is refused with 404, not 400.
Ordinary processors	Should not be sent. A direct Payment Processor may send its own registered identifier and the call proceeds; any other value is refused with 403.
Signature	Bound into the HMAC string to sign whenever present — see Section 3.4.

Resolution status

Sentinal resolves this header against a SINGLE registered identifier held for each issuer, not against a list of alternatives, so there is one correct value per issuer rather than a fallback chain. For this deployment that identifier is the issuer's GIP code, which is what is captured during issuer onboarding — send the GIP code. Where an issuer was onboarded before a GIP code was held, the bank code was registered instead and remains the value that resolves for it.
The practical consequence is that sending the wrong one of the two is indistinguishable from naming an issuer that does not exist: both return 404. Confirm the exact identifier held for each issuer with your Sentinal technical contact before go-live, and treat a 404 on a known issuer as a registration question rather than a code defect.

The header is authorisation-bearing

X-ISSUER-ID determines which issuer a call is attributed to, so it is verified before any merchant is resolved and before the request body is acted upon. A Delegated Processor Agent naming an issuer that has not appointed it is refused before the merchant lookup takes place. Contrary to earlier revisions the outcomes ARE distinguishable: an issuer that cannot be resolved returns 404, while an issuer that resolves but has not appointed you returns 403.

Each delegation failure carries a machine-readable code in the response content object:

Condition	HTTP	code	Applies to
Header absent on an agent principal	400	MISSING_ISSUER_ID	Both agent types.
Value outside the permitted shape	404	UNKNOWN_ISSUER	Both agent types.
No issuer holds that identifier	404	UNKNOWN_ISSUER	Both agent types.
More than one active issuer holds it	409	AMBIGUOUS_ISSUER	A configuration fault — raise it with Sentinal.
Issuer resolved, but has not appointed you	403	PROCESSOR_NOT_AUTHORIZED	Delegated Processor Agent only.
Endpoint outside your permitted set	403	PRINCIPAL_ROUTE_FORBIDDEN	Both agent types (Sections 2.4.1, 2.4.2).
Direct processor sending another party's identifier	403	ISSUER_MISMATCH	Ordinary Payment Processors.

3.3 RSA Signature — Access Token

The token request is signed with the Payment Processor's RSA private key (SHA256withRSA, per RFC 8017) and checked against the Payment Processor's registered public key. The string to sign is the partner ID and timestamp joined by a pipe character, and the resulting signature is submitted as LOWERCASE HEXADECIMAL — not base64:

```
stringToSign = X-PARTNER-ID + "|" + X-TIMESTAMP
signature    = lowerHex( RSA_SHA256_sign( pspPrivateKey, utf8Bytes(stringToSign) ) )
// place `signature` in the X-SIGNATURE header
// 512 lowercase hex characters for a 2048-bit key
```

Example string to sign (illustrative values):

```
PARTNER-abc123|2026-07-09T12:00:00.000Z
```

The token signature is HEX, not base64 — correction to versions 1.5.6 and 1.5.7

Both previous revisions specified base64 encoding for the access-token signature. That is incorrect, and it is the cause of token requests failing verification despite a correct key pair, a correct string to sign and a correct clock. The platform decodes X-SIGNATURE as hexadecimal. A base64 signature is not decoded, not verified, and the request is rejected with HTTP 401 and the generic message "Token generation failed!" — which gives no hint of an encoding problem.

For a 2048-bit key the correct header value is exactly 512 lowercase hexadecimal characters. If your value is 344 characters and ends in "=", you are sending base64.

Java: org.apache.commons.codec.binary.Hex.encodeHexString(sig). .NET: Convert.ToHexString(sig).ToLowerInvariant(). Python: sig.hex(). Node: sign.sign(privateKey, "hex"). PHP: bin2hex($sig).

This applies ONLY to the RSA token signature. The HMAC signature in Section 3.4 was already correctly documented as lowercase hex and is unchanged.

Prerequisite

Register your RSA public key with Sentinal before requesting tokens. A 2048-bit key pair is recommended. Keep the private key confidential; only the public key is shared.

3.4 HMAC Signature — Authenticated Requests

Authenticated requests are signed with the Payment Processor secret using HMAC-SHA256. The request body is serialised to canonical (compact) JSON, hashed with SHA-256, and the lowercase hex digest is embedded in the string to sign together with the HTTP method, request path, access token, and timestamp:

```
bodyHash     = lowerHex( SHA256( canonicalJson(requestBody) ) )
stringToSign = METHOD + ":" + requestPath + ":" + accessToken
               + ":" + bodyHash + ":" + X-TIMESTAMP
signature    = lowerHex( HMAC_SHA256( pspSecret, utf8Bytes(stringToSign) ) )
// place `signature` (64 hex chars) in the X-SIGNATURE header
```

METHOD is the upper-case HTTP verb. requestPath is the request path EXACTLY as transmitted, beginning with a leading slash and including the /v1 prefix and any query string — the server signs the raw request target, so a path differing by so much as a trailing slash produces a mismatch. accessToken is the bearer token without the "Bearer " prefix. The signature is 64 lowercase hex characters; any other length or character set is rejected before values are compared.

Agents: X-ISSUER-ID is appended to the string to sign

When — and only when — X-ISSUER-ID is present on the request, its trimmed value is appended as a final segment:

stringToSign = METHOD + ":" + requestPath + ":" + accessToken + ":" + bodyHash + ":" + X-TIMESTAMP + ":" + X-ISSUER-ID

Callers that do not send the header sign exactly the string described above, so this is not a breaking change for direct Payment Processors. Because the server appends the segment if and only if the client sent the header, adding, altering or stripping X-ISSUER-ID in transit all produce a signature mismatch. Agents sign with their OWN secret, not the target issuer's.

Golden rule

Compute bodyHash over exactly the same bytes you transmit as the request body. The server re-serialises the received JSON to canonical form before hashing (Section 3.5); your signature matches only if your canonical JSON is byte-identical to the server's.

Worked Example

Given the following inputs (illustrative secret and token):

```
pspSecret    = psp_secret_9f8e7d6c
method       = POST
requestPath  = /v1/transaction/creditInformation
accessToken  = eyJhbGciOiJIUzI1NiJ9.eyJwc3BJZCI6IjY0YSJ9.EXAMPLEsig
X-TIMESTAMP  = 2026-07-09T12:00:00.000Z
requestBody  = {"creditRef":"CR-2026-0007","creditAmount":1500.5,
                "creditTime":"2026-07-09T11:59:00+00:00",
                "settlementReportDate":"2026-07-09"}
```

Step 1 — SHA-256 of the canonical body:

```
bodyHash = d866521433e8f59398b914206370dabe6cc6c4704822826fd06c58eced599398
```

Step 2 — assemble the string to sign:

```
POST:/v1/transaction/creditInformation:eyJhbGciOiJIUzI1NiJ9.eyJwc3BJZCI6
IjY0YSJ9.EXAMPLEsig:d866521433e8f59398b914206370dabe6cc6c4704822826fd06c58
```

```
eced599398:2026-07-09T12:00:00.000Z
```

Step 3 — HMAC-SHA256 with the Payment Processor secret (this value goes in X-SIGNATURE):

```
X-SIGNATURE = 15c33f588324933a8cbd3c4ab9783daeb795de81f526ad52e9d4fb0697da2761
```

3.5 Canonical JSON & Cross-Language Signing

The server computes bodyHash over the request body re-serialised as canonical, compact JSON — equivalent to JavaScript's JSON.stringify applied to the parsed body. Your serialisation must observe the following rules:

- No insignificant whitespace — no spaces after colons or commas, no newlines or indentation.
- UTF-8 output — encode the string as UTF-8 before hashing.
- Do not escape non-ASCII characters — emit raw UTF-8, not \uXXXX escapes.
- Do not escape forward slashes ("/"), and do not HTML-escape the characters < > &.
- Preserve the field order you transmit — hash exactly what you send.
- Numbers use JavaScript formatting — trailing zeros after the decimal point are removed, and integers carry no decimal point (see below).

Canonical JSON Example

A body may be authored with any formatting; it is canonicalised before hashing. Note that creditAmount 1500.50 becomes 1500.5.

Authored (pretty):

```
{
  "creditRef": "CR-2026-0007",
  "creditAmount": 1500.50,
  "creditTime": "2026-07-09T11:59:00+00:00",
  "settlementReportDate": "2026-07-09"
}
```

Canonical (what is hashed):

```
{"creditRef":"CR-2026-0007","creditAmount":1500.5,"creditTime":"2026-07-
09T11:59:00+00:00","settlementReportDate":"2026-07-09"}
```

Decimal Handling

Sentinal serialises numbers using JavaScript number formatting, which strips trailing zeros after the decimal point and uses a single unified representation for integers and decimals. Clients in languages that preserve trailing zeros (for example Java, where a double or BigDecimal may serialise 15.0 or 1.50) must normalise numbers to the same form before hashing, or the signature will not match.

Input value	Canonical form (hashed)	Note
15.0	15	Trailing zero removed; no decimal point.
2.50	2.5	Single trailing zero removed.
0.150	0.15	Trailing zero removed.
100.00	100	Becomes an integer form.
1500.50	1500.5	As used in the worked example above.

Reference behaviour

The reference integration harness derives its canonical JSON from the server's own serialised values (it re-serialises the parsed JSON tree rather than re-formatting native language numbers). This guarantees the client's number formatting matches the server's — the recommended approach for any non-JavaScript client.

Troubleshooting tip

If a signature fails only for some payloads — typically those containing accented characters, forward slashes, or amounts with trailing zeros — the cause is almost always a serialisation difference from the rules above rather than an incorrect secret.

3.6 Timestamp & Replay Protection

X-TIMESTAMP must be a valid ISO 8601 timestamp within five minutes (300 seconds) of Sentinal's server time. Requests outside this window are rejected with HTTP 400. Use a synchronised clock (NTP) and generate the timestamp immediately before signing.

3.7 Credential & Key Management

- API key (X-PARTNER-ID) — identifies the Payment Processor; sent on every request.
- Payment Processor secret — used for HMAC request signing; never transmitted. Store it securely.
- RSA key pair — the private key signs token requests; the public key is registered with Sentinal. Rotate keys periodically and re-register the public key on rotation.
- Merchant key / secret — returned from merchant registration (Section 5.1); used as X-MERCHANT-KEY and, where applicable, for merchant-scoped signing.

4. Access Token

POST /v1/psp/token

Authenticates a Payment Processor using the RSA signature scheme (Section 3.3) and returns a short-lived JWT access token for use as the Authorization bearer on all subsequent requests.

Request Body

Field	Type	Required	Description
grantType	string	Yes	Must be the literal value "client_credentials".

Authentication is carried in the headers (Section 3.2.1). No API key or secret is placed in the body.

Example Request

```
POST /v1/psp/token
Content-Type: application/json
X-PARTNER-ID: PARTNER-abc123
X-TIMESTAMP: 2026-07-09T12:00:00.000Z
X-SIGNATURE: <base64 RSA-SHA256 of "PARTNER-abc123|2026-07-09T12:00:00.000Z">

{ "grantType": "client_credentials" }
```

Response — 200 OK

Field	Type	Description
success	boolean	true
message	string	"Access token generated successfully."
content.accessToken	string	JWT bearer token.
content.tokenType	string	"Bearer"
content.expiresIn	number	Token lifetime in seconds (3600 = 1 hour).

Error Responses

HTTP	Condition	Meaning & Resolution
401	Invalid or expired timestamp	X-TIMESTAMP missing, malformed, or outside the 5-minute window. NOTE: this returns 401, not 400 — the check runs inside the handler and every handler failure is reported as 401.
400	Unsupported grant type	grantType is not "client_credentials".
401	Invalid Payment Processor credentials	X-PARTNER-ID does not match a known, active Payment Processor.
401	Invalid RSA signature	X-SIGNATURE fails verification. Confirm it is LOWERCASE HEX (Section 3.3) before suspecting the key — a base64 signature fails here.
401	Payment Processor public key not configured	No public key on file; register one before requesting tokens.

Every token failure returns the same message

The token endpoint returns one generic body — {"success": false, "message": "Token generation failed!", "content": null} — for an unknown partner ID, an unregistered public key, a bad signature, a skewed timestamp and an archived processor alike. The conditions above are distinguished in Sentinal server logs, not in the response. Work through the causes in order rather than reading the message; signature encoding (Section 3.3) is by far the most common.

5. Register Services

5.1 POST /v1/client/registerClient

Registers a merchant under the authenticated Payment Processor (HMAC authentication). If a matching merchant already exists, existing credentials are returned. At least one of classification or mccCode must be supplied so that VAT treatment can be resolved.

Request Body

Field	Type	Req.	Rules & Description
registeredName	string	No	Legal registered name (max 100).
registrationNumber	string	No*	Business registration number (max 30). *Strongly recommended — the VAT Rules Engine keys merchant-specific overrides on the hashed registration number. Card processors: populate this with DE 42 (see the note below).
countryCode	string	Yes	ISO 3166-1 alpha-2, uppercase. Must equal the configured country (GH).
currency	string	Yes	ISO 4217, uppercase (e.g. GHS).
email	string	No	Valid email (max 30).
phoneNumber	string	No	Validated and normalised (max 20).
classification	string	Cond.	Merchant Type (Section 9). One of classification / mccCode is required.
mccCode	string	Cond.	4-digit ISO 18245 MCC. One of classification / mccCode is required.
organizationTin	string	No	Ghana Tax Identification Number (VAT registration). Also used by the VAT Rules Engine — its hash is an alternative merchant match key.
merchantName	string	Yes	Trading name (1–100). Required.
merchantCity	string	No	City of operation (max 30).
merchantCountryCode	string	Yes	ISO 3166-1 alpha-2, uppercase.
acquirerInstitutionId	string	No	Acquirer institution identifier (max 16).
merchantId	string	No	Acquirer-side merchant identifier (max 32).
merchantGroup	string	No	Merchant group / chain identifier.

Example Request

```
POST /v1/client/registerClient
Authorization: Bearer <accessToken>
X-PARTNER-ID: PARTNER-abc123
X-TIMESTAMP: 2026-07-09T12:00:00.000Z
X-SIGNATURE: <HMAC-SHA256 per Section 3.4>
Content-Type: application/json

{
  "registeredName": "Globex Cloud Services Inc",
  "registrationNumber": "GBX-8841920",
  "countryCode": "GH",
  "currency": "GHS",
  "mccCode": "5734",
  "merchantName": "Globex Cloud",
  "merchantCountryCode": "US"
```

```
}
```

The example registers a cross-border digital-service merchant (SaaS, MCC 5734) that sells to Ghana buyers: countryCode is GH (the deployment/tax country) and registrationNumber is supplied for VAT-rules matching.

Response — 200 OK (content)

Field	Type	Description
apiKey	string	Merchant API key for the X-MERCHANT-KEY header.
secret	string	Merchant secret (returned once).
vatTreatment.vatRate	number	Effective VAT rate as a PERCENTAGE, e.g. 20 means 20%. It is NOT a decimal fraction — divide by 100 before applying it. See the units warning in Section 7.1.
vatTreatment.defaultCollectionStatus	string	COLLECTED | NOT_COLLECTED | POTENTIALLY_COLLECTED.
vatTreatment.collectionRules	array	Payment-method-specific collection rules.
vatTreatment.vatExempt	boolean	Whether the merchant is VAT exempt.

Merchant matching in the VAT Rules Engine

The VAT Rules Engine matches a merchant to its VAT overrides (exemptions and status overrides) by the SHA-256 hash of EITHER the registrationNumber OR the organizationTin (VAT registration number). Either identifier resolves the merchant — you may use either, and supplying both is ideal.

Without at least one of these identifiers, the merchant cannot be matched to its overrides in the snapshot and will fall back to its MCC / Type default.

Strongly recommended: always send the Merchant Registration Number. It is the primary, most reliable match key for the VAT Rules Engine.

Card processors: use DE 42 as the registrationNumber

DE 42 (Card Acceptor Identification Code) from the scheme authorisation message is the only value carried on the card rails that corresponds to a merchant registration number. Sentinal's recommendation is therefore to populate registrationNumber with DE 42.

Send it in registrationNumber, not only in merchantId. merchantId is stored for reference and is NOT a match key — a merchant identified solely by merchantId can never be matched to its overrides in the snapshot, and will silently fall back to its MCC default. Populating both fields with DE 42 is acceptable and is the safest option.

Ghana merchants are registered VAT-EXCLUSIVE

Merchants in this deployment are set up on a VAT-exclusive basis. Submit the net amount on verifyTransaction; VAT is calculated as a surcharge on top of it and the cardholder is charged totalAmount + vatAmount (Section 6.1). This is a Sentinal-side registration setting — Payment Processors do not send a VAT-basis flag on registerClient.

VAT treatment resolution

Effective VAT treatment is resolved with the priority: merchant override → registered MCC fee → classification (Type) fee.
Where a merchant matches NEITHER an MCC fee NOR a classification fee, this response returns defaultCollectionStatus COLLECTED at the country default VAT rate. That differs deliberately from the rule you apply to the snapshot: rule 5 in Section 7.3 tells a Payment Processor NOT to collect when nothing matches, because a Payment Processor cannot see the country default and must fail safe. Sentinal can see it, so this response and verifyTransaction use it.
The consequence is that vatTreatment here is authoritative for the merchant and may name a rate you cannot derive from your own snapshot — persist what is returned rather than recomputing it. The snapshot's defaultVatRate field remains informational and must not be applied as a fallback rate by a Payment Processor.

5.2 Merchant Pre-Registration (Whitelisting)

Pre-registration is the submission of your merchant estate to Sentinal during onboarding, ahead of live traffic, so that each merchant carries a deterministic VAT treatment from its first transaction.

One category of merchant does not depend on it. Merchants that Sentinal has approved and marked VAT exempt are distributed to EVERY Payment Processor, in the national exempt list carried by every snapshot, whether or not you have ever registered or transacted with them. You can therefore identify an exempt merchant before its first transaction, from your own snapshot alone. Pre-registration aligns everything else.

5.2.1 What your snapshot already tells you, and what it does not

The national exempt list is delivered to everyone. merchantOverrides.exemptMerchants carries every merchant whose Sentinal record is APPROVED and which holds either a VAT-exemption flag or a NOT_COLLECTED collection status, and that list is written into EVERY Payment Processor snapshot regardless of who registered the merchant. An exempt merchant is therefore visible to you before you have transacted with it.

Property	Detail
Distributed to	Every Payment Processor, in every snapshot, without regard to which processor registered the merchant.
Inclusion test	The merchant record is APPROVED, and carries either a VAT-exemption flag or a NOT_COLLECTED collection status.
Identified by	SHA-256 of registrationNumber and / or organizationTin. A merchant holding neither identifier is OMITTED from the list, because no Payment Processor would be able to match it.
reason	VAT_EXEMPT, KNOWN_MERCHANT or MANUAL_OVERRIDE — the ground on which the exemption rests.
How to apply it	Rule 1 of Section 7.3. Hash the registration number or TIN you hold, test the exempt list FIRST, and stop on a match.

Check the exempt list before you quote VAT, not after
Because the list is national, the largest source of disagreement between your figure and Sentinal's is removed at source: a merchant that GRA has exempted is already in your snapshot. Test rule 1 against every merchant before applying an MCC or Type rate, including merchants you consider new.
The list is only as fresh as your snapshot. An exemption granted mid-window reaches you through the merchant.vat_updated webhook (Section 8.4), through POST /v1/client/statusChanges (Section 5.3), or at your next snapshot refresh (Section 7.4) — whichever you consume first.

What the exempt list does not cover, and what pre-registration therefore still resolves:

- Status overrides remain Payment-Processor-scoped. merchantOverrides.statusOverrides carries merchants deviating from their own MCC or Type default, and appears only for the Payment Processors a merchant is linked to. A merchant held elsewhere whose deviation is an override rather than an exemption is not in your snapshot.
- Matching is central. A merchant is identified by the composite key merchantName + mccCode + classification + merchantCountryCode, and by name matching against the known-merchant list, including normalised and fuzzy name matching. You cannot reproduce that on your side.
- Stored attributes win. Where registerClient resolves to an existing record, the classification and MCC held by Sentinal may differ from the ones you submitted, and your values do NOT overwrite them. Your local rate is then derived from attributes Sentinal is not using.

Sequence	Outcome
Merchant is nationally exempt	Already present in tier 1 of your snapshot. No discrepancy arises, provided you tested rule 1 before applying a rate.
Merchant exists with different attributes	registerClient returns the stored record. Its classification or MCC may differ from yours, and your submitted values do not replace them.
Merchant carries a status override held elsewhere	Absent from your snapshot. Only the registration response, or verifyTransaction, reveals it.
Discrepancy	VAT quoted — and in some flows collected — against a rate Sentinal is not applying.

> **verifyTransaction is canonical**
> verifyTransaction returns the canonical VAT amount and collection instruction. Where a locally computed amount differs, the returned value governs and must be applied. Pre-registration exists so that the two do not differ at the point of collection.

5.2.2 Pre-registration procedure

Step	Action
1	Submit each merchant to POST /v1/client/registerClient (Section 5.1) at onboarding — not at first transaction.
2	Treat the response as authoritative. Persist the returned vatTreatment, classification, mccCode and collection status against the merchant, overwriting local assumptions.
3	Retrieve GET /v1/psp/vatRules (Section 7). The merchant is now linked to you, so its treatment is carried in the next snapshot.
4	Apply snapshot rules only to merchants confirmed by step 2. An unconfirmed merchant has no reliable local treatment.

Always supply registrationNumber and, where held, organizationTin. These are the hashed keys the engine uses to match a merchant to its overrides; without at least one, the merchant cannot be matched and falls back to its MCC or Type default (Section 5.1).

5.3 POST /v1/client/statusChanges

Returns every merchant registered under the authenticated Payment Processor whose record has changed since a given timestamp. HMAC authenticated with the Payment Processor secret; X-MERCHANT-KEY is NOT sent, because the call is scoped to the Payment Processor and returns the whole estate rather than one merchant. This endpoint was omitted from every previous revision of this document.

Request Body

Field	Type	Req.	Description
lastUpdatedAfter	string	Yes	ISO 8601 datetime. Merchants whose status history or record was updated strictly AFTER this instant are returned. Pass the timestamp of your previous successful poll.

Example Request

```
POST /v1/client/statusChanges
Authorization: Bearer <accessToken>
X-PARTNER-ID: PARTNER-abc123
X-TIMESTAMP: 2026-07-09T12:00:00.000Z
X-SIGNATURE: <HMAC-SHA256 per Section 3.4>
Content-Type: application/json

{ "lastUpdatedAfter": "2026-07-08T12:00:00.000Z" }
```

Response — 200 OK (content)

content is an ARRAY of merchant objects, ordered most-recently-updated first. An empty array means nothing changed in the window — it is not an error. Each element carries:

Field	Description
registeredName	Legal registered name held by Sentinal.
merchantName	Trading name held by Sentinal. May differ from what you submitted where the merchant matched the national known-merchant list.
status	Merchant record status, e.g. APPROVED, DECLINED.
collectionStatus	COLLECTED | NOT_COLLECTED | POTENTIALLY_COLLECTED — the merchant-level collection status. This is the field that changes when an exemption is granted.
classification	Sentinal Type (Section 9).
mccCode	Registered MCC.
currency	Registered currency.
countryCode	Deployment country code.
registrationNumber	Registration number as stored, in PLAIN TEXT — unlike the snapshot and the webhook, which carry SHA-256 hashes.
merchantCity	City of operation.
merchantCountryCode	Merchant country code.
acquirerInstitutionId	Acquirer institution identifier.
merchantId	Acquirer-side merchant identifier.
merchantGroup	Merchant group / chain identifier.
lastUpdated	ISO 8601 timestamp of the most recent change. Use the maximum value seen as the lastUpdatedAfter of your next poll.

Use this where the webhook is not an option

Section 7.4 warns that a Payment Processor which does not consume merchant.vat_updated will not see a mid-window exemption until it next re-fetches the snapshot — up to a month on the production cadence. This endpoint is the supported answer. Poll it on a schedule matching your tolerance for staleness (hourly is reasonable), pass the highest lastUpdated you have already seen, and re-fetch GET /v1/psp/vatRules when a returned merchant shows a collectionStatus you were not expecting.

It also serves as a reconciliation tool during the stabilisation period of Section 5.2.3, because it returns registrationNumber in plain text and so lets you align your merchant keys with Sentinal's without hashing anything.
Note the asymmetry: this endpoint returns identifiers in plain text because it is scoped to your own estate, whereas the snapshot and the webhook hash them because the exempt list is national.

6. Transactional Services

6.1 POST /v1/transaction/verifyTransaction

Verifies and records an already-authorised transaction and returns the applicable VAT treatment. This endpoint sits outside the authorisation flow — it is called after authorisation completes and never affects the authorisation outcome or settlement. HMAC authenticated. X-MERCHANT-KEY is required for PURCHASE transactions; for non-PURCHASE transactions without a merchant key, participant identifiers are required (see business rules).

Amounts are exclusive of VAT (Ex-VAT) — submit within 60 seconds

Merchants send the transaction amount EXCLUSIVE of VAT. The integrating party pushes that Ex-VAT amount as totalAmount; Sentinal calculates VAT on it (per Section 7) and returns vatAmount. Example: a GHS 100.00 Ex-VAT service is submitted as totalAmount 100.00; at a 20% rate Sentinal returns vatAmount 20.00, and the amount charged to the cardholder is 120.00.

This call takes place AFTER authorisation has completed and forms no part of the authorisation decision. Once authorised, the integrating party has 60 seconds to submit the Ex-VAT transaction to Sentinal for validation of the full amount. Sentinal never gates, delays, or reverses the authorisation.

Sentinal does not return a VAT-inclusive total — compute it as totalAmount + vatAmount

The totalAmount in the response is the amount you submitted, echoed back (converted to GHS if you submitted in a foreign currency). Sentinal never adds VAT to it. There is no VAT-inclusive total field in the response.

Whether that figure is inclusive or exclusive of VAT is therefore decided entirely by what you send, which follows the merchant's VAT basis. Ghana merchants are registered VAT-EXCLUSIVE (Section 5.1), so for this deployment: you submit the net amount, vatAmount is calculated as a surcharge on top of it (totalAmount x vatRate / 100, since the published rate is a percentage — see Section 7.1), merchantAmount equals totalAmount, and the gross amount charged to the cardholder is totalAmount + vatAmount — which the Payment Processor computes.

Do not assume the response echoes a gross figure. A deployment whose merchants are registered VAT-inclusive behaves differently: there VAT is extracted from the submitted amount rather than added to it, and merchantAmount is the net. The arithmetic below applies to Ghana.

Field	Ghana (VAT-exclusive merchants)
totalAmount (request)	The net, Ex-VAT transaction amount as presented at the acceptance device.
vatAmount (response)	totalAmount x vatRate / 100 — VAT charged as a surcharge on top of the submitted amount. The division is required because the published rate is a percentage (Section 7.1).
totalAmount (response)	Unchanged echo of the submitted amount (GHS-converted where applicable). NOT VAT-inclusive.
merchantAmount (response)	Equals totalAmount for a VAT-exclusive merchant — the merchant receives the full net amount.
Cardholder is charged	totalAmount + vatAmount. Computed by the Payment Processor; Sentinal does not return this value.

Request Body

Field	Type	Req.	Rules & Description
ref	string	Yes	Transaction reference (max 130).
totalAmount	number	Yes	Positive; up to 10 decimal places (rounded to 2).

Field	Type	Req.	Rules & Description
foreignAmount	number	No	Foreign-currency amount. Where currency differs from GHS and this field is supplied it must be strictly greater than 0 — it is the divisor that derives the exchange rate. Omit it entirely to have Sentinal apply its own reference rate instead.
currency	string	Yes	ISO 4217, uppercase.
countryCode	string	Yes	ISO 3166-1 alpha-2, uppercase.
paymentMethods	object	Yes	Map of method → positive amount. Methods: CREDIT_CARD, DEBIT_CARD, WALLET, SAVINGS, BANK_TRANSFER, PAYLATER. At least one entry, and no other keys are accepted. The values MUST sum exactly to totalAmount — see the business rules below.
transactionFee	number	No	OMIT THIS FIELD for Ghana — fee reporting is not enabled. Sending it at all, including as 0, is rejected. See the note below.
mccCode	string	No	4-digit MCC (nullable).
issuingBank	string	Yes	Issuing bank code. Maximum 8 characters. Free text — send whatever institution identifier you hold (e.g. FBGL for Fidelity Bank Ghana Limited); Sentinal does not validate it against a register.
issuerCountryCode	string	Yes	ISO 3166-1 alpha-2; must equal the configured country (GH).
cardType	string	No	VISA | MASTERCARD | JCB | CUP | AMEX | GPN (nullable).
transactionTime	string	Yes	ISO 8601; must not be later than X-TIMESTAMP.
retrievalRefNumber	string	Yes	ISO 8583 field 37 (1–32).
destinationType	string	Yes	OVERBOOKING | BANK_TRANSFER | PAYMENT | PURCHASE | REMITTANCE | WIRE_TRANSFER.
originParticipantID	string	Cond.	Required for non-PURCHASE without a merchant key.
destinationParticipantID	string	Cond.	Required for non-PURCHASE without a merchant key.
networkHashedDestination	string	Cond.	64-char SHA-256 hash, optional NETv{n}: prefix. Required for non-PURCHASE without a merchant key.
postSettlement	boolean	No	Marks the transaction as potentially collected (default false).
remittanceSurcharge	boolean	Cond.	Required for REMITTANCE transactions.
items	array	No	Optional line-item details; stored as-is, no effect on VAT.

Business Rules

- PURCHASE transactions require the X-MERCHANT-KEY header.
- Non-PURCHASE transactions without a merchant key require originParticipantID, destinationParticipantID and networkHashedDestination.
- REMITTANCE transactions require remittanceSurcharge.
- issuerCountryCode must equal the configured country (GH); otherwise HTTP 400.
- transactionFee must be OMITTED for Ghana. The check is on the field's presence, not its value, so sending transactionFee: 0 is rejected exactly as a non-zero value would be.
- transactionTime must not be later than X-TIMESTAMP, and must not be in the future relative to Sentinal server time.
- The values in paymentMethods must sum EXACTLY to totalAmount, compared to the cent. A split-tender transaction of 250.75 must be submitted as methods totalling 250.75, not as a partial allocation. A mismatch is rejected with HTTP 500 and a message naming both figures.
- The transaction must be submitted on its own calendar day, evaluated in Ghana local time (GMT). A transaction timestamped 2026-07-09 and submitted at any time on 2026-07-10 is rejected. This is the

enforced submission window; the 60-second target is the operational expectation, not the limit the platform applies.

- currency must be GHS, or a foreign currency for which Sentinal holds a reference rate. Where no rate is held and no foreignAmount is supplied, the request is rejected with HTTP 400 naming the currency.
- Re-submitting a ref this merchant has already used returns the ORIGINAL transaction unchanged rather than creating a second one. See the idempotency note below.

Example Request

Scenario: a Ghana cardholder purchases a digitally-delivered service (SaaS) from a cross-border merchant, paying GHS 250.75 with a Ghana-issued VISA card. The issuing bank is in Ghana (issuerCountryCode = GH), countryCode = GH, and currency = GHS.

```
POST /v1/transaction/verifyTransaction
Authorization: Bearer <accessToken>
X-PARTNER-ID: PARTNER-abc123
X-MERCHANT-KEY: <merchant apiKey>
X-TIMESTAMP: 2026-07-09T12:00:00.000Z
X-SIGNATURE: <HMAC-SHA256 per Section 3.4>

{
  "ref": "TXN-2026-0001",
  "totalAmount": 250.75,
  "currency": "GHS",
  "countryCode": "GH",
  "paymentMethods": { "CREDIT_CARD": 250.75 },
  "mccCode": "5734",
  "issuingBank": "GCB",
  "issuerCountryCode": "GH",
  "cardType": "VISA",
  "transactionTime": "2026-07-09T11:58:00+00:00",
  "retrievalRefNumber": "123456789012",
  "destinationType": "PURCHASE"
}
```

eWallet / mobile-money instruments

For a Ghana eWallet or mobile-money account, use the WALLET payment method (e.g. paymentMethods: { "WALLET": 250.75 }). issuerCountryCode remains GH; countryCode and currency remain GH / GHS. The remaining fields are unchanged.

verifyTransaction is idempotent on ref — but wait at least one second before repeating a call

A repeated ref for the same merchant under the same Payment Processor does not create a second transaction and does not recalculate VAT. Once the first submission has settled, Sentinal returns the stored transaction exactly as first recorded, with an additional message field reading "Transaction already approved previously. No changes made." and, where the original was not itself flagged, an auditReason of "Duplicate transaction reference - already approved."

WAIT AT LEAST ONE SECOND before resubmitting a ref. The transaction is written asynchronously, and for a brief window after the first submission the repeat is recognised as a duplicate before the stored record can be read back. A resubmission inside that window returns HTTP 500 with the generic message "Failed to verify transaction!" instead of the original transaction. No second transaction is created and no VAT is double-counted; the only consequence is that you do not get the record back. The backoff schedule in Section 11.4 already satisfies this, because its second attempt waits approximately one second.

If you do receive a 500 on a resubmitted ref, treat it as retryable rather than as a failed transaction. Back off and resend the IDENTICAL request; it will return the original record once the write has settled. Do NOT allocate a new ref, which is the one response that would create a genuine duplicate.
The guarantee is scoped to the merchant. The same ref submitted by a DIFFERENT Payment Processor for a cross-border merchant is a separate transaction and may be flagged as a cross-processor duplicate.

A merchant registered with merchantCountryCode GH is exempted by Sentinal

Where the country rule localMerchantVatExempt is in force, Sentinal sets vatAmount to 0 and returns a non-collecting status for any merchant whose registered merchantCountryCode equals the deployment country. This is applied server-side on the MERCHANT record — not on the countryCode or issuerCountryCode you send, both of which are GH for every in-scope Ghana transaction.
This is the mechanism behind the aggregator exception in Section 7.3. An upstream provider submitting against an aggregator's Ghanaian registration receives a zero VAT amount and a non-collecting instruction, which is the correct outcome: the VAT obligation sits with the aggregator and must not be collected twice.

Response — 200 OK (content)

Field	Type	Description
transactionId	string	Internal transaction identifier.
totalAmount	number	The submitted amount, echoed back (GHS-converted where the request was in a foreign currency). NOT VAT-inclusive — see the note above.
status	string	COLLECTED | NOT_COLLECTED | POTENTIALLY_COLLECTED | DECLINED. DECLINED is returned when the MERCHANT record has been declined on the Sentinal platform — it is a merchant-state value, not a statement about your authorisation, which Sentinal never sees. Treat it as 'do not collect' and raise the merchant with Sentinal.
vatAmount	number	VAT calculated on the submitted amount. For Ghana's VAT-exclusive merchants this is a surcharge: totalAmount x vatRate / 100.
merchantAmount	number	The amount due to the merchant. Equals totalAmount for a VAT-exclusive merchant; totalAmount less vatAmount for a VAT-inclusive one.
foreignAmount	number | null	Foreign-currency amount (null for domestic).
foreignVatAmount	number | null	Foreign-currency VAT (null for domestic).
exchangeRateUsed	number | null	Applied exchange rate; null for a domestic transaction. Note the contrast with the chargeback response (Section 6.5), where the same field is OMITTED rather than null for a GHS request.
auditFlagged	boolean	Whether the transaction was flagged for audit.
auditReason	string | null	Reason, when flagged.
penaltyApplied	boolean	Not in use for Ghana — see the note below.
penaltyAmount	number	Not in use for Ghana — see the note below.
remittanceSurcharge	boolean	Present only for REMITTANCE transactions.
message	string	Present ONLY when the submission duplicated a ref already recorded for this merchant. Its presence is the idempotency signal — see the note above.

Late-submission penalties are not in use for Ghana

penaltyApplied and penaltyAmount are returned by the platform but are NOT part of the Ghana deployment at this stage. No penalty regime has been adopted here. Read them if you wish, but do not build logic on them and do not surface them to merchants; they are response fields only and are never sent on a request.

Foreign-currency (MCA) transactions

For multi-currency acquiring (MCA) transactions — where the transaction currency differs from the local currency — supply foreignAmount alongside totalAmount. Sentinal determines VAT using its reference exchange rate and returns foreignAmount, foreignVatAmount, and exchangeRateUsed. The reference rates themselves are retrievable in advance from POST /v1/rates (Section 6.8).

Three cases occur: (1) non-MCA — transaction, account, and verification are all in the local currency; (2) MCA — verification is performed in the foreign currency; (3) MCA with conversion — the issuer converts to the local currency using Sentinal's reference rate, and the local-currency amounts are submitted. The response reflects the currency in which VAT was calculated.

6.2 POST /v1/transaction/verifyTransactionStatus

Returns the current status of a previously verified transaction, looked up by transactionId or ref. HMAC authenticated; X-MERCHANT-KEY is required. Use this after a client-side timeout to determine the outcome before retrying a verifyTransaction call (see the resilient-retry workflow in Section 11.4).

Request Body

Provide at least one identifier; if both are given, transactionId takes precedence.

Field	Type	Req.	Description
transactionId	string	Cond.	Internal transaction identifier returned by verifyTransaction.
ref	string	Cond.	The transaction reference you submitted. One of transactionId / ref is required.

Example Request

```
POST /v1/transaction/verifyTransactionStatus
Authorization: Bearer <accessToken>
X-PARTNER-ID: PARTNER-abc123
X-MERCHANT-KEY: <merchant apiKey>
X-TIMESTAMP: 2026-07-09T12:00:00.000Z
X-SIGNATURE: <HMAC-SHA256 per Section 3.4>

{ "ref": "TXN-2026-0001" }
```

Response — 200 OK (content)

The response returns the FULL stored transaction record, not the subset returned by verifyTransaction. It carries every field Sentinal holds for the transaction — including status, vatAmount, merchantAmount, totalAmount, processingFee, transactionFee, transactionVAT, paymentMethods, currency, submittedMccCode, effectiveMccCode, mccOverridden, issuingBank, retrievalRefNumber, transactionTime, destinationType, auditFlagged, auditReason, the refund totals (totalRefunded, totalPendingRefund) and createdAt / updatedAt. The merchant, country and Payment Processor references are removed before the record is returned. A 400 is returned if neither identifier is supplied.

The identifier field is _id here, not transactionId

verifyTransaction returns the transaction identifier as transactionId. verifyTransactionStatus returns the same value as _id, because it returns the stored record rather than a constructed response. A client that reads transactionId from this response reads undefined. Map _id here to the transactionId you stored from verifyTransaction.

Lookup precedence: where both are supplied, transactionId is tried first and ref is used only if it finds nothing. A ref lookup is scoped to the merchant identified by X-MERCHANT-KEY; a transactionId lookup is checked against that merchant afterwards and refused if it belongs to another.

KNOWN DEVIATION: a transaction that cannot be found, and one belonging to a different merchant, currently return HTTP 500 rather than the 404 and 403 that Section 11.2 would imply. Do not treat a 500 here as proof of a platform fault — verify the identifier first. This is scheduled for correction; the response body and success path are unaffected.

6.3 POST /v1/transaction/creditInformation

Successful Credit Information. The Payment Processor notifies Sentinal that VAT for a settlement has been credited. HMAC authenticated. This operation is used in the worked signing example in Section 3.4.

Field	Type	Req.	Description
creditRef	string	Yes	Credit reference (max 100).
creditAmount	number	Yes	Credited amount (positive).
creditTime	string	Yes	ISO 8601 with a NUMERIC timezone offset, e.g. 2026-07-09T11:59:00+00:00. A "Z" suffix is REJECTED — the field is pattern-matched and requires an explicit +HH:MM or -HH:MM. Must not be more than five minutes in the future.
settlementReportDate	string	Yes	Settlement report date in yyyy-mm-dd format (e.g. 2026-07-09).

Response — 200 OK (content)

Field	Type	Description
creditId	string	Identifier of the recorded credit.
settlementAmount	number	The settlement amount Sentinal holds for the reported settlement date.
varianceAmount	number	creditAmount less settlementAmount. A non-zero value is the reconciliation difference and is the figure to investigate.
reconciliationStatus	string	Outcome of matching your credit against Sentinal's settlement record for that date.

Error Responses

HTTP	code	Meaning & Resolution
400	INVALID_CREDIT_TIME	creditTime is malformed, lacks a numeric offset, or is more than five minutes in the future.
400	INVALID_AMOUNT	creditAmount is not a positive finite number.
409	DUPLICATE_REF	This creditRef has already been recorded. Credits are idempotent on creditRef — do not retry with a new ref for the same settlement.
409	PSP_DUPLICATE_BLOCKED	A credit for this settlement has already been recorded for this Payment Processor.

HTTP	code	Meaning & Resolution
422	SETTLEMENT_DATA_NOT_FOUND	The request is well formed but Sentinal holds no settlement data for settlementReportDate. Confirm the date, then retry once the settlement has been produced.
500	UNKNOWN_ERROR	Internal error; retry with backoff.

National Collection Agents: this is the only endpoint available to you

A National Collection Agent — the principal type under which GhIPSS operates — may call creditInformation and nothing else; every other endpoint in Sections 5 and 6 returns 403. X-ISSUER-ID is mandatory on every call and carries the target issuer's GIP code (Section 3.2.3), and its value is appended to the HMAC string to sign (Section 3.4). The agent signs with its OWN Payment Processor secret, never the issuer's.

6.4 POST /v1/transaction/declineTransaction

Records a declined transaction and creates a linked PENDING refund. HMAC authenticated. Declines are permitted only on the transaction day (D+0); for later reversals use the chargeback endpoint. The external ref field is optional.

Field	Type	Req.	Description
ref	string	No	External reference (max 256). If omitted, Sentinal generates one in the form DECLINE-{epochMillis}-{transactionRef} and returns it. Either way the RETURNED value is what must be echoed as declineChargebackRef on the refund call — do not reconstruct it.
transactionRef	string	Yes	Reference of the transaction being declined (max 130).
currency	string	Yes	ISO 4217.
refundAmount	number	Yes	Amount to reverse (≥ 0).
refundVatAmount	number	Yes	VAT portion to reverse (≥ 0).
reason	string	Yes	Decline reason. Must be the literal value CANCELLATION.
verifiedTransactionAmount	number	Yes	Must match the recorded transaction amount.
verifiedRetrievalRefNumber	string	Yes	Must match the recorded RRN.
verifiedTransactionTimestamp	string	Yes	ISO 8601; must match the recorded time.
evidence	array	No	0–3 evidence files, each { filename (max 255), contentType, data }, where data is base64 and contentType is one of application/pdf, image/jpeg, image/jpg, image/png, text/plain. Files are submitted as base64 inside the JSON body — multipart/form-data is not accepted.

Preconditions common to declineTransaction and chargeback

ONLY A COLLECTED TRANSACTION MAY BE REVERSED. A transaction whose status is NOT_COLLECTED, POTENTIALLY_COLLECTED or DECLINED is refused with HTTP 400 naming the status found. There is no VAT position to reverse where none was collected, so this is not an error to work around — if a reversal is genuinely required against such a transaction, raise it through the manual process with the GRA / Sentinal team.

ONCE ONLY. A transaction may be declined or charged back exactly once, ever. A second attempt against a transaction already carrying a decline or chargeback is refused with HTTP 400, whichever endpoint is used. Partial reversals are expressed through the amount fields on that single call, not through repeated calls — decide the full reversal amount before you call.

THE VERIFIED FIELDS ARE MATCHED WITH TOLERANCE, NOT EXACTLY. verifiedTransactionAmount must be within 0.01 of the recorded amount; verifiedTransactionTimestamp within 60 seconds of the recorded transaction time; verifiedRetrievalRefNumber must match exactly. A mismatch is refused with HTTP 400 and the expected value is named in the message.
CURRENCY. currency must be GHS, or the original foreign currency of the transaction where one was recorded. Amounts submitted in the foreign currency are converted using the exchange rate recorded on the ORIGINAL transaction, not a current rate.
REMAINING BALANCE. refundAmount and refundVatAmount are each validated against the remaining balance — amounts already refunded plus amounts currently pending cannot exceed the transaction amount, and the same applies independently to the VAT portion. The reservation is atomic, so concurrent attempts cannot over-reverse.

Refund linkage
The response returns a ref (the decline record's reference) and a refundId. Echo the ref as declineChargebackRef, and use refundId, when calling the refund endpoint (6.6). A deprecated alias POST /v1/transaction/decline runs the identical validation and handler; migrate to /declineTransaction.

Chargebacks, refunds & returns — build now, process finalising
A GRA regulatory committee is defining the internal organisational processes for chargebacks and refunds. That work concerns how the GRA handles a dispute internally; it does not change the API.
Payment Processors should therefore include the decline, chargeback and refund endpoints in this development cycle. The endpoints are live and their request and response contracts are stable and will not change when the committee's process is adopted — so building them now means NO further code changes are required at that point. Deferring them means a second integration cycle later.
Until the process is finalised, dispute reconciliation and the resulting VAT position are settled MANUALLY in coordination with the GRA / Sentinal team. Calling the endpoints records the event correctly; the downstream settlement is what remains manual.

6.5 POST /v1/transaction/chargeback

Records a chargeback against a previously verified transaction and creates a linked PENDING refund. HMAC authenticated; X-MERCHANT-KEY is required. A chargeback is the mechanism for reversing a transaction after settlement — for same-day reversals use declineTransaction (Section 6.4) instead.

Request Body

Field	Type	Req.	Rules & Description
ref	string	No	External reference (max 256). A value is generated if omitted; the generated value is returned and must be echoed to the refund endpoint.
transactionRef	string	Yes	Reference of the transaction being charged back (max 64).
currency	string	Yes	ISO 4217, uppercase (3 characters).
refundAmount	number	Yes	Amount to reverse (≥ 0). Cannot exceed the transaction amount less amounts already refunded or pending.
refundVatAmount	number	Yes	VAT portion to reverse (≥ 0). Subject to the same remaining-balance validation.

Field	Type	Req.	Rules & Description
reason	string	Yes	CANCELLATION | TAX_CORRECTION | PARTIAL. See the reason table below.
verifiedTransactionAmount	number	Yes	Must match the recorded transaction amount.
verifiedRetrievalRefNumber	string	Yes	Must match the recorded RRN (max 64).
verifiedTransactionTimestamp	string	Yes	ISO 8601; must match the recorded transaction time.

Reason Codes

Reason	Use	Resulting Status
CANCELLATION	The underlying purchase was cancelled and the transaction reversed.	APPROVED (D+1 to D+12)
PARTIAL	Part of the transaction value is being reversed.	APPROVED (D+1 to D+12)
TAX_CORRECTION	Correction of the VAT position on an otherwise valid transaction.	PROCESSED

Timing Rules

The permitted window is determined by the calendar-day difference between the transaction time and the request, evaluated in Ghana local time (GMT). D+0 is the transaction day.

Window	Accepted	Chargeback Status	Refund Status
D+0 (same day)	No — HTTP 400	—	Use declineTransaction (Section 6.4) instead.
D+1 to D+12	Yes	APPROVED	PENDING — call the refund endpoint (Section 6.6).
D+13 and later	Yes	PROCESSED	PENDING — settled through the manual process.

Example Request

Scenario: a GHS 250.75 transaction verified on 9 July is charged back on 14 July (D+5) because the purchase was cancelled. The full amount and its VAT are reversed.

```
POST /v1/transaction/chargeback
Authorization: Bearer <accessToken>
X-PARTNER-ID: PARTNER-abc123
X-MERCHANT-KEY: <merchant apiKey>
X-TIMESTAMP: 2026-07-14T09:30:00.000Z
X-SIGNATURE: <HMAC-SHA256 per Section 3.4>
Content-Type: application/json

{
  "ref": "CB-2026-0001",
  "transactionRef": "TXN-2026-0001",
  "currency": "GHS",
  "refundAmount": 250.75,
  "refundVatAmount": 50.15,
  "reason": "CANCELLATION",
  "verifiedTransactionAmount": 250.75,
  "verifiedRetrievalRefNumber": "123456789012",
  "verifiedTransactionTimestamp": "2026-07-09T11:58:00+00:00"
}
```

Response — 200 OK (content)

Field	Type	Description
chargebackId	string	Identifier of the chargeback record (24-hex ObjectId).
transactionId	string	Identifier of the underlying transaction.
refundId	string	Identifier of the PENDING refund created by this call. Supply it to the refund endpoint (Section 6.6).
ref	string	The chargeback's external reference — supplied or generated. Echo this as declineChargebackRef on the refund call.
refundStatus	string	PENDING once the chargeback is accepted.
status	string	Chargeback status: APPROVED or PROCESSED, per the timing and reason rules above.
chargebackReason	string	Echo of the submitted reason.
transactionRef	string	Reference of the underlying transaction.
requestCurrency	string	Currency as submitted.
requestRefundAmount	number	Reversal amount as submitted.
requestVatAmount	number	VAT reversal amount as submitted.
convertedRefundAmount	number	Reversal amount in the local currency (GHS).
convertedVatAmount	number	VAT reversal amount in the local currency (GHS).
exchangeRateUsed	number	The rate applied where the request currency is not GHS. OMITTED ENTIRELY from the response for a GHS request — it is not returned as 1. Previous revisions stated otherwise and their example response was wrong. Read it as optional.
message	string	Human-readable outcome, including the D+n position of the request.

Example Response

```
200 OK

{
  "success": true,
  "message": "Chargeback processed successfully.",
  "content": {
    "chargebackId": "66a1f2c4e8b9d0a1c2e3f4a5",
    "transactionId": "66a1f2c4e8b9d0a1c2e3f4a6",
    "refundId": "66a1f2c4e8b9d0a1c2e3f4a7",
    "ref": "CB-2026-0001",
    "refundStatus": "PENDING",
    "status": "APPROVED",
    "chargebackReason": "CANCELLATION",
    "transactionRef": "TXN-2026-0001",
    "requestCurrency": "GHS",
    "requestRefundAmount": 250.75,
    "requestVatAmount": 50.15,
    "convertedRefundAmount": 250.75,
    "convertedVatAmount": 50.15,
    "message": "Chargeback approved (D+5). A PENDING refund has been created."
  }
}
```

Validation

The three verified* fields must match the recorded transaction exactly; a mismatch is rejected with HTTP 400 and the expected value is named in the message. The reversal is also validated against the remaining balance —

the sum of amounts already refunded and currently pending cannot exceed the transaction amount, and the same applies to the VAT portion.

/chargeback and /chargeBack are DIFFERENT endpoints

The endpoint specified in this section is POST /v1/transaction/chargeback, all lower case. A separate legacy endpoint exists at POST /v1/transaction/chargeBack, with a capital B. It is NOT a spelling-tolerant alias: it takes a completely different body, accepting only ref and a MANDATORY evidence array of two or three files, and none of the transactionRef, currency, amount, reason or verified* fields documented here.

Sending the body from this section to /chargeBack fails validation with HTTP 400 naming fields you believe you sent correctly. Integrate against the lower-case path. Sentinal error messages and hints occasionally refer to "/chargeBack" generically; read those as referring to the chargeback operation, not as an instruction to change the path you call.

6.6 POST /v1/transaction/refund

Processes a refund against a prior decline or chargeback. HMAC authenticated; X-MERCHANT-KEY is required.

> **Build the integration now — the process is manual for the present**
> The refund endpoint is live and its contract is stable. Payment Processors should build and test the refund integration as part of this delivery, alongside chargeback (Section 6.5).
> The internal organisational refund process is still being defined by a GRA regulatory committee, so refund settlement and the associated VAT position are currently handled MANUALLY in coordination with the GRA / Sentinal team. Calling the endpoint records the refund; it does not yet trigger an automated settlement.
> The request and response contract will not change when that process is finalised. Building it now means no additional code changes are required at cut-over, and the switch can be made at any time. Deferring it means returning for a second integration cycle.

Request Body

Field	Type	Req.	Description
ref	string	Yes	External reference for the refund (max 256).
refundId	string	Yes	The refundId returned by declineTransaction / chargeback (24-hex ObjectId).
declineChargebackRef	string	Yes	Echo of the ref returned by the decline/chargeback response (max 256). Validated against the decline/chargeback record.
transactionRef	string	Yes	Reference of the transaction being refunded (max 130).
refundAmount	number	Yes	Amount to refund (positive).
refundTime	string	Yes	ISO 8601 datetime when the refund was initiated (e.g. 2026-07-09T12:12:35.123+00:00).

Example Request

Continuing the chargeback example above: ref and refundId are taken from the chargeback response, and the chargeback's ref is echoed as declineChargebackRef.

```
POST /v1/transaction/refund
Authorization: Bearer <accessToken>
X-PARTNER-ID: PARTNER-abc123
X-MERCHANT-KEY: <merchant apiKey>
X-TIMESTAMP: 2026-07-14T09:35:00.000Z
X-SIGNATURE: <HMAC-SHA256 per Section 3.4>

{
  "ref": "RF-2026-0001",
  "refundId": "66a1f2c4e8b9d0a1c2e3f4a7",
  "declineChargebackRef": "CB-2026-0001",
  "transactionRef": "TXN-2026-0001",
  "refundAmount": 250.75,
  "refundTime": "2026-07-14T09:34:50.123+00:00"
}
```

Response — 200 OK (content)

Field	Type	Description
ref	string	The refund's external reference, as submitted.

Field	Type	Description
transactionRef	string	Reference of the underlying transaction.
transactionId	string	Identifier of the underlying transaction.
refundAmount	number	Amount refunded by this call, in GHS.
refundedVAT	number	VAT portion refunded by this call.
refundedMerchantAmount	number	Merchant portion refunded by this call.
totalRefunded	number	Cumulative amount refunded against this transaction, including this call.
remainingRefundable	number	Transaction amount less totalRefunded. Zero means fully refunded.
originalTotalAmount	number	The transaction's original amount.
refundPercentage	number	This refund as a percentage of the original amount, to two decimals.
status	string	Refund status after processing.
refundTime	string	The refundTime you submitted.
processedAt	string	When Sentinal processed the refund.
declineReason	string | null	Reason carried from the originating decline or chargeback.
currency	string	GHS.
message	string	Human-readable outcome, naming the remaining refundable amount.

Validation

Condition	HTTP	Meaning & Resolution
declineChargebackRef does not resolve	422	The value matches the ref of no decline or chargeback recorded for YOUR Payment Processor. Echo the exact ref returned in the decline or chargeback response, including one Sentinal generated for you.
refundAmount not positive	400	refundAmount must be strictly greater than 0.
Amount exceeds remaining refundable	400	The refund would take the cumulative total past the transaction amount.
Transaction not found for this merchant	404	The transaction must belong to BOTH the authenticated Payment Processor AND the merchant identified by X-MERCHANT-KEY.
refundId malformed	400	refundId must be a 24-character hexadecimal identifier.

> **refundId and declineChargebackRef are both required, and are not interchangeable**
> refundId is the identifier of the PENDING refund record the decline or chargeback created. declineChargebackRef is the external reference of the decline or chargeback itself. Both come back in the same response and both must be sent. Supplying one in place of the other fails with a 422 naming declineChargebackRef, which is easy to misread as a problem with the refund rather than the field mapping.

6.7 Timeout Behaviour

Transactional endpoints apply a server-side processing budget. Payment Processors should set a client-side timeout of at least 60 seconds. On timeout, call POST /v1/transaction/verifyTransactionStatus (or the relevant status endpoint) to determine the outcome before retrying, to avoid duplicate submissions. See Section 11.4 for the recommended retry-with-backoff workflow.

6.8 POST /v1/rates — Exchange Rates

Returns Sentinal's reference exchange rates. HMAC authenticated with the Payment Processor secret; X-MERCHANT-KEY is not sent. These are the rates Sentinal itself applies when determining VAT on a multi-currency (MCA) transaction, so retrieving them lets a Payment Processor reproduce the figure verifyTransaction will return rather than discovering it after the fact (Section 6.1).

Request Body

Field	Type	Req.	Description
source	string	Yes	Must be the literal value "system".
channel	string	Yes	Must be the literal value "API".

Example Request

```
POST /v1/rates
Authorization: Bearer <accessToken>
X-PARTNER-ID: PARTNER-abc123
X-TIMESTAMP: 2026-07-09T12:00:00.000Z
X-SIGNATURE: <HMAC-SHA256 per Section 3.4>
Content-Type: application/json

{ "source": "system", "channel": "API" }
```

Response — 200 OK

Field	Type	Description
effectiveDate.start	string	ISO 8601 — the date the active rates were published.
effectiveDate.end	string	ISO 8601 — six days after the start date.
rates[].currencyCode	string	ISO 4217 code.
rates[].currencyName	string	English currency name.
rates[].rate	string	The rate, as a STRING with two decimal places — not a number. Parse accordingly.

This response is NOT wrapped in the standard envelope

Unlike every other endpoint in this document, /v1/rates returns its payload at the top level: { "effectiveDate": {...}, "rates": [...] }. There is no success / message / content wrapper (Section 11.1). Do not look for content.rates — read rates directly.

Rates are published weekly and carry a seven-day effective window. A 500 is returned where no active rates are configured. The rate values are strings, and rate is rounded to two decimal places, so it is a reference for reconciliation and forecasting — the authoritative conversion for a given transaction is the exchangeRateUsed returned by verifyTransaction.

7. VAT Rules Distribution

Sentinal maintains a versioned, cryptographically hashed snapshot of the active VAT ruleset for each Payment Processor. Payment Processors retrieve this snapshot to apply VAT determination client-side and to reference the active snapshot in transaction submissions for audit linkage. The ruleset is organised in three tiers, applied in order of specificity: merchant overrides (Tier 3), MCC rules (Tier 2), and classification rules (Tier 1).

7.1 GET /v1/psp/vatRules

Returns the active snapshot for the authenticated Payment Processor. Uses the same HMAC authentication as the transactional endpoints. Cache the snapshot until its expiresAt and re-fetch then; see the refresh cadence in Section 7.4.

Response — 200 OK (content)

Field	Type	Description
signature	string	Snapshot signature: SNT-GH-YYYYMMDD-{16 hex, uppercase}.
countryCode	string	"GH".
generatedAt	string	ISO 8601 generation timestamp.
expiresAt	string	ISO 8601 advisory cache expiry.
defaultVatRate	number	The country default rate, as a PERCENTAGE (e.g. 20). Informational only: do NOT apply it as a fallback rate — there is no country-default tier (Section 7.3, rule 5).
localMerchantVatExempt	boolean	Whether local merchants are VAT exempt.
rulesetHash	string	SHA-256 of the canonical JSON, for integrity verification.
classificationRules	array	Tier 1 — one rule per classification (Type).
mccRules	array	Tier 2 — one rule per configured MCC fee.
merchantOverrides.exemptMerchants	array	Tier 3a — exempt merchants (hashed identifiers). NATIONAL: carries every approved merchant on the platform holding an exemption or a NOT_COLLECTED status, not only merchants linked to you. See Section 5.2.1.
merchantOverrides.statusOverrides	array	Tier 3b — merchants deviating from their rule default. Scoped to merchants linked to YOUR Payment Processor only.

vatRate is a PERCENTAGE, not a decimal fraction — correction to versions 1.5.6 and 1.5.7

Every vatRate in this snapshot, and vatTreatment.vatRate in the registration response, is expressed as a PERCENTAGE. A 20% rate is delivered as the number 20, not 0.2. Previous revisions of this document described it as "a decimal (e.g. 0.2)" and their worked examples used 0.2. That is wrong.

The consequence of following the old text is severe and silent: a Payment Processor that multiplies the transaction amount by the delivered value of 20 computes VAT at 2000%, over-collecting by a factor of one hundred. Unlike a signature error, nothing rejects the request — the processor's own figure is simply wrong, and it will disagree with the amount verifyTransaction returns.

Apply the rate as amount x vatRate / 100. Sentinal performs this division internally when it computes VAT, which is why its own figures are correct; only the published field and the previous documentation disagreed.

> defaultVatRate at the top of the snapshot is expressed the same way. It remains informational and must not be applied as a fallback (rule 5, Section 7.3).

classificationRules[] item

Field	Type	Description
classification	string	Merchant Type (Section 9).
vatRate	number	VAT rate as a PERCENTAGE, e.g. 20 means 20%. Divide by 100 before applying. See the units warning below.
defaultCollectionStatus	string?	COLLECTED | NOT_COLLECTED | POTENTIALLY_COLLECTED.
collectionRules	array?	Payment-method-specific overrides (present only when configured). Item shape below.

mccRules[] item

Field	Type	Description
mccCodes	string[]	MCC codes covered by this rule. One entry is emitted per configured fee row, so the array mirrors the source configuration and rows are NOT merged. Take the first entry whose mccCodes contains your MCC and stop.
classification	string?	Associated Type, when applicable.
vatRate	number	VAT rate as a PERCENTAGE, e.g. 20 means 20%. Divide by 100 before applying.
defaultCollectionStatus	string?	Collection status enumeration.
collectionRules	array?	Payment-method-specific overrides (present only when configured). Item shape below.

collectionRules[] item (classificationRules and mccRules)

collectionRules is present on a rule only where payment-method-specific overrides are configured; it is omitted entirely, not returned empty, when none are. Each item overrides the rule's defaultCollectionStatus for the payment methods it names.

Field	Type	Description
paymentMethods	string[]	Payment methods this override applies to, drawn from the same set accepted on verifyTransaction: CREDIT_CARD, DEBIT_CARD, WALLET, SAVINGS, BANK_TRANSFER, PAYLATER.
collectionStatus	string	The collection status to apply when the transaction uses one of those methods, in place of the rule's defaultCollectionStatus.

> **Applying collectionRules, including split tender**
> A rule matches when ANY payment method on the transaction appears in its paymentMethods list. Where more than one item matches — which split tender can cause — the MOST SPECIFIC item wins, meaning the one naming the fewest payment methods. This makes the outcome deterministic and independent of the order the items appear in the snapshot.
> Where collectionRules is present but no item matches, the rule's defaultCollectionStatus applies. A merchant carrying a NOT_COLLECTED status is never lifted out of it by a collectionRule.

merchantOverrides.exemptMerchants[] item

Field	Type	Description
registrationNumberHash	string?	SHA-256 of the merchant registration number (if set).
organizationTinHash	string?	SHA-256 of the merchant TIN (if set).
collectionStatus	string	NOT_COLLECTED.
reason	string	VAT_EXEMPT | KNOWN_MERCHANT | MANUAL_OVERRIDE.

merchantOverrides.statusOverrides[] item

Field	Type	Description
registrationNumberHash	string?	SHA-256 of the merchant registration number (if set).
organizationTinHash	string?	SHA-256 of the merchant TIN (if set).
overrideStatus	string	The merchant's deviating collection status.
vatRate	number?	Applicable VAT rate for the merchant.
appliedMccCode	string?	The MCC rule that would otherwise apply (with mccDefaultStatus).
mccDefaultStatus	string?	The default status of that MCC rule.
appliedClassification	string?	The Type rule that would otherwise apply.
classificationDefaultStatus	string?	The default status of that Type rule.

Error Responses

HTTP	Condition	Meaning & Resolution
401	Unauthorized	Invalid or expired access token / HMAC signature.
400	Invalid timestamp	X-TIMESTAMP outside the 5-minute window.
404	Snapshot not ready	No active snapshot exists for this Payment Processor yet; retry later.
500	Snapshot generation error	Internal error; retry later.

> **Rate limit**
> Recommended maximum 10 requests per hour per Payment Processor. This is advisory guidance, not an enforced quota — no rate limiter rejects a caller that exceeds it. Cache the snapshot until expiresAt. The response carries Cache-Control: max-age=3600, which is a transport hint only and is unrelated to the ruleset's own validity window; drive your refresh from expiresAt, not from the header.

7.2 Integrity Verification (rulesetHash)

Payment Processors may verify snapshot integrity by recomputing the SHA-256 of the canonical JSON of the ruleset payload and comparing it with rulesetHash. Previous revisions did not say which fields the payload contains, which made the check impossible to reproduce. It is EXACTLY these six, and no others:

Included in the hash	Excluded from the hash
countryCode	signature
defaultVatRate	generatedAt
localMerchantVatExempt	expiresAt

Included in the hash	Excluded from the hash
classificationRules	rulesetHash itself
mccRules	—
merchantOverrides	—

Canonicalisation rules for the hash:

- Object keys are sorted alphabetically at EVERY level of nesting, including inside the objects held in classificationRules, mccRules and the two merchantOverrides arrays.
- Array ORDER is preserved exactly as delivered — arrays are never sorted. The server emits them deterministically, so a re-ordered array is a different ruleset, not the same one.
- Compact separators, no insignificant whitespace, and the number formatting described in Section 3.5.
- Absent fields are absent, not null. A rule carrying no collectionRules omits the key rather than emitting an empty array, and the hash reflects that.

If the hash does not match, do not reject the snapshot

rulesetHash is an integrity aid, not an authorisation control. A mismatch most often means a canonicalisation difference on your side rather than a corrupted snapshot — the same trailing-zero and key-ordering issues described in Section 3.5 apply here. Apply the snapshot, log the mismatch, and raise it with your Sentinal technical contact. Refusing to process transactions because a hash did not reproduce would be a far worse outcome than applying a snapshot that is almost certainly intact.

7.3 Applying the Snapshot (Resolution Order)

Precondition — establish that the transaction is in scope first

Sentinal's mandate covers CROSS-BORDER transactions only. Before any rule below is evaluated, determine whether the transaction is cross-border. A genuinely domestic Ghana transaction is out of scope: no VAT is applied and no rule in this section is consulted.

This is why the snapshot's localMerchantVatExempt field requires no processor-side logic. The out-of-scope determination is made ahead of the ruleset, in line with the mandate and regulations issued to Payment Processors — not by evaluating a rule.

One important exception applies where an aggregator sits in the flow. Read the next note before implementing the out-of-scope test.

Exception — a cross-border transaction that PRESENTS as local (aggregator in the flow)

An aggregator collects locally, under its own Ghanaian business details, and settles an offshore merchant outside the payment rail. The upstream provider therefore registers the merchant using the AGGREGATOR's details, so the transaction reaches it carrying a Ghana-registered merchant and looks domestic — even though the underlying supply is cross-border and in scope.

A naive 'merchant is local, therefore out of scope' test will discard exactly these transactions. Do not discard them. Sentinal requires visibility of them, so an upstream provider must still submit the transaction on verifyTransaction even though the registered merchant appears local.

The VAT obligation sits with the AGGREGATOR, not with the upstream provider. The aggregator is the party required to apply and remit the VAT on the underlying cross-border supply, and integrates with Sentinal in its own right for that purpose (Sections 2.1 and 5.2, and the Aggregator Flow in Appendix E). The upstream provider

submits for visibility and applies the treatment Sentinal returns — it must not collect VAT a second time on the same supply.
In short: 'local-looking merchant' is not by itself a reason to suppress a submission. Submit the transaction and let the returned collection status determine whether you collect. verifyTransaction remains canonical (Section 5.2.1).

Having established that the transaction is cross-border and therefore in scope, evaluate the snapshot rules strictly TOP-DOWN, from rule 1 to rule 5. Apply the FIRST rule that matches the merchant or transaction and STOP — do not evaluate any later rule, and never merge results from more than one rule.

Order	Rule to test	If it matches, apply
1	merchantOverrides.exemptMerchants — hashed registrationNumber or organizationTin equals the merchant's.	collectionStatus NOT_COLLECTED. Stop.
2	merchantOverrides.statusOverrides — hashed identifier equals the merchant's.	overrideStatus (and vatRate, if present). Stop.
3	mccRules — mccCodes contains the transaction/merchant MCC.	That rule's vatRate and defaultCollectionStatus. Stop.
4	classificationRules — classification equals the merchant's Type.	That rule's vatRate and defaultCollectionStatus. Stop.
5	No rule matched.	Do NOT collect VAT — treat as NOT_COLLECTED. Do not apply a default rate.

Safe default — do not charge VAT when nothing matches

If a transaction matches no rule (rule 5), the correct action is to NOT collect VAT (NOT_COLLECTED). Integrators are not expected to know or apply the system's default or local VAT rate; VAT is charged only where a rule in tiers 1–4 explicitly determines it. This fail-safe ensures a transaction is never over-charged because a rule was missing from the snapshot.

Rule 2 (statusOverrides) is in scope — correction to earlier guidance

An earlier revision of this document carried a manually added remark that rule 2, merchantOverrides.statusOverrides, was not in scope for the Ghana deployment. That remark has been withdrawn. Following consultation, Payment Processors are advised to implement rule 2 alongside the other tiers.

The reason is that a status override is the only mechanism able to express a merchant that deviates from its own MCC. An MCC may sit outside the scope of VAT collection as a whole while individual merchants carrying that MCC remain in scope — Education is the working example: the MCC is out of scope, yet a cross-border digital-learning provider such as Udemy would be in scope and must still be collected on.

Implement rule 2 exactly as specified above: it is evaluated after exemptMerchants and before the MCC and classification tiers, and the first match stops evaluation. A Payment Processor that skips it will apply the MCC-level treatment to a merchant that Sentinal has deliberately overridden, and its locally computed VAT will disagree with the amount returned by verifyTransaction.

Which rules apply to you

Rules 3 and 4 are alternative classification tiers serving different processor types. Implement the one that matches how your platform identifies a merchant; rules 1, 2 and 5 apply to everyone.

Rule	Card processors	Other Payment Processors
1 — exemptMerchants	Required.	Required.
2 — statusOverrides	Required. See the correction above.	Required.
3 — mccRules	Required. This is the classification tier for card processors, which identify merchants by MCC.	Required where an MCC is held.
4 — classificationRules	Out of scope. Card processors work from the MCC carried on the scheme message, not from a Sentinal Type.	Required where merchants are held against a Sentinal Type rather than an MCC.
5 — no match	Required. Do not collect.	Required. Do not collect.

Rule 4 is out of scope for card processors only

A card processor receives an MCC on every authorisation message and has no Sentinal classification (Type) to match on, so rule 4 can never fire for it and may be omitted. Rule 3 is its classification tier.

This exemption is specific to card processors. Any Payment Processor that registers merchants against a Type rather than an MCC must implement rule 4, or those merchants will fall through to rule 5 and VAT will be under-collected.

Worked Examples

Each example evaluates 1 → 5 and applies the first match.

Merchant / transaction	First match (top-down)	Result
Merchant's hashed registrationNumber is in exemptMerchants; its MCC 5734 also has a 20% mccRule.	Rule 1 (exempt)	NOT_COLLECTED — rules 3/4 are never consulted.
Not exempt; hashed id is in statusOverrides with overrideStatus POTENTIALLY_COLLECTED.	Rule 2 (override)	POTENTIALLY_COLLECTED (with the override's vatRate).
No merchant override; transaction MCC 5734 is in an mccRule (20%, COLLECTED).	Rule 3 (MCC)	vatRate 20 (i.e. 20%), COLLECTED — classificationRules skipped.
No override; MCC not in any mccRule; merchant Type (e.g. SOFTWARE_AND_SAAS) matches a classificationRule.	Rule 4 (Type)	That rule's vatRate and status.
No override; MCC and Type match nothing.	Rule 5 (default)	NOT_COLLECTED — VAT is not charged when nothing matches.

Key point

The rules are ordered most-specific (a named merchant) to least-specific (the no-match fail-safe). A merchant in exemptMerchants is always NOT_COLLECTED even if its MCC or Type rule would otherwise collect VAT — because rule 1 matches first and evaluation stops there.

Acting on the Resolved Collection Status

Collect VAT only where the resolved status is COLLECTED. For any other value, do not collect.

Status	Collect VAT?	Still submit to verifyTransaction?
COLLECTED	Yes	Yes.
NOT_COLLECTED	No	Yes — Sentinal records the transaction and the reason no VAT was due.

Status	Collect VAT?	Still submit to verifyTransaction?
POTENTIALLY_COLLECTED	No	Yes — submission is REQUIRED. The transaction is flagged for reconciliation.

POTENTIALLY_COLLECTED must still be submitted
This status means VAT may already have been collected further up the chain, so collecting again would double-charge. Do not collect — but do submit the transaction to verifyTransaction exactly as you would any other. Sentinal uses these records to reconcile the position and determine whether VAT was ultimately accounted for. Suppressing them leaves a gap that cannot be reconstructed.

On verifyTransaction, echo the active snapshot signature in X-RULES-SIGNATURE for audit linkage. This is advisory: a missing or stale signature is never rejected.

7.4 Snapshot Lifecycle

A Payment Processor has exactly one active snapshot at a time. When the ruleset is regenerated, the previous snapshot is retired and the new one becomes active in a single atomic step, so a Payment Processor is never left without a servable ruleset — if generation fails, the previous snapshot simply stays active. The expiresAt field is an advisory cache hint: the server never expires a snapshot by time, only by replacement, so the active snapshot remains valid and servable after expiresAt has passed. Treat expiresAt as 'check for a newer one now', not as 'this one has become invalid'.

Refresh Cadence

The validity window differs by environment. Always read expiresAt from the snapshot itself rather than hard-coding an interval.

Environment	Ruleset validity	Refresh
Production	Monthly — expiresAt is 00:01 UTC on the first day of the next calendar month (Ghana is UTC+0, so this is 00:01 local).	Re-fetch at or after expiresAt. Note this is a calendar boundary, not a rolling 30 days: a snapshot issued on the 28th expires a few days later.
UAT	expiresAt is generatedAt plus 24 hours, and the ruleset is regenerated daily to support testing.	Re-fetch at or after expiresAt; do not assume the production cadence while testing.

Do not hard-code a 30-day cache
The production window closes on the calendar month boundary, so the life of any given snapshot is between one day and one month depending on when it was issued. A fixed 30-day timer will hold a stale ruleset past its replacement. Drive the refresh from expiresAt.

Re-fetch GET /v1/psp/vatRules at or after expiresAt, and additionally whenever you receive a merchant.vat_updated webhook (Section 8).

If you do not consume the merchant webhook, the refresh is your only path

A merchant's collection status or exemption can change at any point within a validity window — for example when a merchant is granted an exemption. The merchant.vat_updated webhook is what makes that change available immediately.

A Payment Processor that does not subscribe to the webhook will not see such a change until it next re-fetches the snapshot. On the production cadence that can be several weeks, during which its locally computed VAT will disagree with the amount returned by verifyTransaction. If the webhook is not implemented, re-fetch the snapshot more frequently than expiresAt requires, and treat verifyTransaction as canonical in the interim (Section 5.2.1).

8. Webhooks

Sentinal delivers event notifications to processor-registered HTTPS endpoints. Each subscription has its own signing secret (prefixed whsec_), returned once at creation and rotatable. Deliveries are HMAC-signed so Payment Processors can authenticate them. Payment Processors create and maintain their own subscriptions through the endpoints in Section 8.6 — subscriptions are not provisioned on your behalf.

8.1 Delivery Headers

Header	Description
Content-Type	application/json
X-PARTNER-ID	The Payment Processor identifier.
X-EVENT-TYPE	The event type (e.g. merchant.vat_updated).
X-WEBHOOK-ID	Unique delivery/message identifier.
X-WEBHOOK-ATTEMPT	Delivery attempt number.
X-TIMESTAMP	ISO 8601 send time (verify within 5 minutes).
X-SIGNATURE	HMAC-SHA256 signature — see 8.2.

8.2 Signature Verification

Recompute the signature using your subscription secret and compare with X-SIGNATURE using a timing-safe comparison. Reject deliveries whose X-TIMESTAMP is more than five minutes from your clock. The canonical-JSON rules in Section 3.5 apply to the received payload, and the hash is taken over the body EXACTLY as delivered to you. path is the PATH COMPONENT of the webhook URL you registered — no scheme, host, port or query string; for a URL registered as https://psp.example.com/sentinal/hooks the value is /sentinal/hooks, and for a URL with no path it is '/'. The method segment is always the literal 'POST'.

```
bodyHash     = lowerHex( SHA256( canonicalJson(body) ) )
stringToSign = "POST" + ":" + path + ":" + X-EVENT-TYPE
               + ":" + bodyHash + ":" + X-TIMESTAMP
expected     = lowerHex( HMAC_SHA256( subscriptionSecret, stringToSign ) )
```

8.3 Events in Scope

The event type appears in the X-EVENT-TYPE header. One event is in scope for the Ghana deployment:

Event type (X-EVENT-TYPE)	Purpose
merchant.vat_updated	A merchant's collection status or VAT-exemption flag has changed. This is the event the VAT integration depends on — see Sections 8.4 and 7.4.

Subscribe to merchant.vat_updated explicitly — the default is every event

The platform also emits transaction, decline, chargeback and refund lifecycle events. They are NOT part of the Ghana integration and this document does not specify them; their payloads may change without a revision of this specification.

This matters at subscription time. The events field on POST /v1/webhook/subscribe is OPTIONAL, and omitting it subscribes you to EVERY event type the platform emits, not to none. A processor that leaves it out will begin

receiving the full lifecycle stream — high volume, and undocumented. Always send events explicitly: ["merchant.vat_updated"]. See Section 8.6.

8.4 Event: merchant.vat_updated

Emitted when a merchant's collection status or VAT-exemption flag changes on the Sentinal platform. Delivered only to the Payment Processor that registered the merchant. All identifiers are hashed (SHA-256); no plain-text registration numbers or TINs are transmitted. On receipt, refresh the VAT rules snapshot (Section 7).

Webhook bodies are wrapped in an envelope — correction to versions 1.5.6 and 1.5.7

Every webhook is delivered inside the same envelope Sentinal uses for API responses, NOT as the bare event object shown in earlier revisions. The body is:

{ "success": true, "message": "Event merchant.vat_updated occurred.", "content": { ...the event fields... }, "metadata": { "eventId": "...", "eventType": "merchant.vat_updated", "timestamp": "...", "version": "v1" } }

The fields tabulated below therefore live under content — content.event, content.occurredAt, content.signature and content.merchant.collectionStatus, not event and merchant.collectionStatus at the top level. A consumer written against the previous documentation reads undefined for every field and silently ignores the notification, which is the worst possible failure mode for an exemption change.

The signature is computed over the ENVELOPE as transmitted, so verification is unaffected: hash the bytes you received. metadata.eventId is stable across delivery attempts of the same event and, with X-WEBHOOK-ID, gives you two options for idempotency.

Field	Type	Description
content.event	string	"merchant.vat_updated".
content.occurredAt	string	ISO 8601 timestamp of the change.
content.signature	string?	Active snapshot signature at the time of change (null if none).
content.merchant.registrationNumberHash	string?	SHA-256 of the registration number (omitted if unset).
content.merchant.organizationTinHash	string?	SHA-256 of the TIN (omitted if unset).
content.merchant.collectionStatus	string	New collection status.
content.merchant.vatExempt	boolean	New VAT-exemption flag.
content.merchant.reason	string	VAT_EXEMPT | KNOWN_MERCHANT | MANUAL_OVERRIDE | STATUS_CHANGE.

8.5 Delivery Security & Retries

- Webhook URLs must use HTTPS in production; localhost and private IP ranges are rejected.
- Each delivery carries an attempt counter (X-WEBHOOK-ATTEMPT); make your consumer idempotent on X-WEBHOOK-ID.
- Rotate the subscription secret periodically; the previous secret stops validating once rotated. Rotation is self-service — see Section 8.6.

8.6 Managing Your Subscription

Payment Processors create and maintain their own webhook subscriptions. All of these endpoints authenticate as the Payment Processor and are scoped to your own subscriptions — you cannot see or alter another processor's. Note the authentication asymmetry: the endpoints that CHANGE something require the full HMAC signature of Section 3.4, while the read-only endpoints require the bearer token and X-PARTNER-ID but not X-SIGNATURE.

Endpoint	Auth	Purpose
POST /v1/webhook/subscribe	Token + HMAC	Create a subscription. Returns the secret once.
GET /v1/webhook/subscriptions	Token only	List your subscriptions.
GET /v1/webhook/subscriptions/{id}	Token only	Retrieve one subscription.
PUT /v1/webhook/subscriptions/{id}	Token + HMAC	Change the URL, events or settings.
POST /v1/webhook/subscriptions/{id}/toggle	Token + HMAC	Enable or disable deliveries.
POST /v1/webhook/subscriptions/{id}/rotate-secret	Token + HMAC	Issue a new signing secret.
DELETE /v1/webhook/subscriptions/{id}	Token + HMAC	Remove the subscription.
GET /v1/webhook/subscriptions/{id}/statistics	Token only	Delivery success / failure counts.
POST /v1/webhook/test	Token + HMAC	Send a test delivery to your endpoint.

{id} is the 24-character hexadecimal subscriptionId returned at creation.

POST /v1/webhook/subscribe

Field	Type	Req.	Rules & Description
webhookUrl	string	Yes	Your HTTPS endpoint. Must be a valid URL; HTTPS is required in production, where localhost and private IP ranges are rejected.
events	string[]	No*	*Optional in the schema but effectively required in practice: OMITTING IT SUBSCRIBES YOU TO EVERY EVENT THE PLATFORM EMITS. Send ["merchant.vat_updated"].
description	string	No	Free text, max 500 characters.
settings.timeout	number	No	Delivery timeout in milliseconds, 1000–120000. Default 30000.
settings.retryEnabled	boolean	No	Default true.
settings.maxRetries	number	No	0–10. Default 3.
settings.includeFullPayload	boolean	No	Default true.
customHeaders	object	No	String key/value pairs added to each delivery — useful for routing through your own gateway.
ipWhitelist	string[]	No	Valid IP addresses permitted to receive the delivery.

Example Request

```
POST /v1/webhook/subscribe
Authorization: Bearer <accessToken>
X-PARTNER-ID: PARTNER-abc123
X-TIMESTAMP: 2026-07-09T12:00:00.000Z
X-SIGNATURE: <HMAC-SHA256 per Section 3.4>
Content-Type: application/json

{
  "webhookUrl": "https://psp.example.com/sentinal/hooks",
```

```
  "events": ["merchant.vat_updated"],
  "description": "Ghana VAT merchant updates"
}
```

Response — 201 Created (content)

Field	Type	Description
subscriptionId	string	24-hex identifier. Use it in the management paths above.
webhookUrl	string	Echo of the registered URL.
events	string[]	The event types actually registered, after validation. CHECK THIS — an unrecognised event name is dropped rather than rejected, so a typo yields a subscription that never fires.
secret	string	The signing secret, prefixed whsec_. RETURNED ONCE AND NEVER AGAIN — store it before you discard the response. If it is lost, rotate to obtain a new one.
active	boolean	true on creation.
createdAt	string	ISO 8601 creation timestamp.

The secret is shown once

The plaintext secret appears only in the creation response and in the rotate-secret response; it is encrypted at rest and no endpoint will return it again. Capture it into your secret store at the moment of creation.
The path segment you sign with is the pathname of the webhookUrl you register here — for https://psp.example.com/sentinal/hooks that is /sentinal/hooks (Section 8.2). Changing the URL later through PUT therefore changes the string you must verify against.

POST /v1/webhook/subscriptions/{id}/rotate-secret

Takes no request body. Returns content { secret } carrying the new whsec_ value, shown once. The previous secret stops validating immediately, so deploy the new one before rotating, or accept a short window in which deliveries fail signature verification and are retried.

POST /v1/webhook/subscriptions/{id}/toggle

Body { active: boolean }. Disabling suspends delivery without discarding the subscription or its secret — the appropriate action during planned maintenance on your endpoint, in preference to deleting and recreating.

POST /v1/webhook/test

Sends a test delivery to a subscription so you can verify your signature verification and endpoint reachability before go-live. It exercises the same signing path as a real delivery (Section 8.2), so a test that verifies correctly on your side is a genuine end-to-end check of the secret and the path segment.

Recommended order at onboarding

Subscribe with events set explicitly to ["merchant.vat_updated"]; store the returned secret; call POST /v1/webhook/test and confirm your endpoint validates the signature and returns 2xx; then confirm the subscription is active. Only then rely on the event for VAT state (Section 7.4).

9. Merchant Classification

Every merchant resolves to a Sentinal Type, which determines the applicable VAT rate and default collection status. A Type groups a set of Merchant Category Codes (MCC). The authoritative, per-processor rates and statuses are always those delivered in the VAT rules snapshot (Section 7); the table below reflects the current Ghana configuration.

> **Choosing Type vs MCC**
> Payment Processors and EMIs that do not carry MCC codes classify each merchant by its Type (the values in the first column below); the corresponding VAT rate and default collection status apply.
> Banks, TPPs, and Payment Processors that do carry MCC codes resolve VAT treatment by MCC code, which maps to the same Types. Both paths converge on the same VAT treatment for a given merchant.

Ghana VAT configuration — Types, default collection status, current VAT rate, and MCC codes:

Type	Rate	Default Status	MCC Codes
TELECOM	20%	COLLECTED	4813, 4814, 4815, 4816, 4821
TELECOM_EQUIPMENT_SPLIT	20%	POTENTIALLY_COLLECTED	4812
MEDIA_AND_ENTERTAINMENT	20%	COLLECTED	4899, 5815, 5817, 5818, 7311
SOFTWARE_AND_SAAS	20%	COLLECTED	5734, 7372, 7375, 7379
GAMING	20%	COLLECTED	5816, 5968
EDUCATION_ONLINE	20%	COLLECTED	8299
E_COMMERCE	20%	COLLECTED	5311, 5399, 5732, 5733, 5735, 5941, 5942, 5943, 5945, 5946, 5961, 5962, 5963, 5964, 5965, 5966, 5967, 5969, 5999, 7273, 7841
PROFESSIONAL_DIGITAL_SERVICES	20%	COLLECTED	7333, 7338, 7392, 7399, 8111, 8931
DIGITAL_PUBLISHING	20%	COLLECTED	2741, 5045, 5192
PLATFORM_HYBRID_SERVICES	20%	POTENTIALLY_COLLECTED	4722, 4789, 5811, 5812, 7829
EDUCATION_FORMAL	0%	POTENTIALLY_COLLECTED	8211, 8220, 8241, 8244, 8249, 8351
AGRICULTURAL_HYBRID	20%	POTENTIALLY_COLLECTED	0742, 0763, 0780
TRANSPORT_PLATFORM_FEE	20%	POTENTIALLY_COLLECTED	3000–4784 (travel, transport & lodging range; see mccRules for the full list)
FOOD_RETAIL_ZERO_RATED	0%	POTENTIALLY_COLLECTED	5411, 5422, 5451, 5462, 5499
FINANCIAL_SERVICES	0%	NOT_COLLECTED	4829, 6010, 6011, 6012, 6050, 6051, 6211, 6300, 6381, 6399, 6513, 6529, 6530, 6535, 6536, 6537, 6538, 6539, 6540, 6611, 6760, 9700, 9701, 9702
HEALTHCARE	0%	NOT_COLLECTED	8011, 8021, 8031, 8041, 8042, 8043, 8044, 8049, 8050, 8062, 8071, 8099
GAMBLING	0%	NOT_COLLECTED	7800, 7801, 7802, 7932, 7933, 7941, 7991, 7992, 7993, 7994, 7995, 7996, 7997, 7998, 9754
GOVERNMENT_SERVICES	0%	NOT_COLLECTED	9211, 9222, 9223, 9311, 9399, 9402, 9405, 9950
CONSTRUCTION_SERVICES	0%	NOT_COLLECTED	1520, 1711, 1731, 1740, 1750, 1761, 1771, 1799
HOSPITALITY_LOCAL	0%	NOT_COLLECTED	7011, 7012, 7032, 7033

Type	Rate	Default Status	MCC Codes
PERSONAL_SERVICES	0%	NOT_COLLECTED	7210, 7211, 7216, 7217, 7221, 7230, 7251, 7261, 7276, 7277, 7278, 7280, 7295, 7296, 7297, 7298, 7299
AUTOMOTIVE_SERVICES	0%	NOT_COLLECTED	5511, 5521, 5531, 5532, 5533, 5571, 5592, 5599, 7511, 7512, 7513, 7519, 7523, 7524, 7531, 7534, 7535, 7538, 7542, 7549
MISCELLANEOUS_SERVICES	0%	NOT_COLLECTED	4900, 7321, 7332, 7339, 7342, 7349, 7361, 7393, 7394, 7395, 8398, 8641, 8651, 8661, 8675, 8699, 8734, 8911, 8999
REPAIR_SERVICES	0%	NOT_COLLECTED	7622, 7623, 7629, 7631, 7641, 7692, 7699
LIVE_ENTERTAINMENT	0%	NOT_COLLECTED	7832, 7833, 7911, 7922, 7929, 7999
PHYSICAL_RETAIL_WHOLESALE	0%	NOT_COLLECTED	Selected physical retail/wholesale MCCs in the 2791–5998 range (e.g. 5300, 5309, 5310, 5331, 5411…5999); see mccRules for the full list.

Notes

• 20% Types are digital / cross-border categories where VAT is collected or potentially collected; 0% Types are local or exempt categories (NOT_COLLECTED or POTENTIALLY_COLLECTED).

• Local (in-country) merchants may be treated as exempt where the country rule localMerchantVatExempt applies.

• The complete MCC lists — including the full TRANSPORT_PLATFORM_FEE and PHYSICAL_RETAIL_WHOLESALE sets — are delivered in the mccRules of the VAT rules snapshot, which is the authoritative source.

10. Transaction Flows

Each flow is presented as a numbered step sequence. The corresponding sequence diagrams are provided in Appendix C.

10.1 Onboarding & First Transaction

Step	Actor → Action	Sentinal Endpoint / Result
1	Payment Processor obtains an access token (RSA-signed).	POST /v1/psp/token → accessToken (1h)
2	Payment Processor registers the merchant (HMAC).	POST /v1/client/registerClient → apiKey, secret, vatTreatment
3	Payment Processor retrieves the VAT rules snapshot (HMAC).	GET /v1/psp/vatRules → signature, tiers, rulesetHash
4	Payment Processor verifies a PURCHASE with X-MERCHANT-KEY.	POST /v1/transaction/verifyTransaction → status, vatAmount
5	On settlement, Payment Processor notifies credit.	POST /v1/transaction/creditInformation → 200

10.2 Decline → Refund (Same Day, D+0)

Step	Actor → Action	Sentinal Endpoint / Result
1	Payment Processor declines the transaction (HMAC).	POST /v1/transaction/declineTransaction → ref, refundId (PENDING)
2	Payment Processor initiates the refund, echoing the decline ref.	POST /v1/transaction/refund (declineChargebackRef = ref, refundId)
3	Sentinal validates and processes the refund.	200 → refund processed

10.3 Chargeback (D+1 to D+12)

Step	Actor → Action	Sentinal Endpoint / Result
1	Payment Processor records the chargeback (HMAC).	POST /v1/transaction/chargeback → refundId (PENDING; APPROVED ≤ D+12)
2	Payment Processor initiates the refund once resolved.	POST /v1/transaction/refund

10.4 Merchant Status Change Notification

Step	Actor → Action	Sentinal Endpoint / Result
1	A merchant's status is changed on the Sentinal platform.	Sentinal dashboard (not a Payment Processor API)
2	Sentinal delivers a signed webhook to the Payment Processor.	merchant.vat_updated (HMAC-signed)
3	Payment Processor verifies the signature and refreshes local state.	Verify X-SIGNATURE; refresh snapshot if needed

11. Error Handling & Troubleshooting

11.1 Response Envelope

All responses share a consistent envelope. Successful responses set success to true and carry the payload in content. Error responses set success to false and provide a human-readable message. Contrary to earlier revisions, content on an error is NOT always null: many endpoints return a diagnostic object holding a machine-readable code, a hint, and sometimes the offending values. Parse content defensively on errors and prefer content.code over string-matching the message, which is not a stable interface. Internal error details and stack traces are never exposed.

```
// Success
{ "success": true,  "message": "...", "content": { ... } }

// Error
{ "success": false, "message": "...", "content": null }
```

11.2 HTTP Status Reference

Status	Meaning	Typical cause
200 OK	Request succeeded.	Normal processing.
202 Accepted	Accepted for asynchronous processing.	Background operations.
400 Bad Request	Validation or business-rule failure.	Malformed body, invalid enum, timestamp skew, country mismatch. Also an ABSENT authentication header on most endpoints — see Section 3.2.2.
401 Unauthorized	Authentication failed.	Bad or expired token, invalid RSA or HMAC signature, unknown partner. An absent header returns 401 only on endpoints that do not declare a header schema, such as registerClient — see Section 3.2.2.
403 Forbidden	Not permitted for this actor.	Operating on a merchant outside your scope.
404 Not Found	Resource not found.	Unknown reference, or no active VAT snapshot yet.
422 Unprocessable	Semantically invalid reference.	declineChargebackRef does not resolve to a decline/chargeback.
409 Conflict	Duplicate or ambiguous.	A creditRef already recorded (Section 6.3); an X-ISSUER-ID matching more than one active issuer (Section 3.2.3).
500 Server Error	Unexpected internal error.	Retry with backoff; contact support if persistent. NOTE: verifyTransactionStatus returns 500 for an unknown transaction (Section 6.2), and verifyTransaction returns 500 for a paymentMethods sum mismatch (Section 6.1) — check the message before assuming a platform fault.
503 Unavailable	Backing store unavailable.	A database timeout on a transactional endpoint. Safe to retry with backoff (Section 11.4).

11.3 Common Errors

Symptom	Likely cause	Resolution
401 Invalid RSA signature (token)	X-SIGNATURE sent as base64 instead of hex; wrong string-to-sign; key mismatch.	Check the ENCODING FIRST: 512 lowercase hex characters, not 344 base64 characters (Section 3.3). Then confirm you signed X-PARTNER-ID + "|" + X-TIMESTAMP and that the registered public key pairs with your signing key.
Token request fails with no detail	The endpoint returns one generic message for every cause.	Work through Section 4 in order: signature encoding, then partner ID, then public key, then clock.
500 on a resubmitted ref, immediately after the first attempt	The first write has not settled, so the duplicate lookup cannot read it back (Section 6.1).	Back off and resend the IDENTICAL request. Do not allocate a new ref — that would create a real duplicate.
Sum of payment methods does not equal total amount	paymentMethods values do not add up to totalAmount.	Allocate the full amount across the methods, to the cent (Section 6.1).
400 verification must be performed on the same calendar day	The transaction was submitted after its own day ended in Ghana (GMT).	Submit on the transaction's calendar day; do not batch overnight (Section 6.1).
400 Only COLLECTED transactions can be declined or charged back	The transaction carries no collected VAT to reverse.	Check the status returned by verifyTransaction before attempting a reversal (Sections 6.4, 6.5).
400 already declined / charged back	A transaction may be reversed once only.	Determine the full reversal amount before the single call you are permitted (Sections 6.4, 6.5).
403 with code PROCESSOR_NOT_AUTHORIZED	The named issuer has not appointed your agent principal.	Have the issuer's appointment added by Sentinal (Section 3.2.3).
404 with code UNKNOWN_ISSUER	X-ISSUER-ID does not resolve to a registered issuer.	Confirm the exact identifier Sentinal holds for that issuer — the GIP code captured at onboarding (Section 3.2.3).
Webhook consumer sees no fields	Reading event / merchant at the top level of the body.	The event fields are under content — see the envelope correction in Section 8.4.
400 creditTime must be in ISO 8601 format with timezone offset	creditTime ends in "Z" rather than a numeric offset.	Send +00:00 rather than Z (Section 6.3).
401 Invalid HMAC signature (requests)	Body canonicalisation mismatch.	Apply the Section 3.5 rules; hash exactly the bytes you send; check number formatting.
400 Invalid or expired timestamp	Clock skew > 5 minutes.	Synchronise via NTP; generate X-TIMESTAMP just before signing.
400 issuerCountryCode must be GH	Wrong deployment country.	Send issuerCountryCode = GH.
400 PURCHASE requires X-MERCHANT-KEY	Missing merchant key.	Include X-MERCHANT-KEY for PURCHASE; or send participant fields for non-PURCHASE.
422 Invalid declineChargebackRef	Wrong echoed reference.	Echo the ref returned in the decline/chargeback response.
404 Snapshot not ready	No active snapshot yet.	Retry GET /v1/psp/vatRules after a short delay; cache when available.
Signature fails only for some payloads	Trailing-zero or unicode/slash escaping.	Normalise numbers (Section 3.5 Decimal Handling); disable unicode/slash escaping.

11.4 Connection Resilience — Retry with Exponential Backoff

Because Sentinal is reached over a site-to-site VPN, transient network conditions are normal. The recommended default for any connection failure, timeout, or 5xx response is a bounded, idempotent retry loop using exponential backoff with jitter. This absorbs momentary tunnel or server hiccups without hammering the API and without creating duplicate transactions. The delay matters in its own right and not only as courtesy to the server: an immediate resubmission of the same ref can itself provoke a 500 (Section 6.1).

Recommended Workflow

- 1. Send the request with a client-side timeout of at least 60 seconds and a stable, unique ref for the operation.
- 2. On a connection error, timeout, or 5xx response, do not fail hard — enter the backoff loop. Never resend immediately: always wait at least the first backoff interval, even when the failure looks instantaneous. A tight retry loop on verifyTransaction can be answered with a 500 that a short pause would have avoided (Section 6.1).
- 3. Before each retry of a transactional call, query the relevant status endpoint (e.g. verifyTransactionStatus). If the original request already succeeded, adopt that result and stop — this guarantees at-most-once effect. Note that this endpoint answers an unrecognised reference with a 500 rather than a 404 (Section 6.2), so treat an error here as 'not yet visible', not as 'never recorded'.
- 4. Otherwise wait the backoff interval (below), then resend the identical request with the same ref.
- 5. Give up after the maximum attempts, surface a clear operational error, and alert — the outstanding operation can be reconciled later via its ref.

Backoff Schedule (exponential, with jitter)

Attempt	Base delay	With ±20% jitter	Cumulative
1	0 s (immediate)	—	First attempt
2	1 s	0.8 – 1.2 s	~1 s
3	2 s	1.6 – 2.4 s	~3 s
4	4 s	3.2 – 4.8 s	~7 s
5	8 s	6.4 – 9.6 s	~15 s
6	16 s (cap 30 s)	12.8 – 19.2 s	~30 s — then stop & alert

Golden rules of resilient retries

• Retry connection errors, timeouts, and 5xx — never retry a 4xx without first correcting the request.
• Double the delay each attempt (exponential), add ±20% random jitter to avoid synchronised retry storms, and cap the delay (e.g. 30 s).
• Keep the ref stable across retries so the server and your reconciliation can correlate them (idempotency).
• Status-check before re-sending a transactional call, so a retry never creates a second transaction.

11.5 Connectivity: IPSEC / Site-to-Site VPN Troubleshooting

All Sentinal URLs are reachable only over the IPSEC / site-to-site VPN to the GRA network (Appendix B). If requests fail to connect at all — rather than returning an API error — the cause is almost always the VPN tunnel or routing, not the API. Work through the table below with your network team before raising an API issue.

Symptom	Likely cause	Resolution
Connection times out / host unreachable	VPN tunnel is down or not established.	Confirm the IPSEC tunnel is UP (IKE Phase 1 and Phase 2 SAs established); re-establish with your network team.
Cannot resolve *.gra.gov.gh	DNS not routed over the tunnel.	Use the GRA-provided DNS (or a hosts entry) and ensure DNS queries egress through the VPN.
"No route to host" / traffic not encrypted	Your source subnet is outside the tunnel's traffic selector.	Confirm your source IP range is included in the agreed encryption domain (proxy IDs) with GRA.
Connects, then large requests hang or truncate	MTU / fragmentation over the tunnel.	Enable TCP MSS clamping (e.g. MSS ~1350) on the tunnel interface.
Works, then drops after a period of inactivity	Phase 2 SA torn down on idle.	Enable Dead Peer Detection / keepalive; align IKE and IPSEC lifetimes both sides.
Tunnel never comes up (handshake fails)	Mismatched IKE parameters or pre-shared key.	Reconcile IKE version, encryption/integrity/DH group, lifetimes, and PSK with GRA networking.

Escalation Steps

- 1. Confirm tunnel status (Phase 1 and Phase 2) with your firewall/network team.
- 2. Verify a TCP connection to the API host on 443 (ICMP/ping may be blocked).
- 3. Verify the gra.gov.gh host resolves via DNS over the tunnel.
- 4. Confirm your source subnet is inside the agreed encryption domain.
- 5. If the handshake fails, reconcile IKE/IPSEC parameters with GRA before retrying.
- 6. Escalate to the GRA / Sentinal networking contact with your tunnel logs and the timestamp of the failure.

Appendix A — Endpoint Summary

Endpoint	Auth
POST /v1/psp/token	RSA signature — LOWERCASE HEX (Section 3.3)
POST /v1/client/registerClient	HMAC (no merchant key)
POST /v1/client/statusChanges	HMAC (no merchant key) — Section 5.3
POST /v1/transaction/verifyTransaction	HMAC + X-MERCHANT-KEY for PURCHASE
POST /v1/transaction/verifyTransactionStatus	HMAC + X-MERCHANT-KEY
POST /v1/transaction/creditInformation	HMAC (no merchant key)
POST /v1/transaction/declineTransaction (+ /decline, deprecated)	HMAC + X-MERCHANT-KEY
POST /v1/transaction/chargeback	HMAC + X-MERCHANT-KEY. Not /chargeBack — see Section 6.5.
POST /v1/transaction/refund	HMAC + X-MERCHANT-KEY
GET /v1/psp/vatRules	HMAC (no merchant key)
POST /v1/rates	HMAC (no merchant key) — Section 6.8
POST /v1/webhook/subscribe	HMAC (no merchant key) — Section 8.6
GET|PUT|DELETE /v1/webhook/subscriptions[/{id}]	GET: token only. PUT / DELETE: HMAC. Section 8.6.
POST /v1/webhook/subscriptions/{id}/toggle | /rotate-secret	HMAC (no merchant key) — Section 8.6
POST /v1/webhook/test	HMAC (no merchant key) — Section 8.6
Webhook events (transaction / decline / chargeback / refund / merchant.vat_updated)	HMAC-signed delivery to Payment Processor

Every path is absolute and begins with /v1

The paths above are complete request paths, appended to the environment base URL in Appendix B and nothing else. There is no additional /api or /partner prefix. This matters beyond routing: the HMAC string to sign binds the request path exactly as transmitted (Section 3.4), so a client that signs "/transaction/verifyTransaction" while sending "/v1/transaction/verifyTransaction" produces a signature that cannot match, and receives a 401 that looks like a secret problem.

Endpoints outside this list

The platform exposes further Payment-Processor-authenticated endpoints that are NOT part of the Ghana integration and are deliberately unspecified here — among them cryptoDeposit, cryptoWithdrawal and ntpn, which are inherited from another deployment. They are reachable with your credentials but carry no Ghana VAT semantics. Do not build against them.

POST /v1/transaction/chargeBack, with a capital B, is likewise not the chargeback endpoint of Section 6.5 — it is a separate evidence-upload path. See the warning in Section 6.5.

Not an endpoint

The merchant pre-registration export (Section 5.2.4) is a one-off onboarding aid provided on request, not an API endpoint.

Appendix B — Environments & Support

All endpoints in this document are relative to the environment base URL below. The Sentinal environments are reachable only over an IPSEC / site-to-site VPN to the GRA network — they are not exposed on the public internet.

Environment	Component	URL / Address
UAT	API (Backend)	https://uat-sentinal-api.gra.gov.gh
UAT	Dashboard (Front End)	https://uat-ecom.gra.gov.gh
UAT	IP Address	10.65.1.5
Production	API (Backend)	https://prod-sentinal-api.gra.gov.gh
Production	Dashboard (Front End)	https://ecom.gra.gov.gh
Production	IP Address	Not configured yet

Production not yet available

As of July 2026 (version 1.5.8), the Production environment is not yet implemented. The Production URLs are listed for reference only and are not live. Integrators will be notified separately once Production is available and its IP address is configured; integrate against UAT in the meantime.

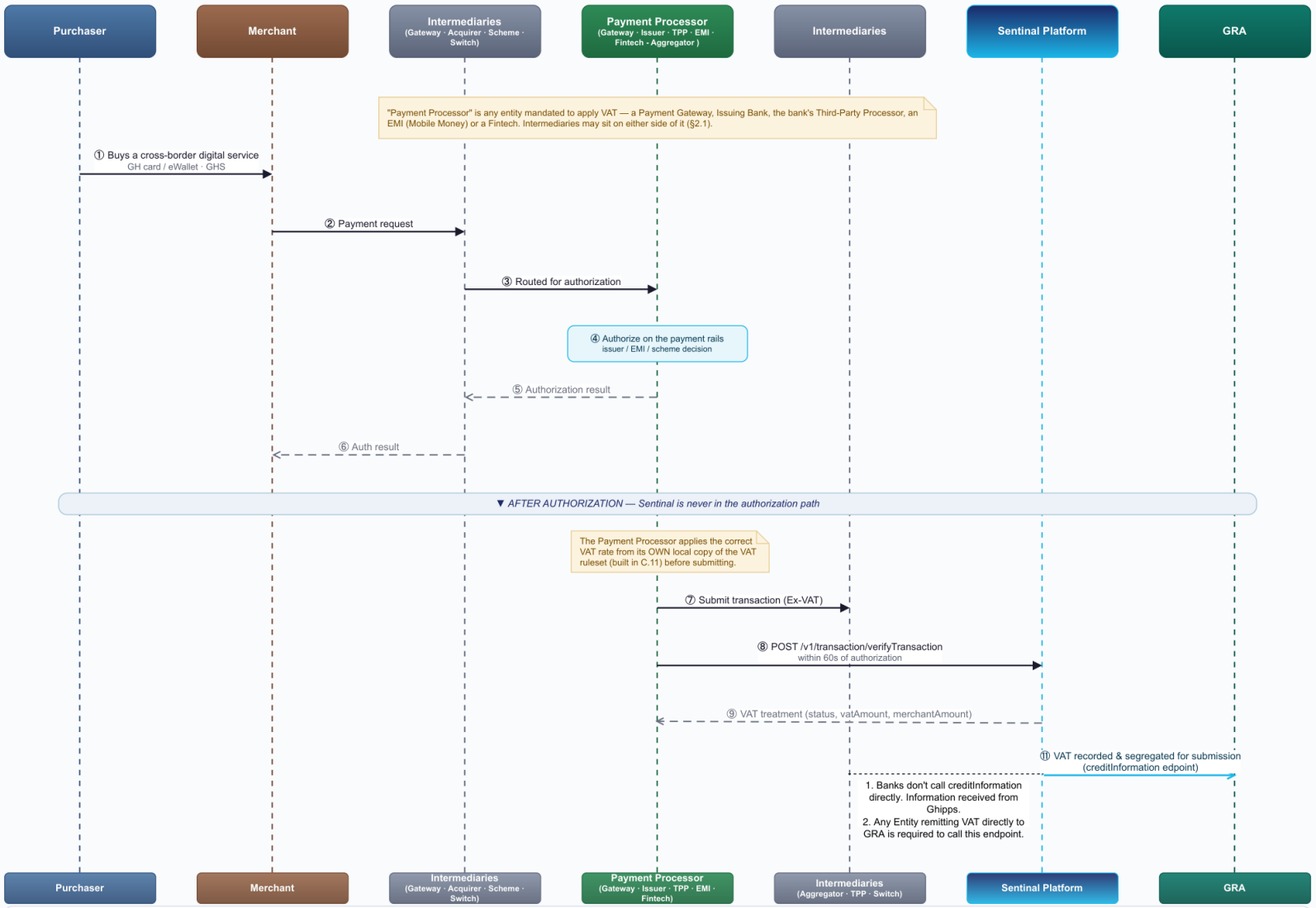
Network access

Access is restricted to IPSEC / site-to-site VPN only. There is no public-internet access to these URLs or IP addresses. Establish and verify your VPN tunnel to the GRA network before attempting to reach the API or dashboard (see Section 11.5 for VPN troubleshooting).

For integration support, contact your Sentinal technical account contact. When reporting an issue, include the endpoint, the X-TIMESTAMP used, the request ref, and the response message (never include secrets or private keys).

C.1 End-to-End Context

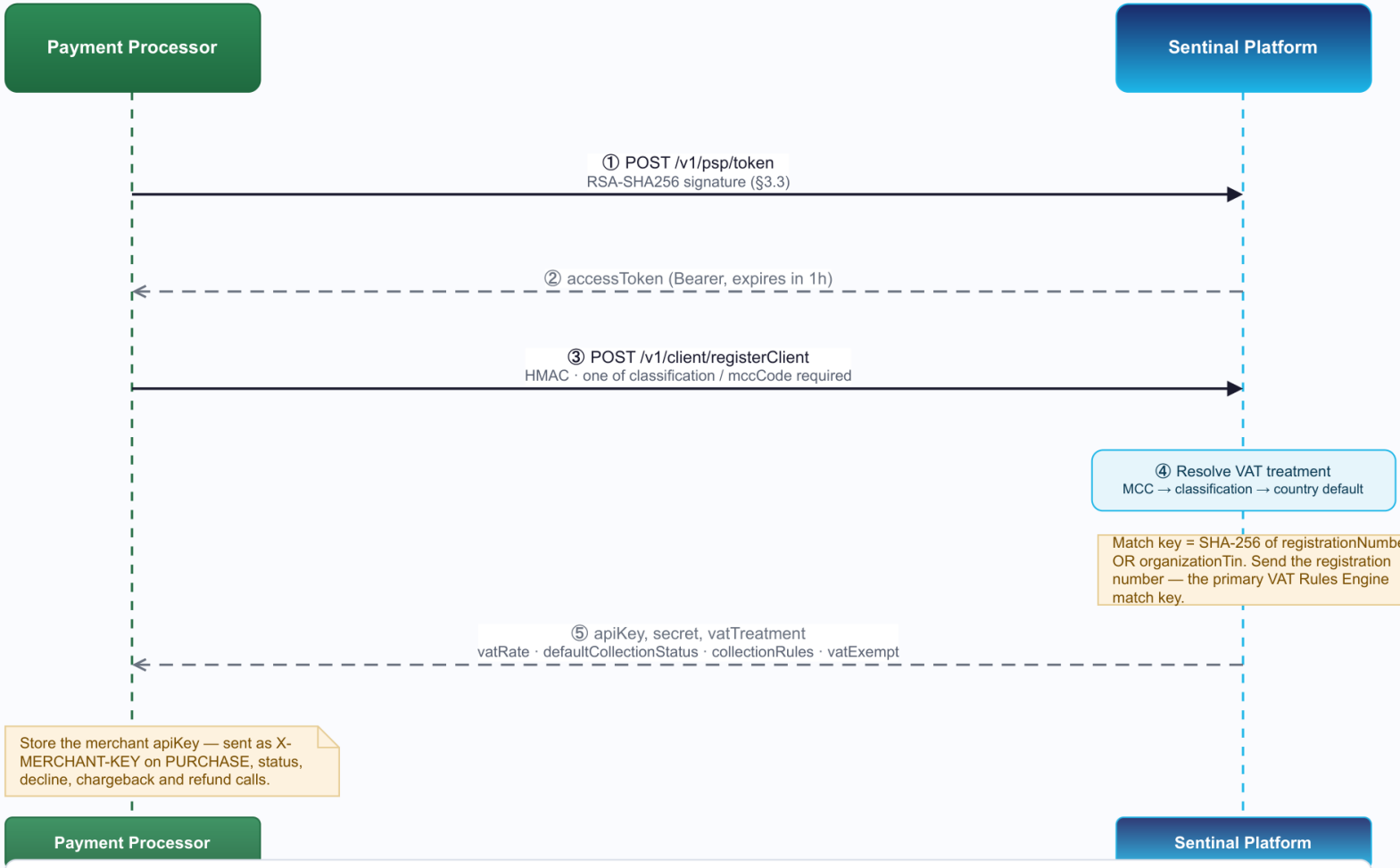
The Payment Processor — any mandated entity in the rail — records VAT with Sentinal after authorization



Sentinal Functional API Specification v1.5.6 — Appendix C · §1.1, §2.1, §2.3, §6.1. Sentinal sits OUTSIDE the authorization flow and never authorises, declines, delays or blocks settlement.

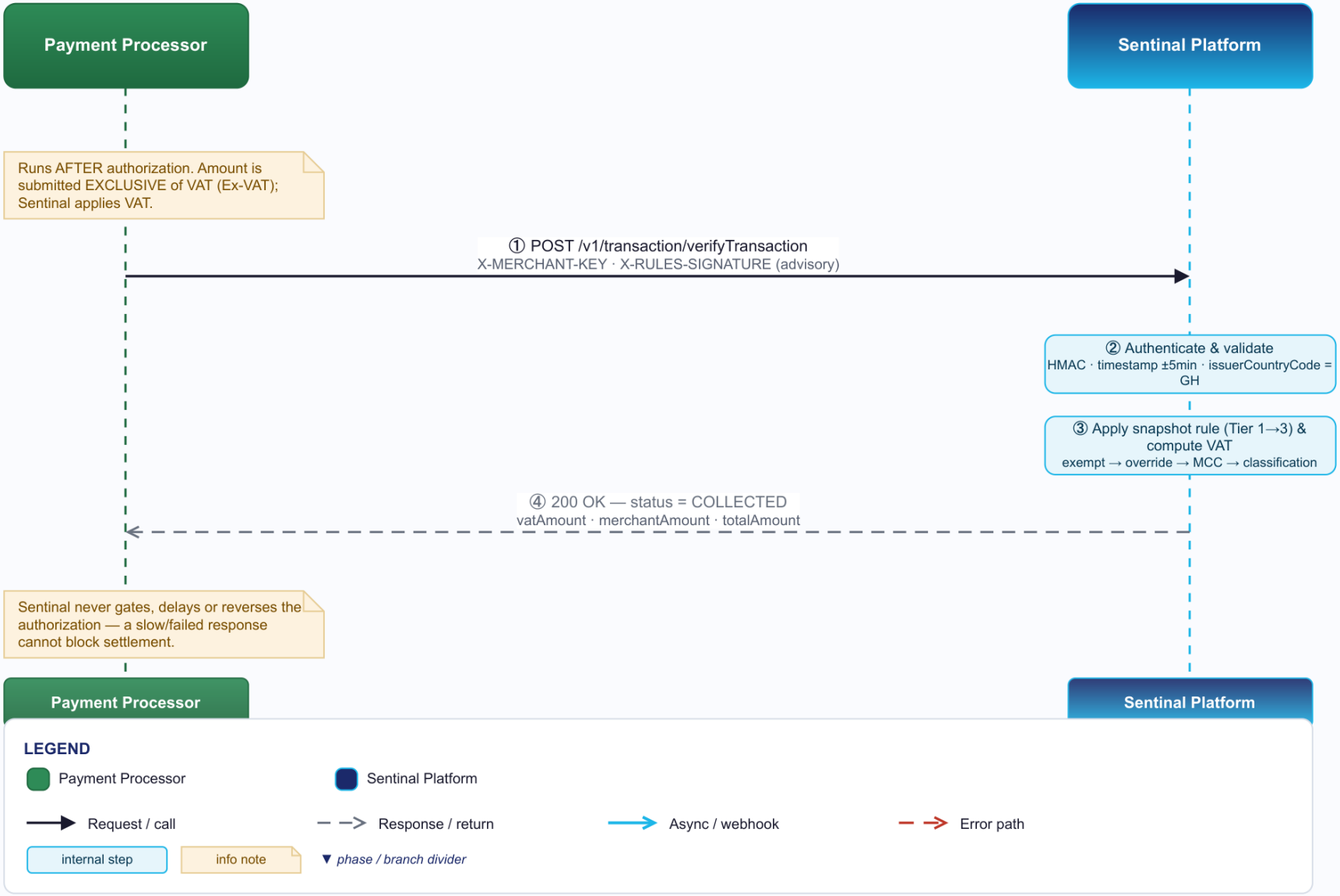
C.2 Merchant Registration

Obtain a token, register a merchant, and receive credentials + resolved VAT treatment

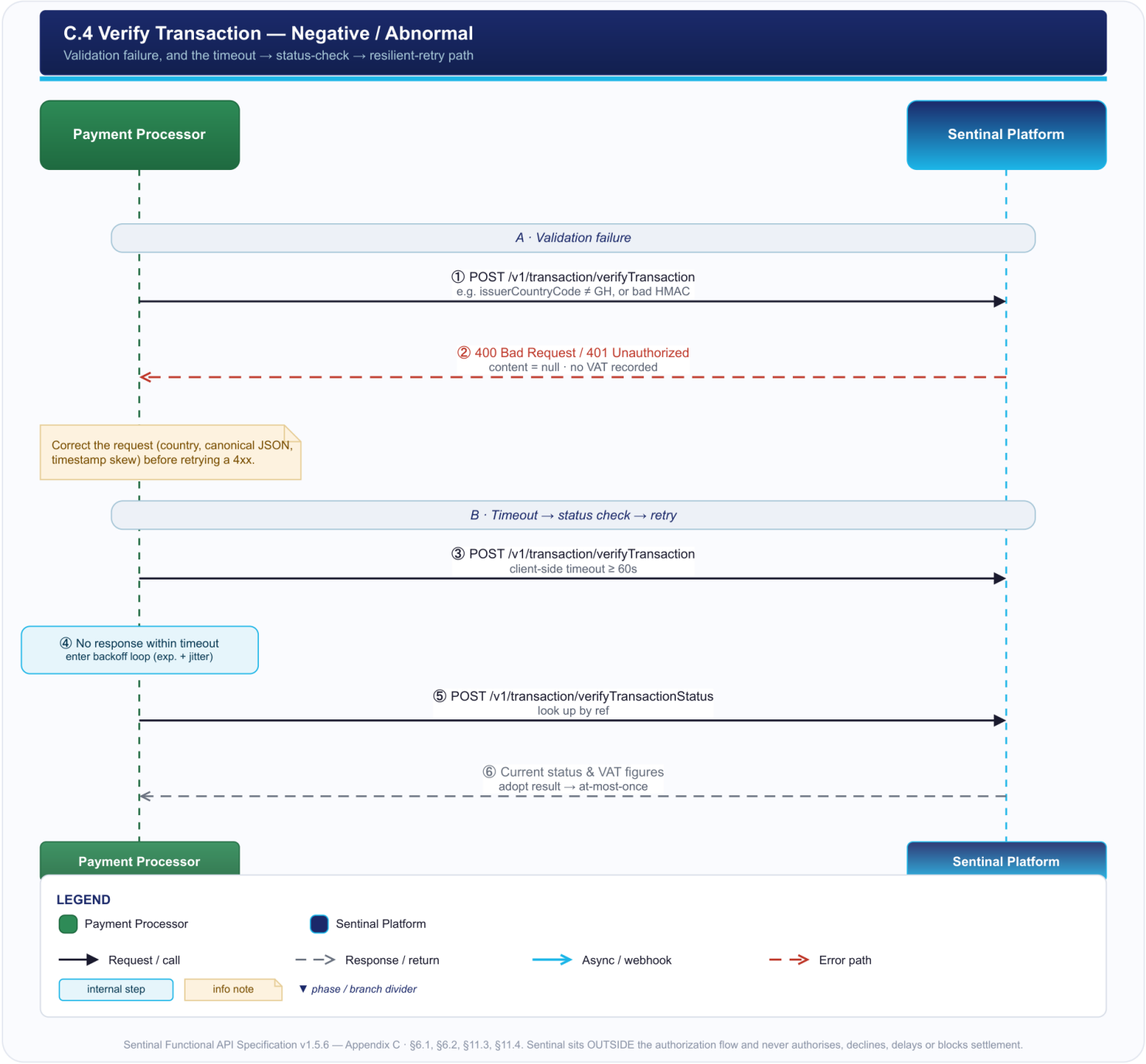


C.3 Verify Transaction — Positive (PURCHASE)

Successful verification and VAT determination for an already-authorised PURCHASE



Sentinal Functional API Specification v1.5.6 — Appendix C · §3.4, §6.1, §7.3. Sentinal sits OUTSIDE the authorization flow and never authorises, declines, delays or blocks settlement.



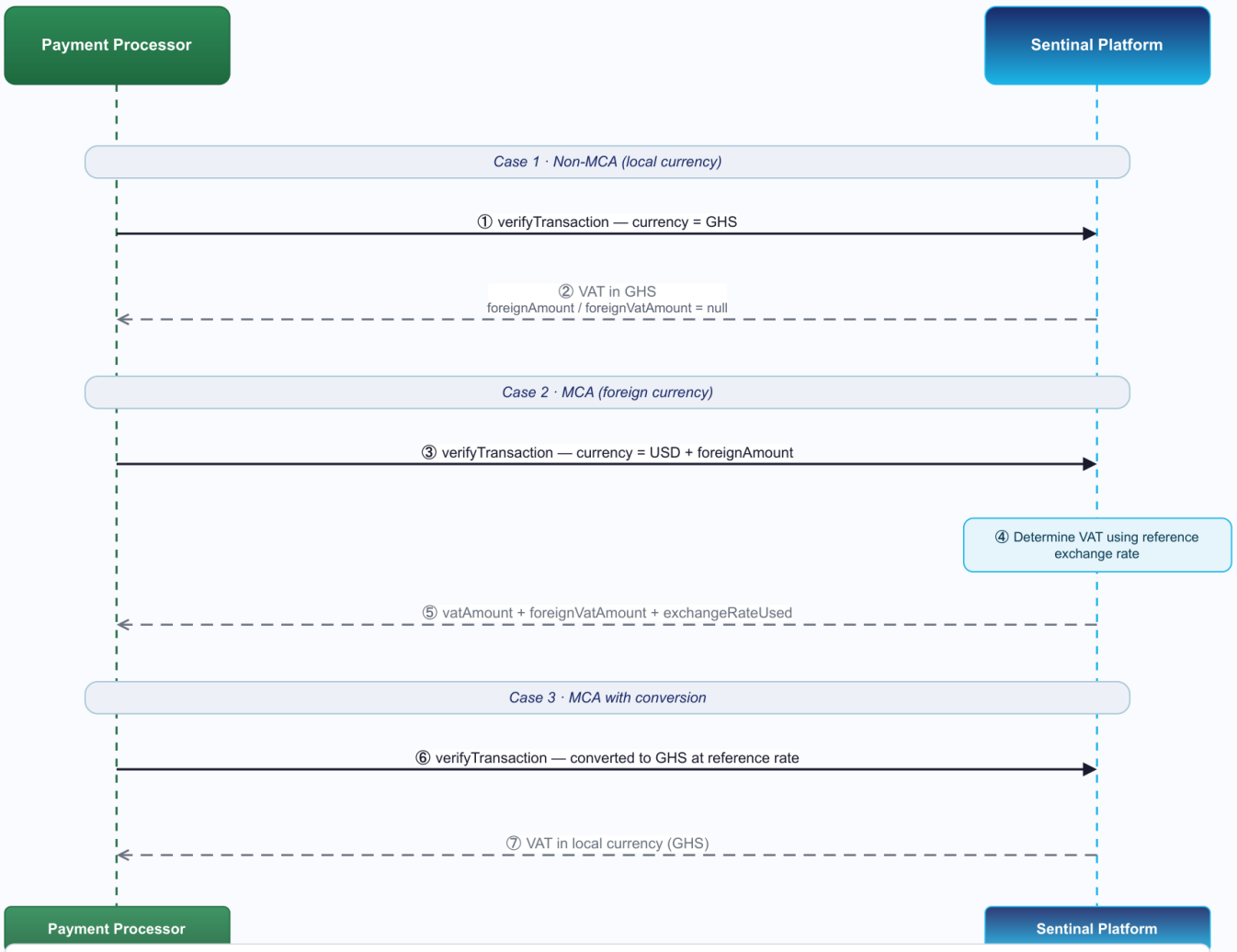
Sentinal Functional API Specification v1.5.6 — Appendix C · §6.1, §6.2, §11.3, §11.4. Sentinal sits OUTSIDE the authorization flow and never authorises, declines, delays or blocks settlement.

C.5 Non-PURCHASE (Participant) Transaction

Verification for non-PURCHASE transactions using participant identifiers (no merchant key)

C.6 Multi-Currency (MCA) Verification

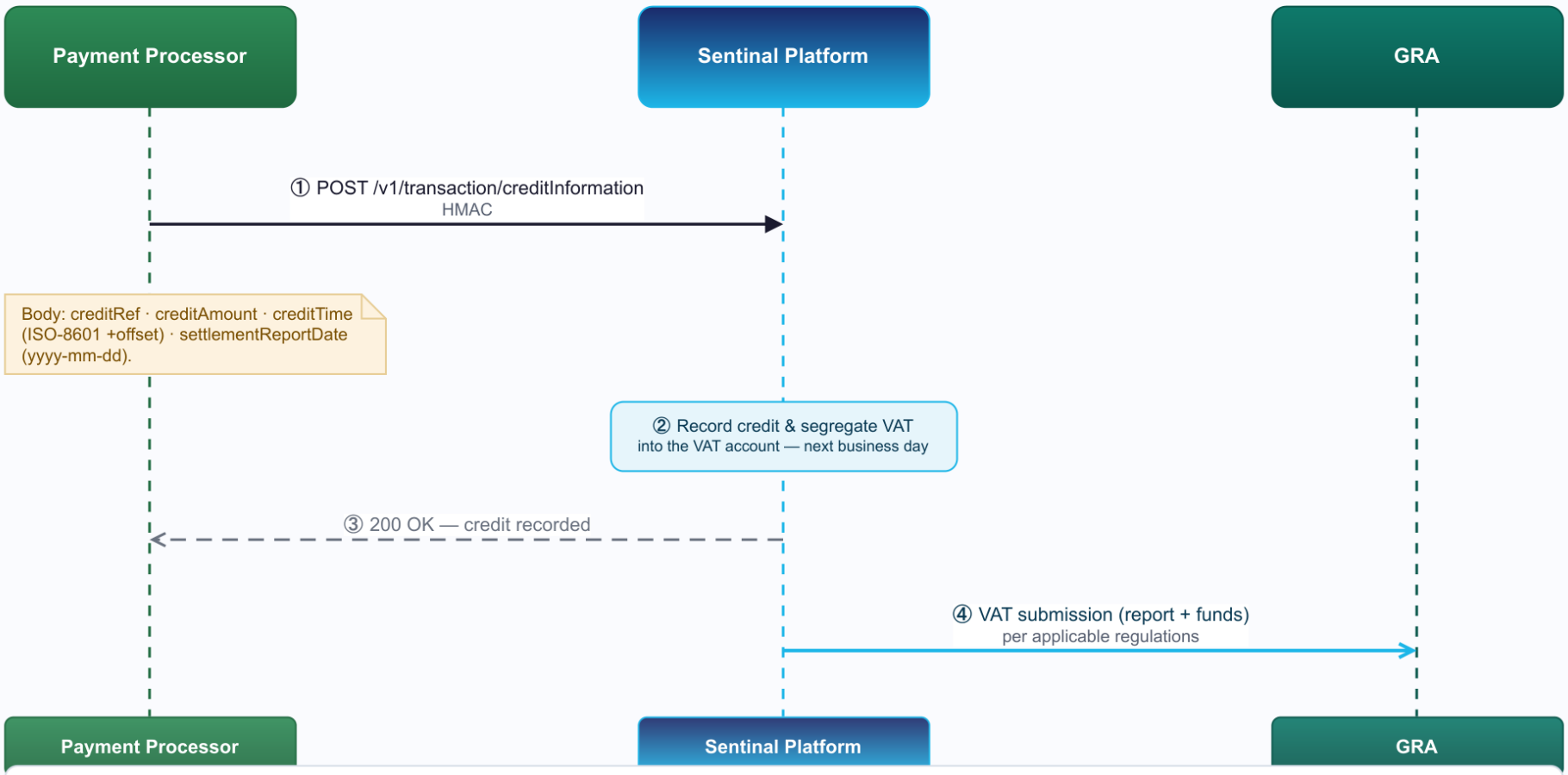
Three currency scenarios: non-MCA, MCA, and MCA-with-conversion



Sentinal Functional API Specification v1.5.6 — Appendix C · §6.1 (MCA). Sentinal sits OUTSIDE the authorization flow and never authorises, declines, delays or blocks settlement.

C.7 Successful Credit Information

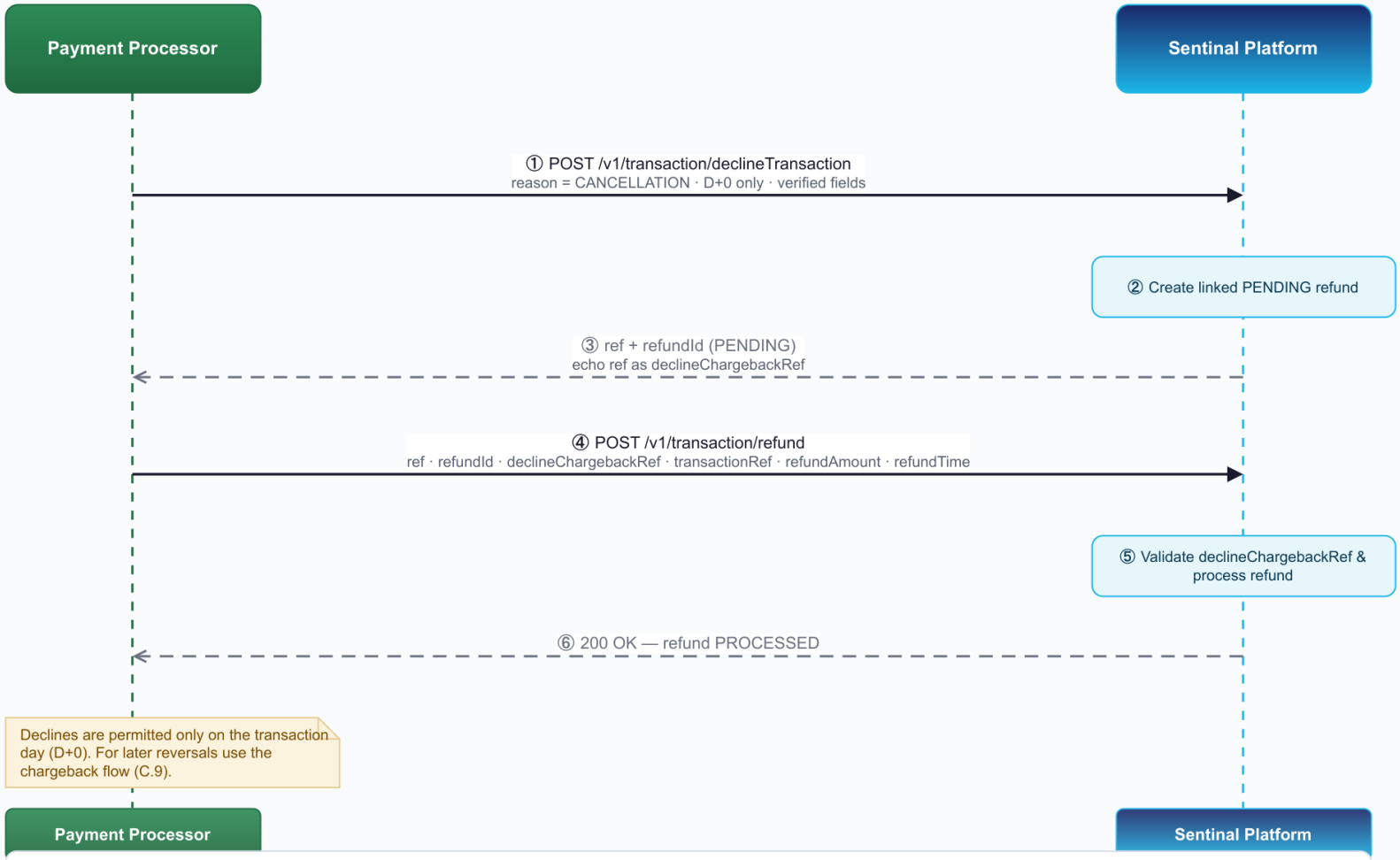
Payment Processor notifies Sentinal that VAT for a settlement has been credited



Sentinal Functional API Specification v1.5.6 — Appendix C · §3.4, §6.3. Sentinal sits OUTSIDE the authorization flow and never authorises, declines, delays or blocks settlement.

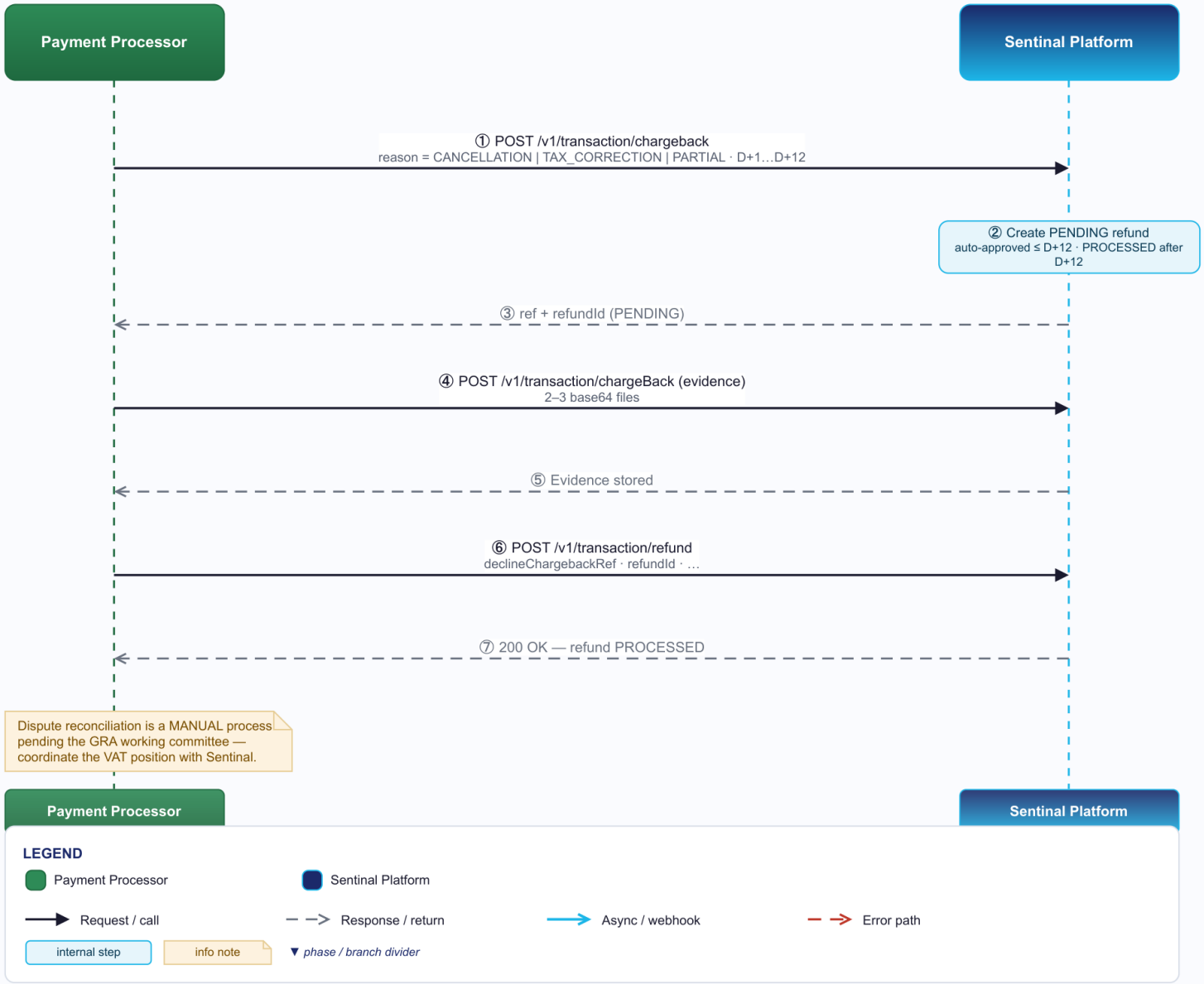
C.8 Decline → Refund (Same Day, D+0)

Same-day decline creates a linked PENDING refund, then the refund is processed



C.9 Chargeback (D+1 to D+12)

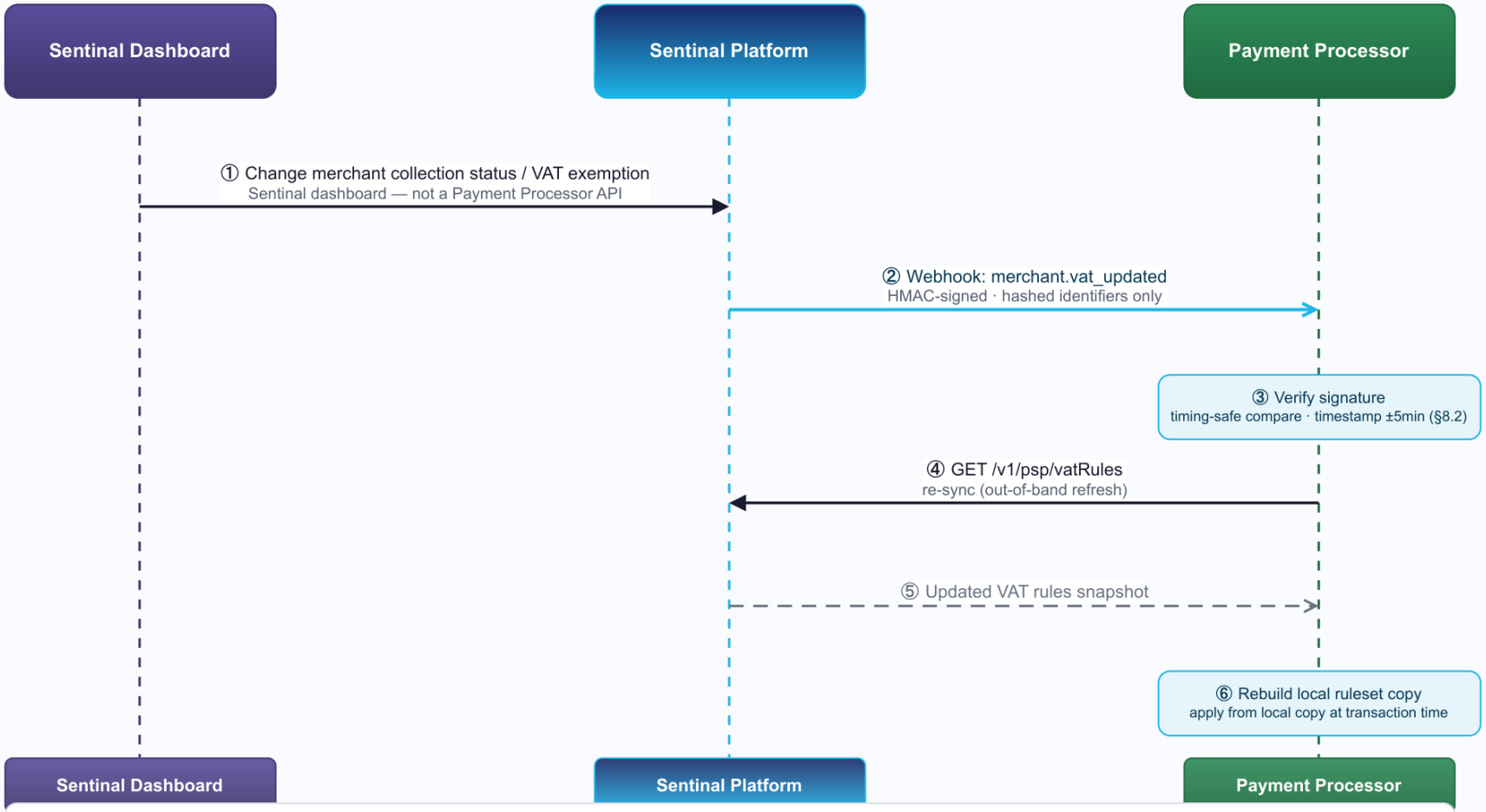
Chargeback record, evidence submission, resolution and refund



Sentinal Functional API Specification v1.5.6 — Appendix C · §6.5, §6.6, §10.3. Sentinal sits OUTSIDE the authorization flow and never authorises, declines, delays or blocks settlement.

C.10 Merchant Status Change

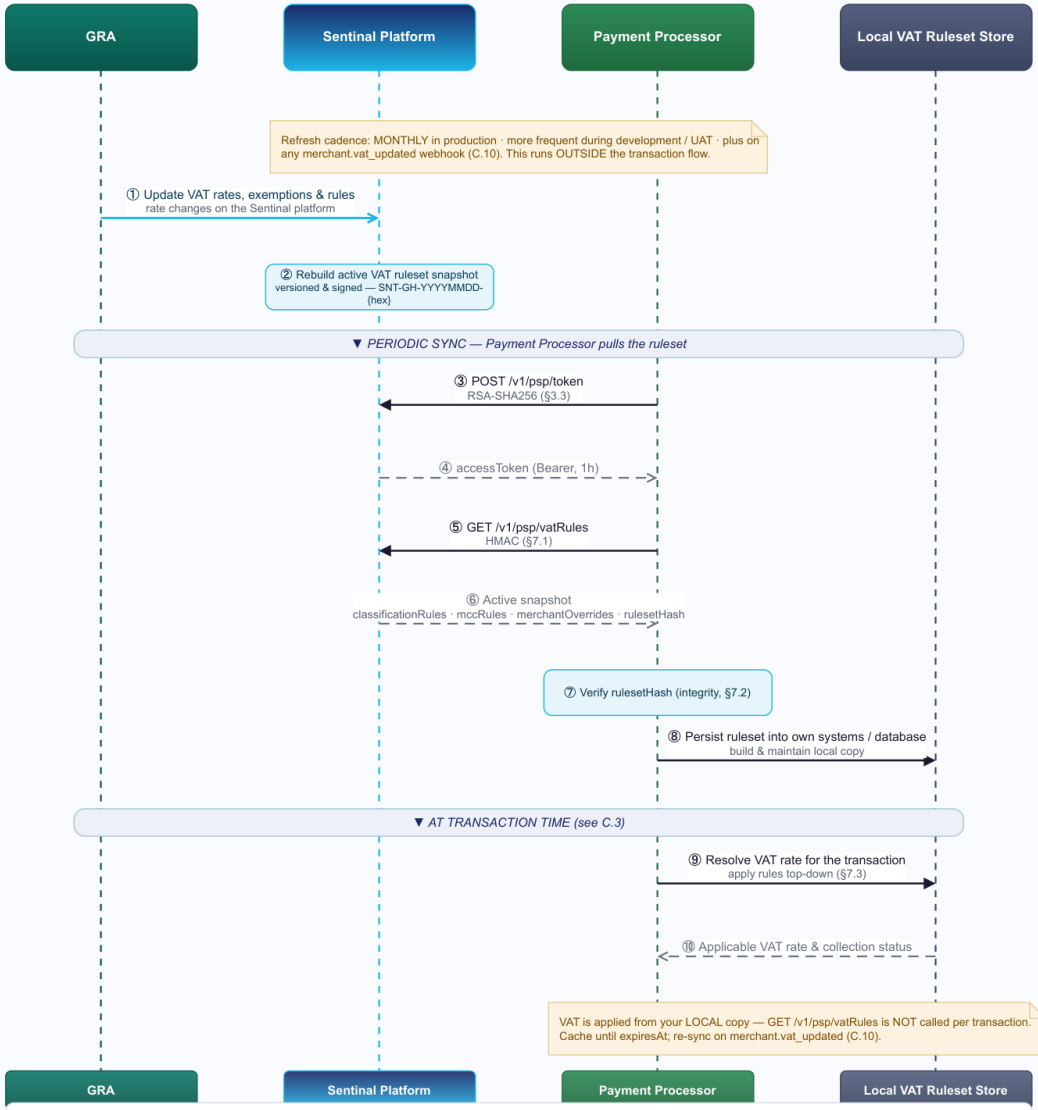
Dashboard status change → signed webhook → local VAT ruleset re-sync



Sentinal Functional API Specification v1.5.6 — Appendix C · §7.4, §8.4, §10.4. Sentinal sits OUTSIDE the authorization flow and never authorises, declines, delays or blocks settlement.

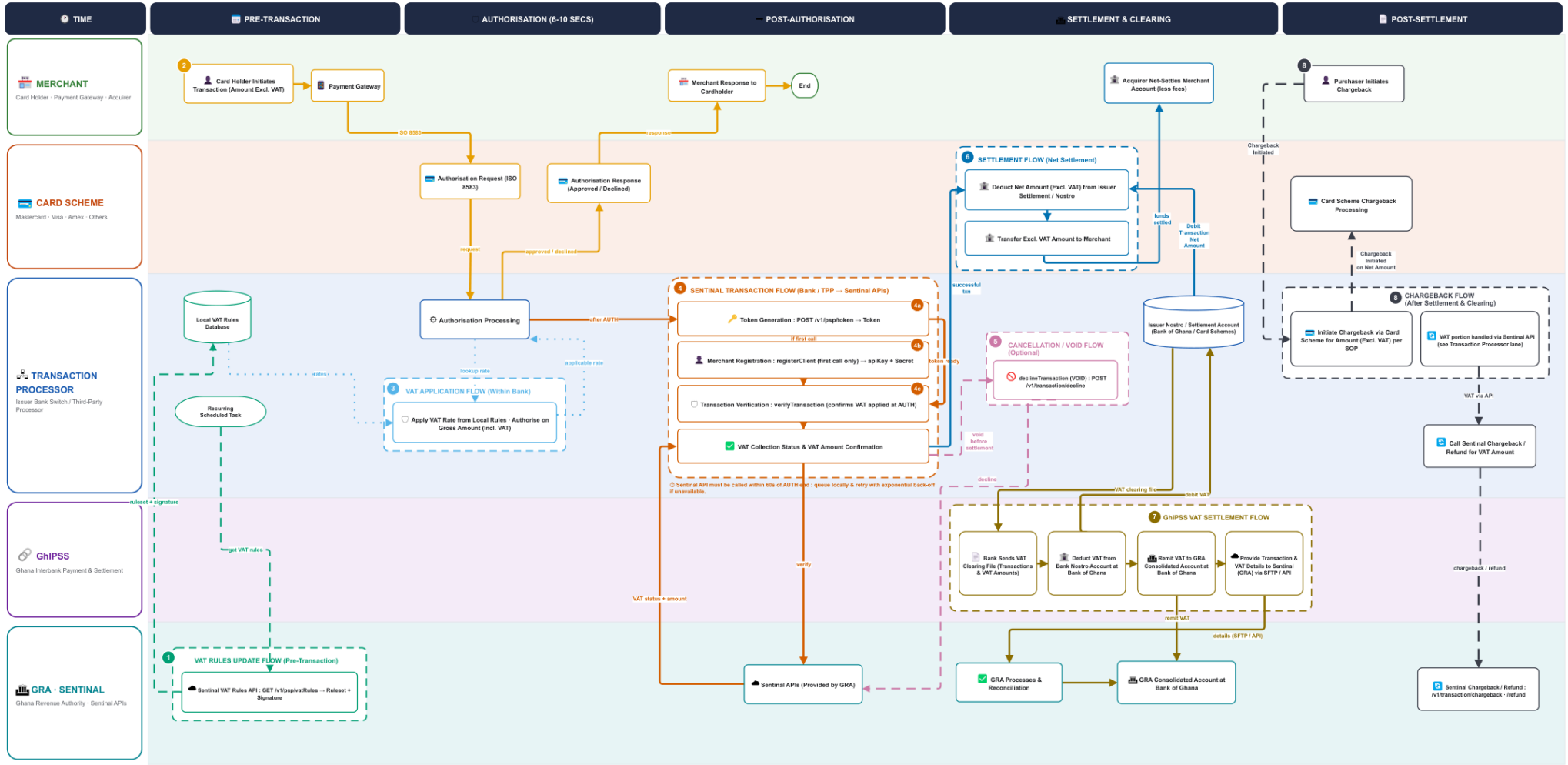
C.11 VAT Rates Collection (Periodic Sync)

The Payment Processor pulls the VAT ruleset out-of-band and applies it locally at transaction time



Sentinal Functional API Specification v1.5.6 — Appendix C · §7, §7.1, §7.2, §7.4. Sentinal sits OUTSIDE the authorization flow and never authorises, declines, delays or blocks settlement.

VAT Collection & Remittance - End-to-End Process (Sentinal · Ghana / GRA)



FLOW LEGEND

- VAT Rules Update Flow (Pre-Transaction)
- Card Scheme Authorisation Flow (ISO 8583)
- VAT Application Flow (Within Bank)
- Sentinal Transaction Flow (API Sequence)
- Cancellation / Void Flow (Optional)
- Settlement Flow (Net Settlement)
- GhiPSS VAT Settlement Flow
- Chargeback Flow (Post-Settlement)

KEY

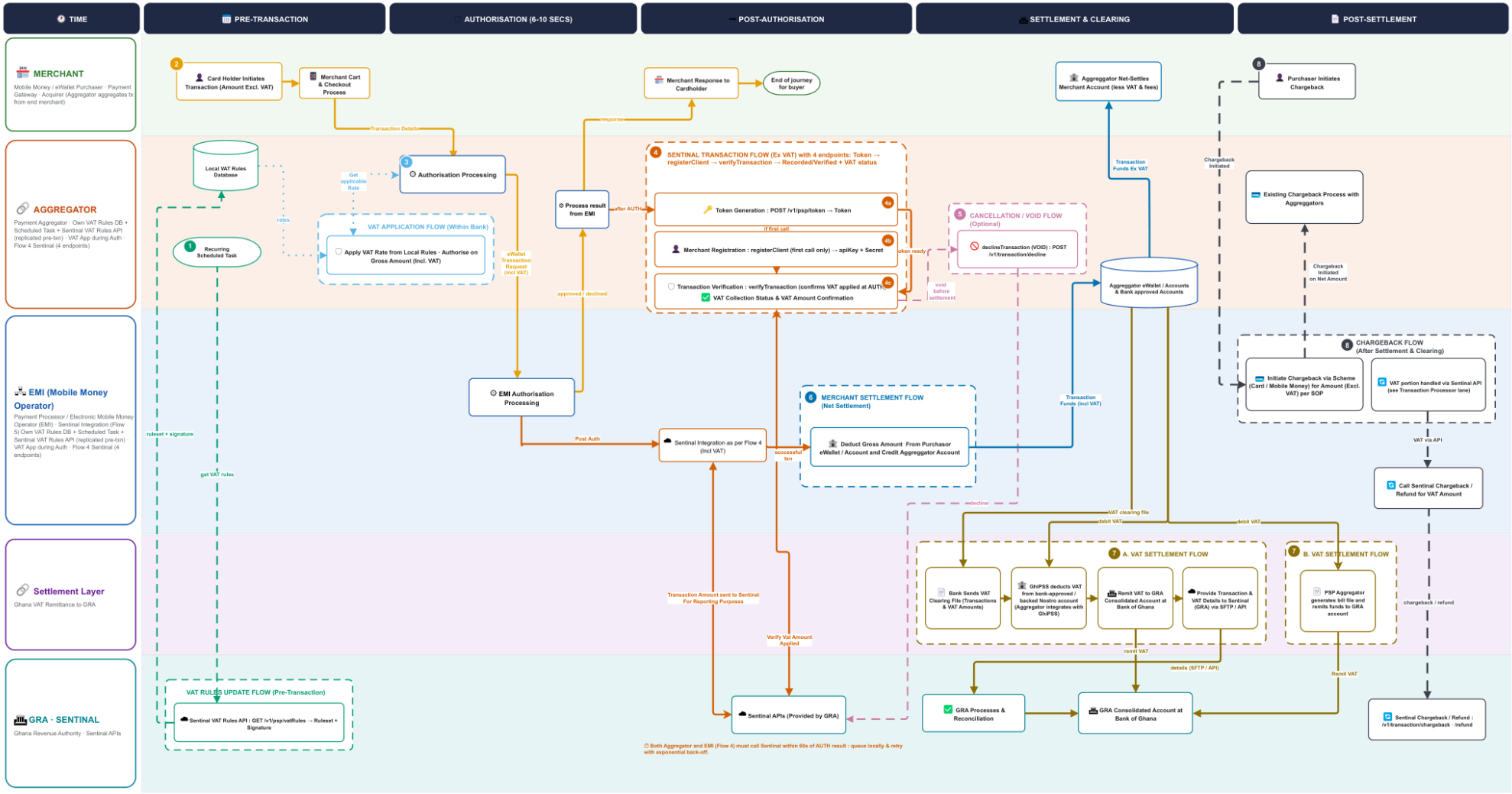
- Card Holder
- Merchant
- Payment Gateway
- Card Scheme
- File / Data
- Transaction Processor / Bank
- GhiPSS
- GRA / Sentinal
- Bank / Account
- Token
- Verification / Status
- Process
- API / File Transfer

NOTES

- Authorisation must complete within 6-10 seconds per Card Scheme rules.
- VAT is applied during AUTH using the bank's local VAT Rules; auth runs on the gross (Incl. VAT) amount.
- Sentinal API order: Token → registerClient (first call) → verifyTransaction; each step needs the prior credential.
- verifyTransaction confirms the VAT amount the bank applied; VAT is remitted to GRA via GhiPSS.
- Chargebacks are handled separately: Excl. VAT via the Card Scheme, VAT via the Sentinal API.
- Base URL : Sandbox: tba · Production: tba

Sentinal Phase 2 · Banks & TPPs · Issuer-Led VAT Collection | Revision 10 (Solution Architecture) | Version 2.0 | 2026-06-26

VAT Collection & Remittance - End-to-End (Aggregator VAT + EMI/eWallet Settlement) — Mobile Money & eWallet Supported



FLOW LEGEND

- VAT Rules Update Flow (Pre-Transaction)
- Scheme Authorisation Flow (ISO 8583 / Mobile Money)
- VAT Application Flow (Within Bank)
- Sentinal Transaction Flow (API Sequence)
- Cancellation / Void Flow (Optional)
- Settlement Flow (Net Settlement)
- GhiPSS VAT Settlement Flow
- Chargeback Flow (Post-Settlement)

KEY

- Card Holder
- Merchant
- Payment Gateway
- Scheme (Card / Mobile Money)
- File / Data
- Transaction Processor / Bank
- GhiPSS
- GRA / Sentinal
- Bank / Account
- Token
- Verification / Status
- Process
- API / File Transfer

NOTES

- Authorisation must complete within 6-10 seconds per Scheme (Card / Mobile Money) rules.
- VAT is applied during AUTH using the bank's local VAT Rules; auth runs on the gross (Incl. VAT) amount.
- Sentinal API order: Token → registerClient (first call) → verifyTransaction; each step needs the prior credential.
- Flow 4 (AGGREGATOR lane): VAT applied here + Sentinal integration. Flow 5 (BANK/TPP/EMI lane): replicated Sentinal integration post-AUTH. Banks/EMIs/TPPs register the tx as originating from the Aggregator (not end merchant) because of aggregation.
- Chargebacks: Excl. VAT via Scheme (Card / Mobile Money); VAT via Sentinal API (supported for Banks, TPPs and Aggregators).
- Base URL : Sandbox: tba · Production: tba

Illustrative structure only — the values in this sample are NOT real

This appendix exists to show the SHAPE of the GET /v1/psp/vatRules response: which fields appear, how the three tiers nest, which fields are optional, and the units each value carries. Build and test your parser against it.

Every value is invented. The signature, hashes, dates, MCC groupings, rates and collection statuses are placeholders chosen to exercise the structure, and several deliberately differ from the real Ghana configuration so this page cannot be mistaken for it. Do NOT copy any rate, status or MCC list from this page into a rules engine, and do not use it to reason about what VAT any real merchant attracts.

The authoritative ruleset is the live snapshot returned to you by the endpoint. The Ghana configuration as communicated is summarised in Section 9, and the snapshot overrides even that.

```
200 OK
{
  "success": true,
  "message": "VAT rules snapshot retrieved.",
  "content": {
    "signature": "SNT-GH-20260701-A1B2C3D4E5F60718",
    "generatedAt": "2026-07-01T00:05:00.000Z",
    "expiresAt": "2026-08-01T00:01:00.000Z",
    "countryCode": "GH",
    "defaultVatRate": 20,
    "localMerchantVatExempt": true,
    "rulesetHash": "9f2c4b7e1a8d05c36be94f27a0d1358e6c47b902fda85e13c760b4a29e8f5d16",

    "classificationRules": [
      {
        "classification": "SAMPLE_DIGITAL_SERVICES",
        "vatRate": 20,
        "defaultCollectionStatus": "COLLECTED"
      },
      {
        "classification": "SAMPLE_PLATFORM_HYBRID",
        "vatRate": 20,
        "defaultCollectionStatus": "POTENTIALLY_COLLECTED",
        "collectionRules": [
          { "paymentMethods": ["WALLET"], "collectionStatus": "NOT_COLLECTED" },
          { "paymentMethods": ["CREDIT_CARD", "DEBIT_CARD"], "collectionStatus": "COLLECTED" }
        ]
      },
      {
```

```
      "classification": "SAMPLE_LOCAL_SERVICES",
      "vatRate": 0,
      "defaultCollectionStatus": "NOT_COLLECTED"
    }
  ],

  "mccRules": [
    {
      "mccCodes": ["1111", "2222"],
      "classification": "SAMPLE_DIGITAL_SERVICES",
      "vatRate": 20,
      "defaultCollectionStatus": "COLLECTED"
    },
    {
      "mccCodes": ["3333"],
      "classification": "SAMPLE_PLATFORM_HYBRID",
      "vatRate": 20,
      "defaultCollectionStatus": "POTENTIALLY_COLLECTED",
      "collectionRules": [
        { "paymentMethods": ["WALLET"], "collectionStatus": "NOT_COLLECTED" }
      ]
    },
    {
      "mccCodes": ["4444", "5555", "6666"],
      "classification": "SAMPLE_LOCAL_SERVICES",
      "vatRate": 0,
      "defaultCollectionStatus": "NOT_COLLECTED"
    }
  ],

  "merchantOverrides": {
    "exemptMerchants": [
      {
        "registrationNumberHash": "3b1f8c25d4e70a96b8c1f2350e7d4a68c93b05e1f7a2d84c60b9e35172af8dc4",
        "organizationTinHash": "c74a09e3b8512df6470ac9b1e2385f0d69c4a71b3e8d520fa6c19b473e0d825a",
        "collectionStatus": "NOT_COLLECTED",
        "reason": "VAT_EXEMPT"
      },
      {
        "registrationNumberHash": "6d20b93af1c845e7302b9d16c58e4f07a3b921ce6540df8b17a2e935c04bd681",
        "collectionStatus": "NOT_COLLECTED",
        "reason": "KNOWN_MERCHANT"
```

```
        }
      ],
      "statusOverrides": [
        {
          "registrationNumberHash": "a48e7c02b913d65fa07c2e83b514d97016fa3c28e57b0d94612ca8f30e7b52d9",
          "overrideStatus": "COLLECTED",
          "vatRate": 20,
          "appliedMccCode": "3333",
          "mccDefaultStatus": "POTENTIALLY_COLLECTED"
        },
        {
          "organizationTinHash": "f05b1d7ae293c684017b5da9c3e28f4160b7d5c19a3ef806249bd0a7c1e35b42",
          "overrideStatus": "POTENTIALLY_COLLECTED",
          "vatRate": 20,
          "appliedClassification": "SAMPLE_DIGITAL_SERVICES",
          "classificationDefaultStatus": "COLLECTED"
        }
      ]
    }
  }
}
```

Points of structure this sample is intended to demonstrate:

- Rates are PERCENTAGES. 20 means 20% — see the warning in Section 7.1. A rate of 0 is a genuine zero-rating, not a missing value.
- collectionRules is OMITTED entirely on rules that have no payment-method overrides, rather than being returned as an empty array. Your parser must treat the key as absent, not empty.
- An exemptMerchants entry may carry registrationNumberHash, organizationTinHash, or both. The second entry above omits the TIN hash — absent, not null. Match on whichever hash you hold.
- A statusOverrides entry explains itself through EITHER appliedMccCode + mccDefaultStatus OR appliedClassification + classificationDefaultStatus, never both. Those fields are diagnostic context; overrideStatus is what you apply.
- The three tiers are evaluated in the order of Section 7.3 — exemptMerchants, then statusOverrides, then mccRules, then classificationRules — and the FIRST match stops evaluation.
- rulesetHash covers only the six fields named in Section 7.2; signature, generatedAt and expiresAt are outside it.