SENTINAL

Digital Tax Compliance Platform

FUNCTIONAL API SPECIFICATION

Payment Processor Integration Reference

Version 1.5.6 — July 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	1.5.6
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 July 2026 (version 1.5.6), 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.

Table of Contents

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, 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.

> **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-l_-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).

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, chargeBack (evidence), and refund; optional otherwise.
X-RULES-SIGNATURE	No	Active snapshot signature, echoed for audit linkage on verifyTransaction (advisory).
Content-Type	Yes	application/json

3.3 RSA Signature — Access Token

The token request is signed with the Payment Processor's RSA private key (SHA256withRSA, verified 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:

```
stringToSign = X-PARTNER-ID + "|" + X-TIMESTAMP
signature    = base64( RSA_SHA256_sign( pspPrivateKey, utf8Bytes(stringToSign) ) )
```

```
// place `signature` in the X-SIGNATURE header
```

Example string to sign (illustrative values):

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

> **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 as sent. accessToken is the bearer token without the "Bearer " prefix.

> **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
400	Invalid or expired timestamp	X-TIMESTAMP missing, malformed, or outside the 5-minute window.
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 against the registered public key.
401	Payment Processor public key not configured	No public key on file; register one before requesting tokens.

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.
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 decimal (e.g. 0.2 = 20%).
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.

VAT treatment resolution

Effective VAT treatment is resolved with the priority: registered MCC fee → classification (Type) fee → country default. The same resolution is reflected in the VAT rules snapshot (Section 7) for the merchant.

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 applies VAT (per Section 7) and returns the VAT and the VAT-inclusive total (transaction amount + VAT). Example: a GHS 100.00 Ex-VAT service is submitted as totalAmount 100.00; Sentinal returns vatAmount and the full amount.

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.

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).
foreignAmount	number	No	Foreign-currency amount (≥ 0).
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.
transactionFee	number	Cond.	≥ 0. Allowed only where the country enables fee reporting; defaults to 0.
mccCode	string	No	4-digit MCC (nullable).
issuingBank	string	Yes	Issuing bank code (max 8, nullable per deployment).
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.

Field	Type	Req.	Rules & Description
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 is rejected where the country configuration does not enable fee reporting.
- transactionTime must not be later than X-TIMESTAMP.

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.

Response — 200 OK (content)

Field	Type	Description
transactionId	string	Internal transaction identifier.
totalAmount	number	Verified total amount.
status	string	COLLECTED | NOT_COLLECTED | POTENTIALLY_COLLECTED.
vatAmount	number	Calculated VAT amount.

Field	Type	Description
merchantAmount	number	Amount net of VAT.
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 domestic).
auditFlagged	boolean	Whether the transaction was flagged for audit.
auditReason	string | null	Reason, when flagged.
penaltyApplied	boolean	Whether a late-submission penalty applies.
penaltyAmount	number	Penalty amount, when applicable.
remittanceSurcharge	boolean	Present only for REMITTANCE transactions.

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.

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)

Returns the transaction's current status and VAT figures (status, vatAmount, merchantAmount, and related fields, as in 6.1). A 400 is returned if neither identifier is supplied.

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 timestamp with timezone offset (e.g. 2026-07-09T11:59:00+00:00).
settlementReportDate	string	Yes	Settlement report date in yyyy-mm-dd format (e.g. 2026-07-09).

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 (optional; max 256). A value is generated if omitted.
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 (base64) with filename and contentType.

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 — process under review

The GRA is establishing a working committee to define the end-to-end process for chargebacks, refunds, and returns. Until that process is finalised, treat these as a MANUAL process and coordinate the VAT position with the GRA / Sentinal team. The decline, chargeback, and refund endpoints documented here are available, but dispute reconciliation is handled manually for now.

6.5 Chargebacks

POST /v1/transaction/chargeback records a chargeback and creates a PENDING refund; it is permitted from D+1, auto-approved within D+12 and set to PROCESSED after D+12. POST /v1/transaction/chargeBack submits chargeback evidence (2–3 files). The external ref field is optional on both.

6.6 POST /v1/transaction/refund

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

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).

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.

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. The response carries Cache-Control: max-age=3600; cache the snapshot until expiresAt.

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	Country default VAT rate (decimal).
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).
merchantOverrides.statusOverrides	array	Tier 3b — merchants deviating from their rule default.

classificationRules[] item

Field	Type	Description
classification	string	Merchant Type (Section 9).
vatRate	number	VAT rate as a decimal (e.g. 0.2).
defaultCollectionStatus	string?	COLLECTED | NOT_COLLECTED | POTENTIALLY_COLLECTED.
collectionRules	array?	Payment-method-specific overrides (present only when configured).

mccRules[] item

Field	Type	Description
mccCodes	string[]	MCC codes covered by this rule.
classification	string?	Associated Type, when applicable.
vatRate	number	VAT rate as a decimal.

Field	Type	Description
defaultCollectionStatus	string?	Collection status enumeration.
collectionRules	array?	Payment-method-specific overrides (present only when configured).

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. Cache the snapshot until expiresAt.

7.2 Integrity Verification (rulesetHash)

Payment Processors may verify snapshot integrity by recomputing the SHA-256 of the canonical JSON of the ruleset payload (object keys sorted alphabetically at every level; compact separators; no insignificant whitespace; the decimal rules in Section 3.5 apply) and comparing it with rulesetHash.

7.3 Applying the Snapshot (Resolution Order)

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. Only when no rule matches (you reach rule 5) do you use the country default.

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.

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 0.2, COLLECTED — classificationRules skipped.
No override; MCC not in any mccRule; merchant Type ECOMM_* 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.

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; a Payment Processor is never left without a servable ruleset. The expiresAt field is an advisory cache hint — the active snapshot remains valid until it is replaced. Poll GET /v1/psp/vatRules at or after expiresAt, and whenever you receive a merchant.vat_updated webhook, to obtain the latest ruleset.

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.

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.

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

8.3 Available Events

The event type appears in the X-EVENT-TYPE header. Subscribe only to the events relevant to your integration. All deliveries share the signing scheme in 8.2.

Domain	Event types (X-EVENT-TYPE)
Transaction	webhook.transaction.status, webhook.transaction.verified, webhook.transaction.failed, webhook.transaction.suspicious
Decline	webhook.decline.status, webhook.decline.created, webhook.decline.processing, webhook.decline.approved, webhook.decline.processed, webhook.decline.rejected
Chargeback	webhook.chargeback.status, webhook.chargeback.created, webhook.chargeback.requested, webhook.chargeback.approved, webhook.chargeback.rejected
Refund	webhook.refund.status, webhook.refund.requested, webhook.refund.approved, webhook.refund.processed, webhook.refund.rejected
Merchant (VAT)	merchant.vat_updated

The transaction, decline, chargeback, and refund events notify the Payment Processor as the corresponding lifecycle state changes (created / processing / approved / rejected / processed, with a rolling ...status event). The merchant.vat_updated event — the primary Ghana VAT event — is detailed below.

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).

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

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 are always 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 submits evidence (optional).	POST /v1/transaction/chargeBack (2–3 files)
3	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, provide a human-readable message, and set content to null; internal error details are never exposed to clients.

```
// 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.
401 Unauthorized	Authentication failed.	Bad/expired token, invalid RSA or HMAC signature, unknown partner.
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.
500 Server Error	Unexpected internal error.	Retry with backoff; contact support if persistent.

11.3 Common Errors

Symptom	Likely cause	Resolution
401 Invalid RSA signature (token)	Wrong string-to-sign or key mismatch.	Sign X-PARTNER-ID + "|" + X-TIMESTAMP; confirm the registered public key.
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.

Symptom	Likely cause	Resolution
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.

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.
- 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.
- 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	12.8 – 19.2 s	~30 s — ~300s (Cap at 5m)

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. 300 s / 5m).
• 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
POST /v1/client/registerClient	HMAC
POST /v1/transaction/verifyTransaction	HMAC
POST /v1/transaction/verifyTransactionStatus	HMAC
POST /v1/transaction/creditInformation	HMAC
POST /v1/transaction/declineTransaction (+ /decline, deprecated)	HMAC
POST /v1/transaction/chargeback	HMAC
POST /v1/transaction/chargeBack (evidence)	HMAC
POST /v1/transaction/refund	HMAC
GET /v1/psp/vatRules	HMAC
Webhook events (transaction / decline / chargeback / refund / merchant.vat_updated)	HMAC-signed delivery to Payment Processor

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 1.5.6), 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).

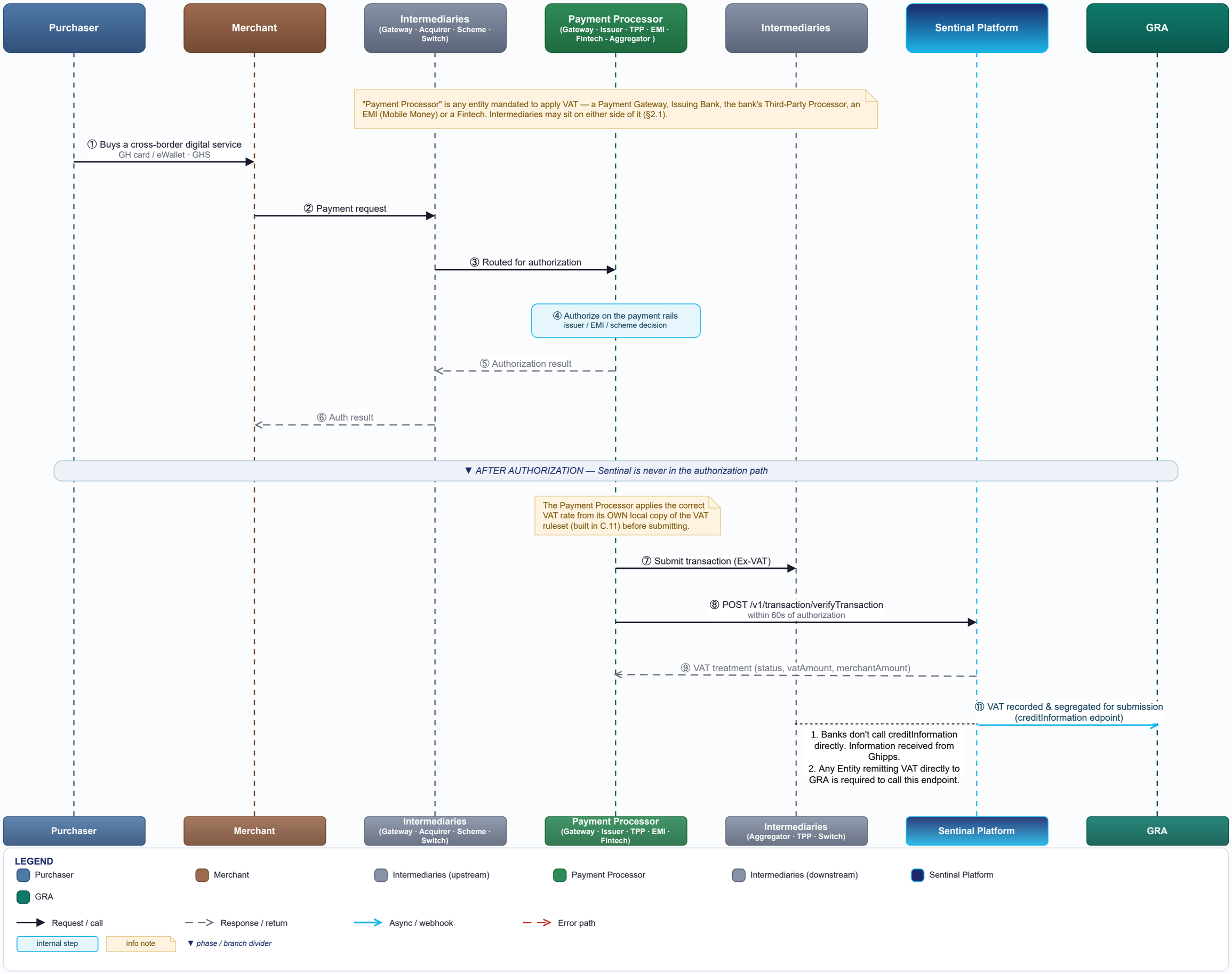
Appendix C — Flow Diagrams

The diagrams below illustrate the Sentinal integration flows. Insert the corresponding sequence diagrams in the marked areas.

<<Diagrams Start on the next page>>

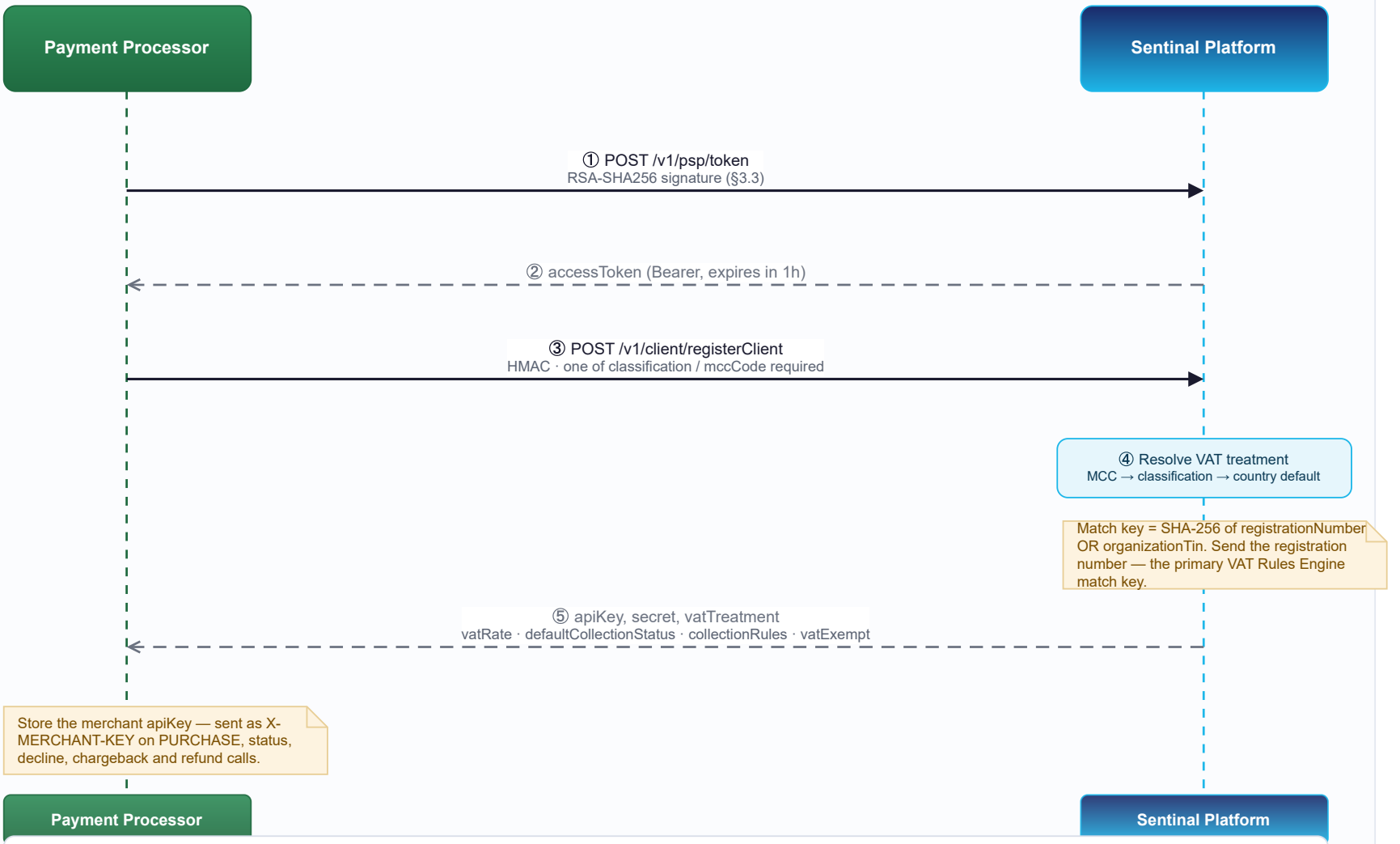
C.1 End-to-End Context

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



C.2 Merchant Registration

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

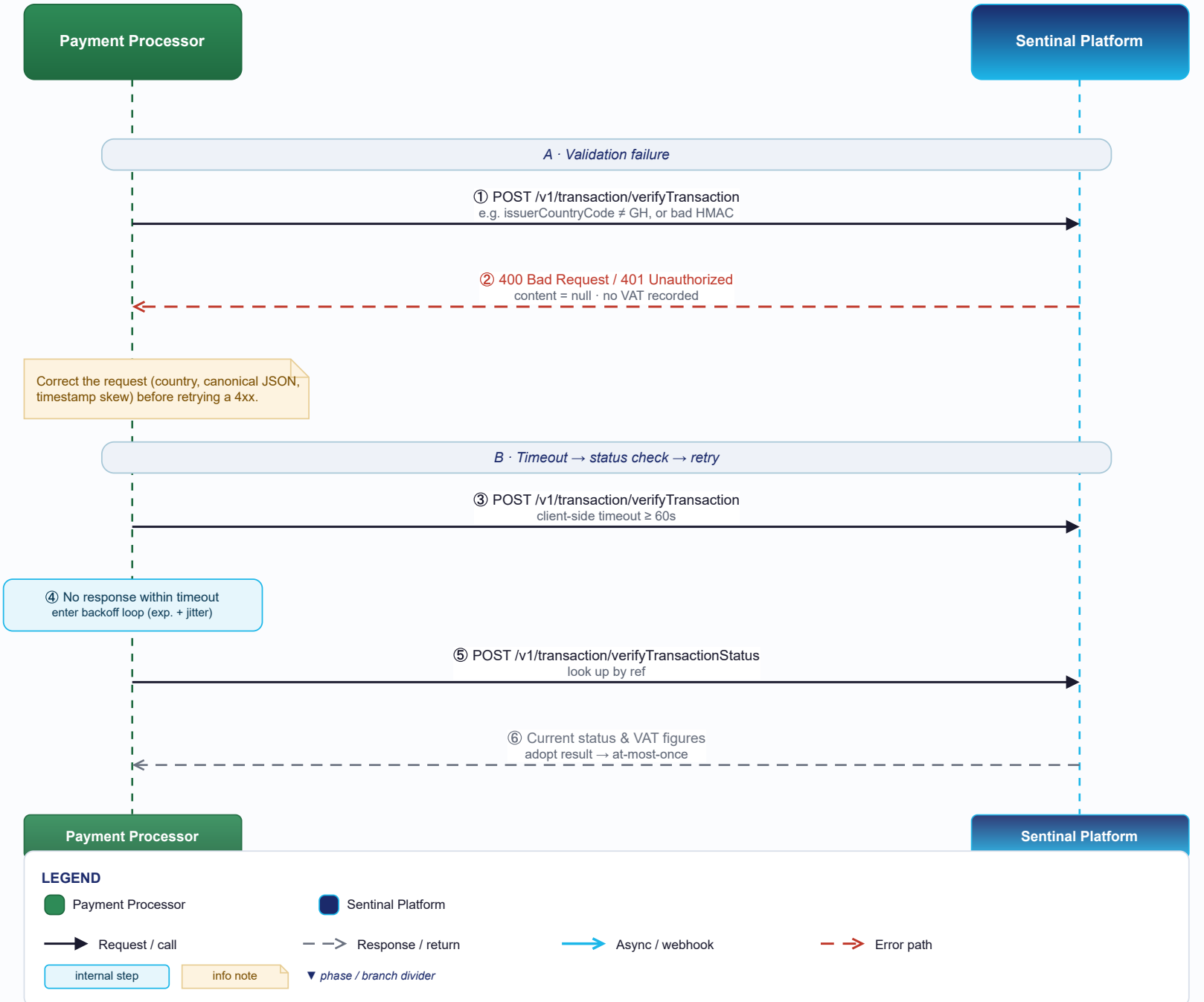


C.3 Verify Transaction — Positive (PURCHASE)

Successful verification and VAT determination for an already-authorised PURCHASE

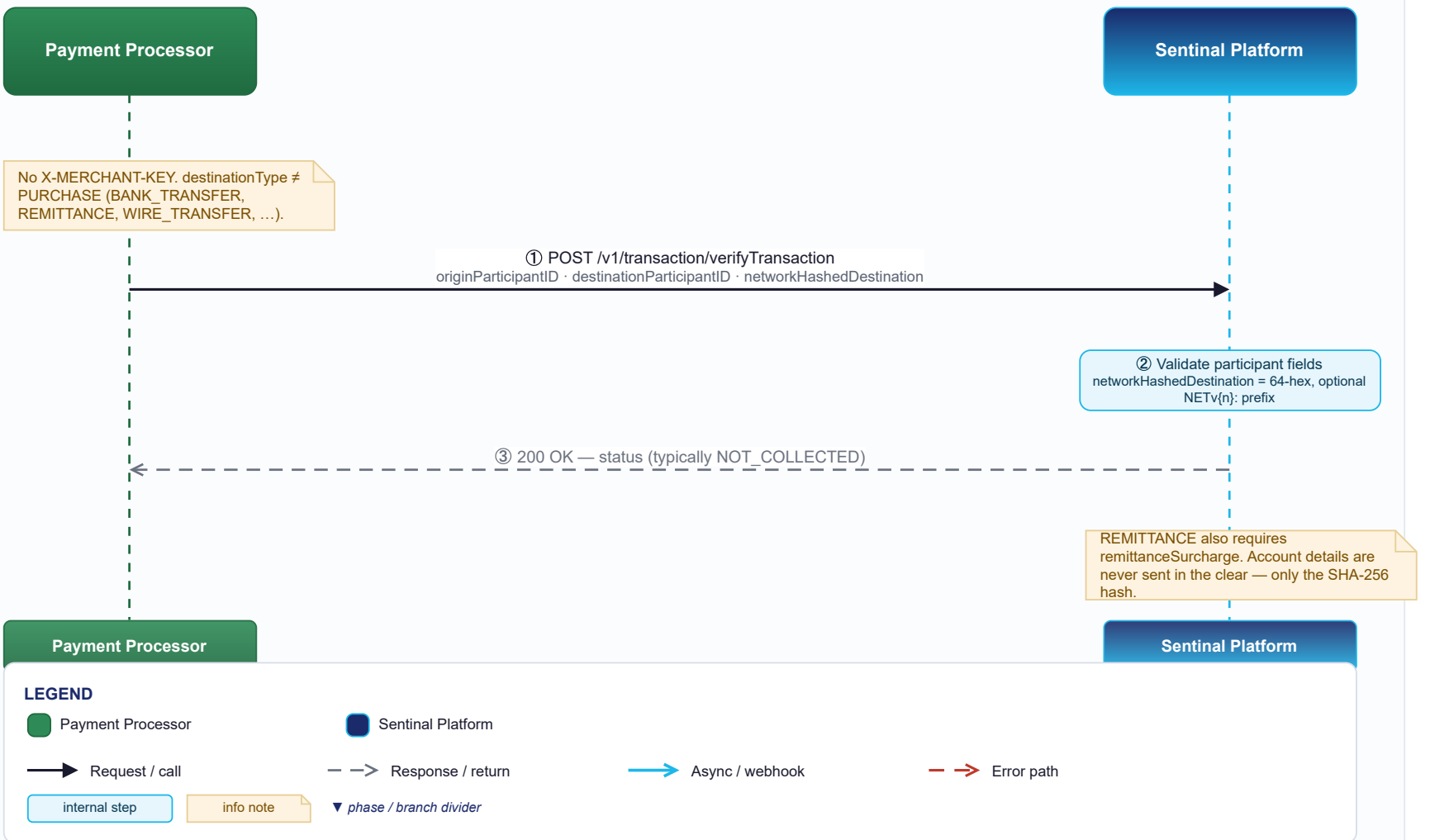
C.4 Verify Transaction — Negative / Abnormal

Validation failure, and the timeout → status-check → resilient-retry path



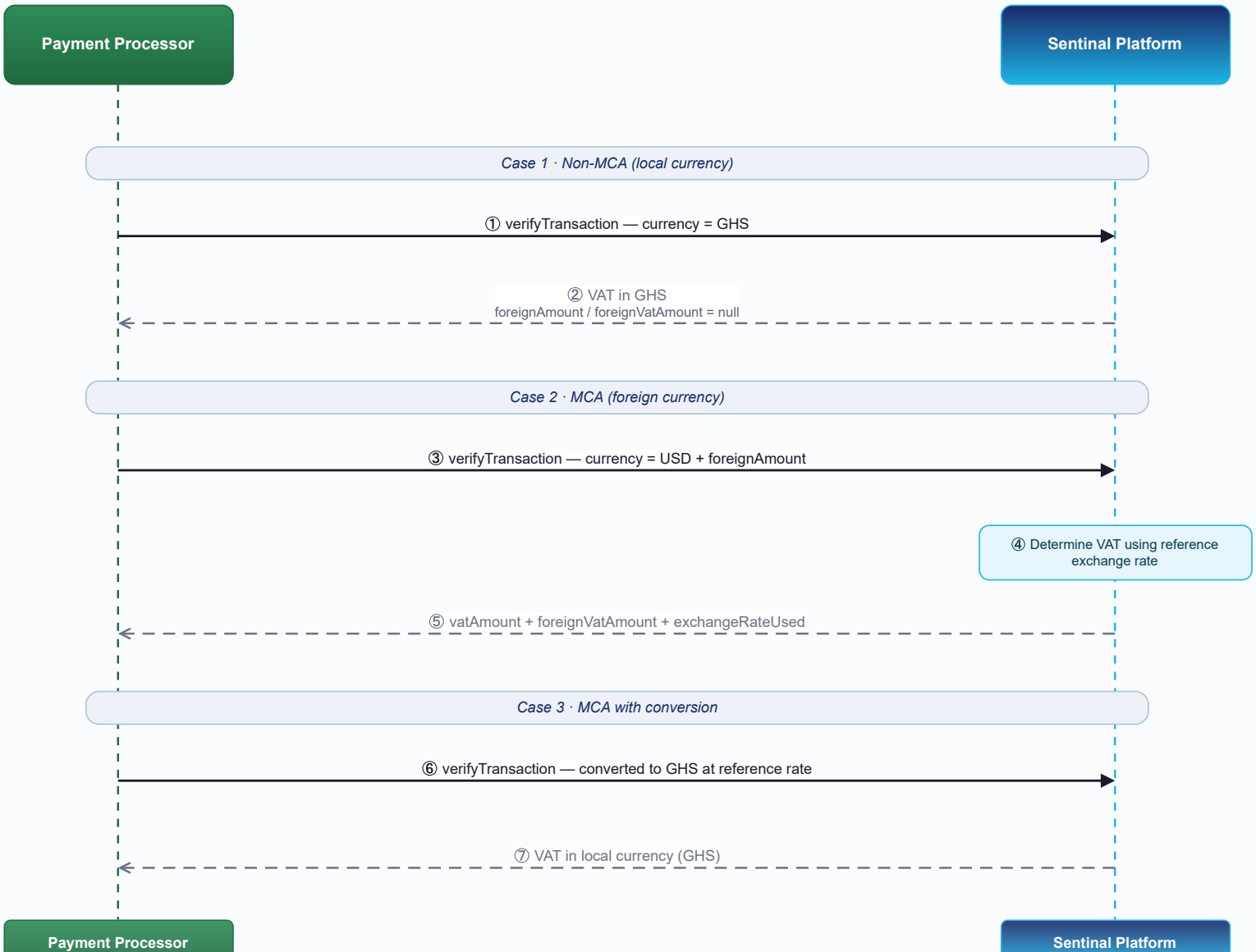
C.5 Non-PURCHASE (Participant) Transaction

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



C.6 Multi-Currency (MCA) Verification

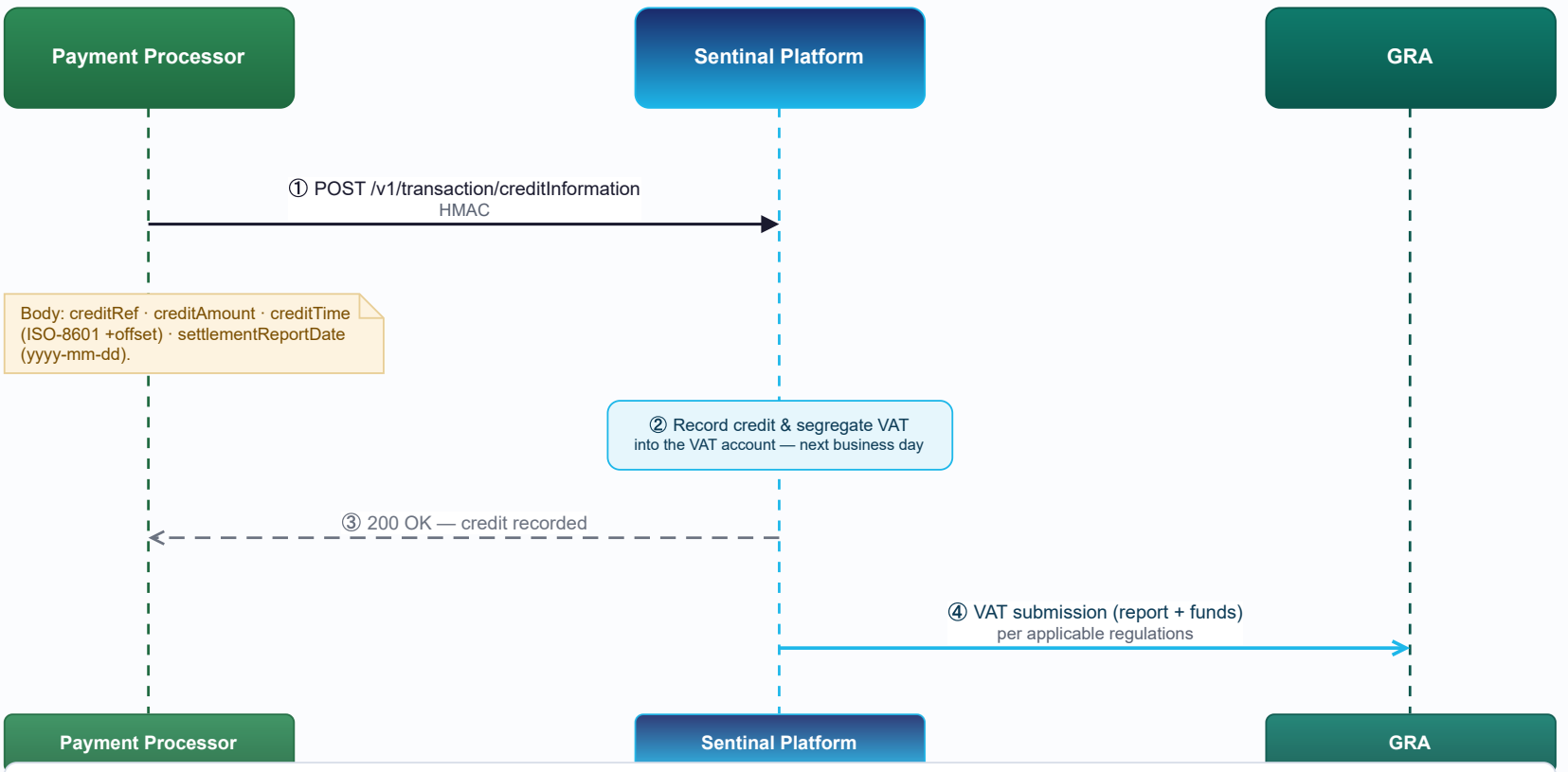
Three currency scenarios: non-MCA, MCA, and MCA-with-conversion



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

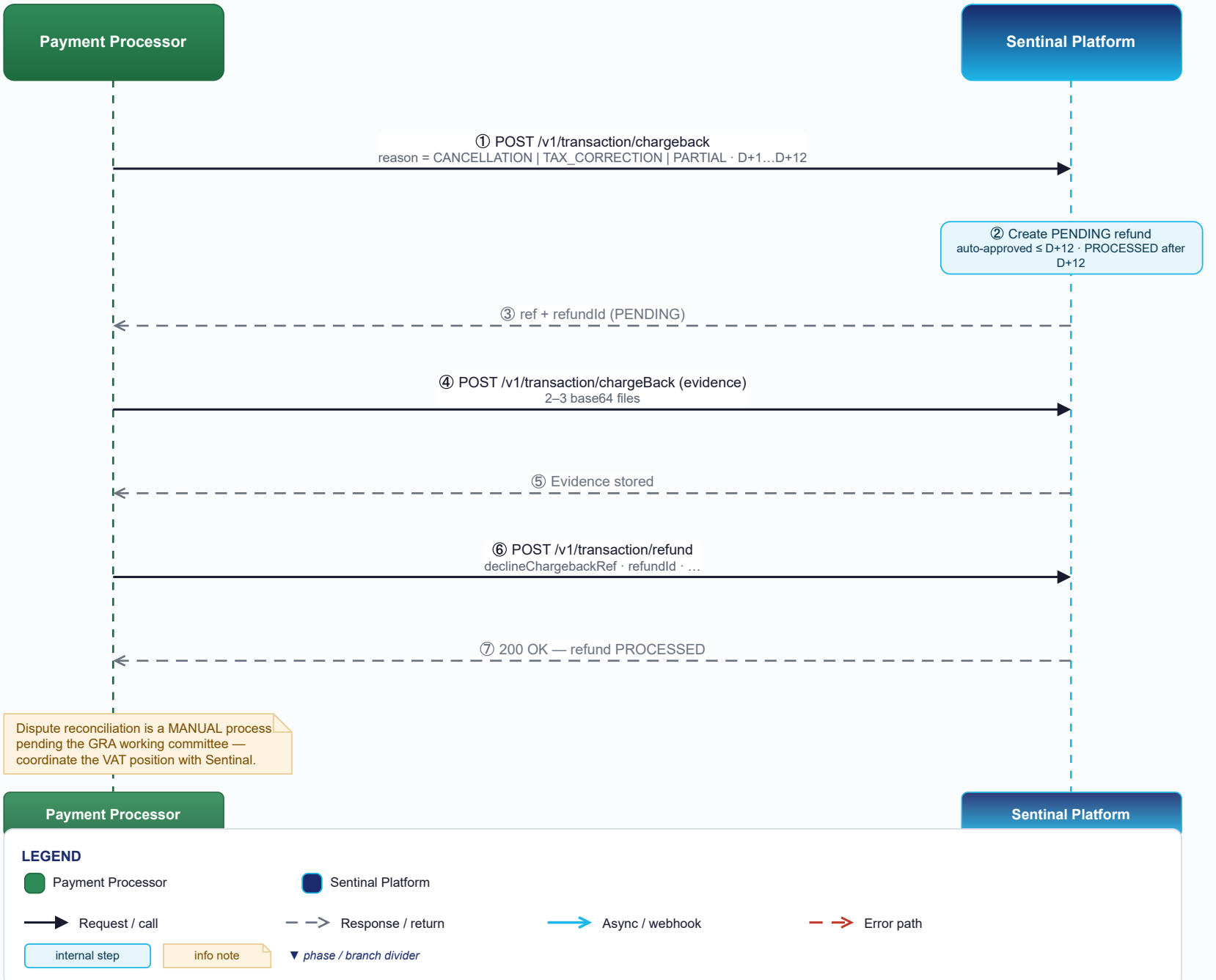


C.8 Decline → Refund (Same Day, D+0)

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

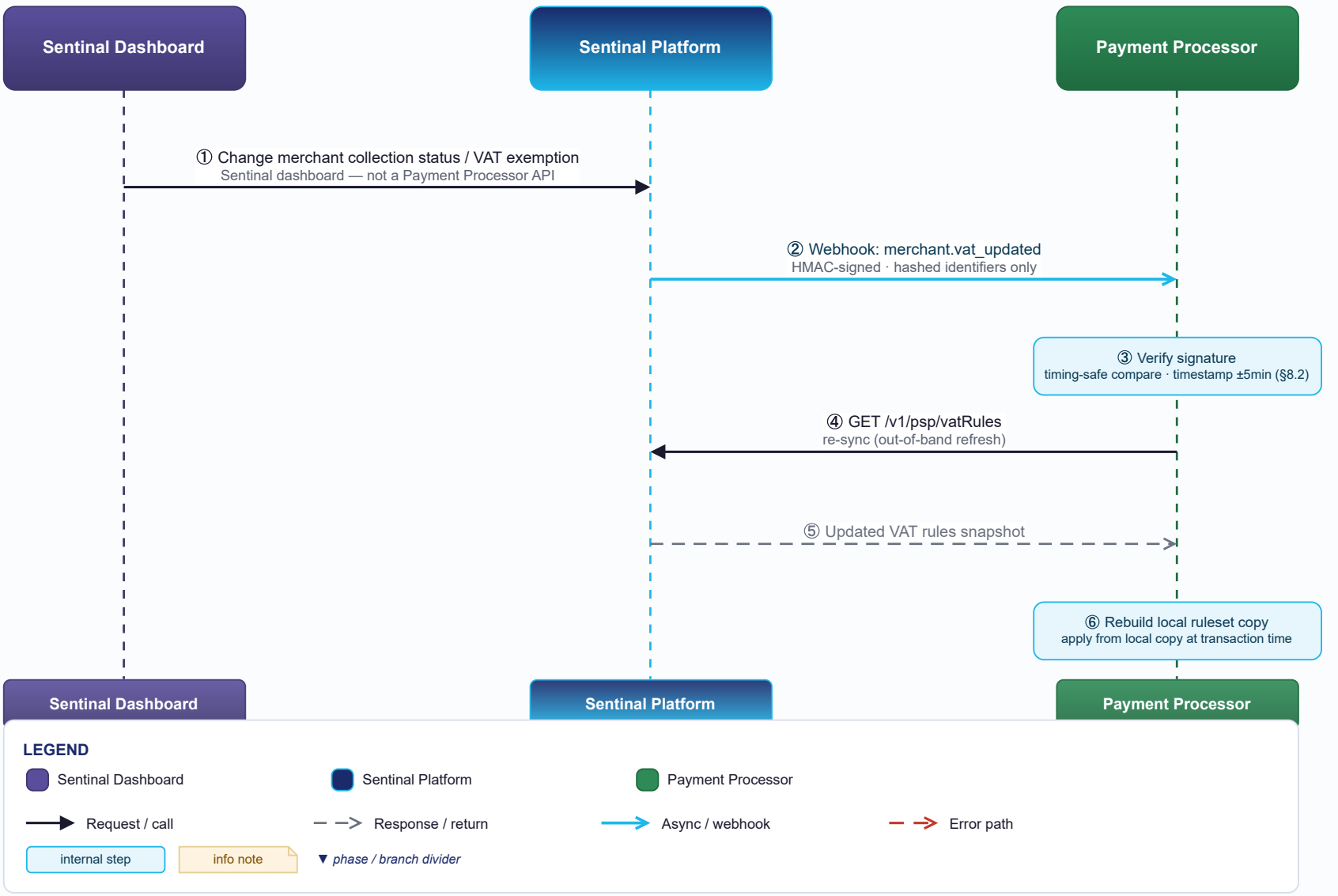
C.9 Chargeback (D+1 to D+12)

Chargeback record, evidence submission, resolution and refund



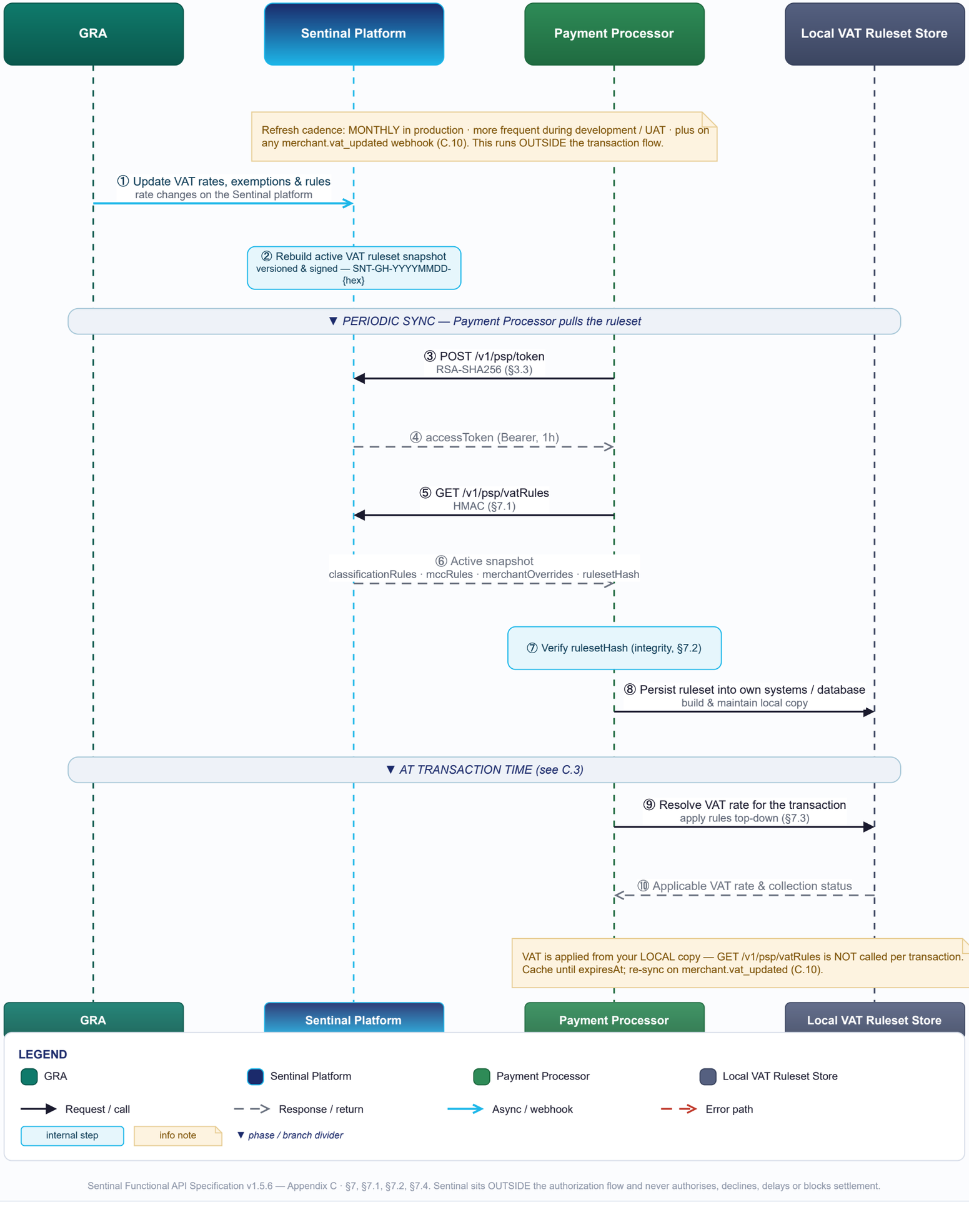
C.10 Merchant Status Change

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



C.11 VAT Rates Collection (Periodic Sync)

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

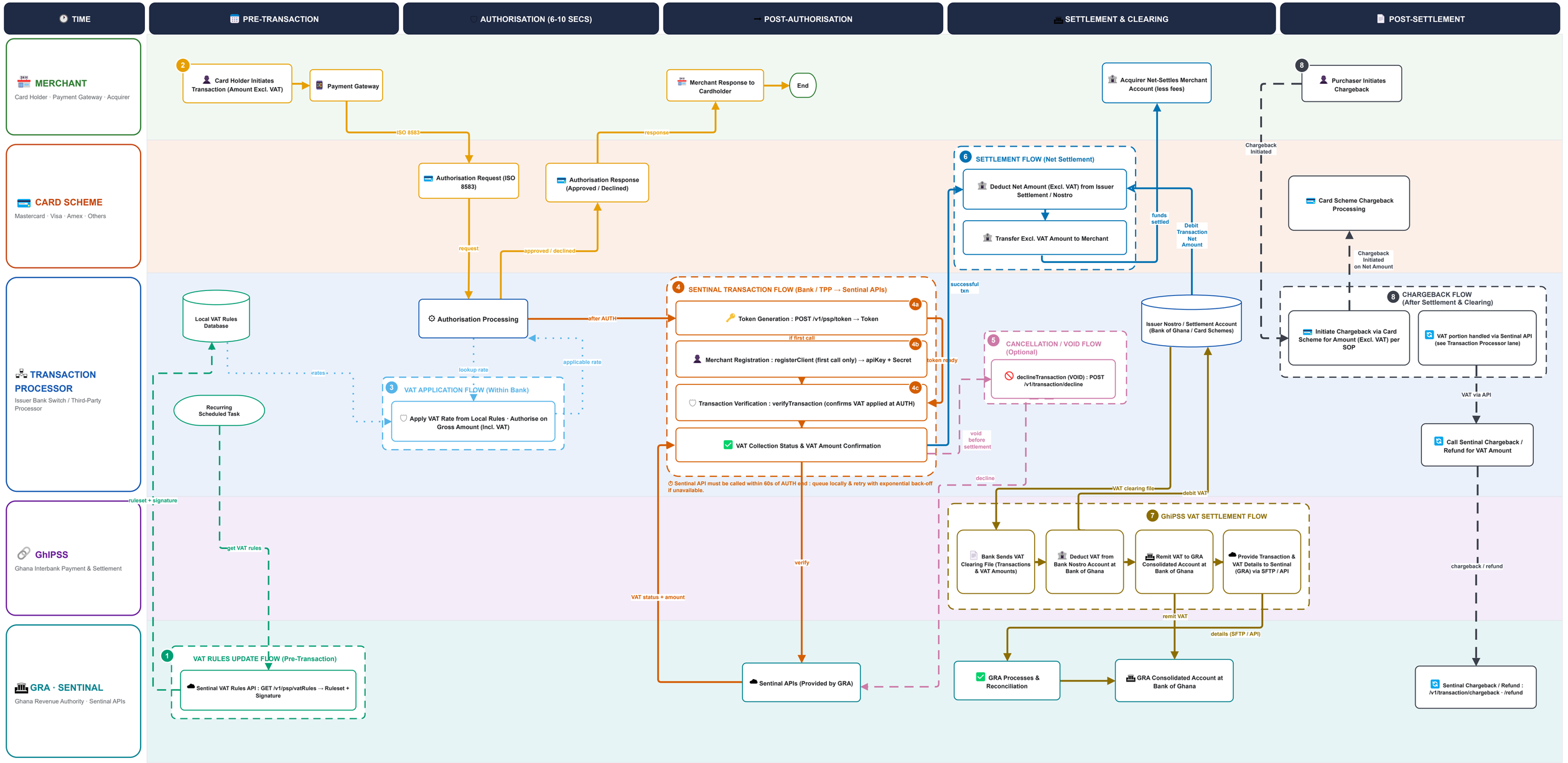


Appendix D — Bank Issuer Flow

Full-page flow — to be inserted as a separate page outside Word. This flow covers issuer-captured transactions where the actual merchant details reach the card scheme. Fintechs operating purely as a merchant-side payment gateway are captured through this flow and do not integrate separately (Section 2.1).

<<Diagram on next page>>

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

NOTES

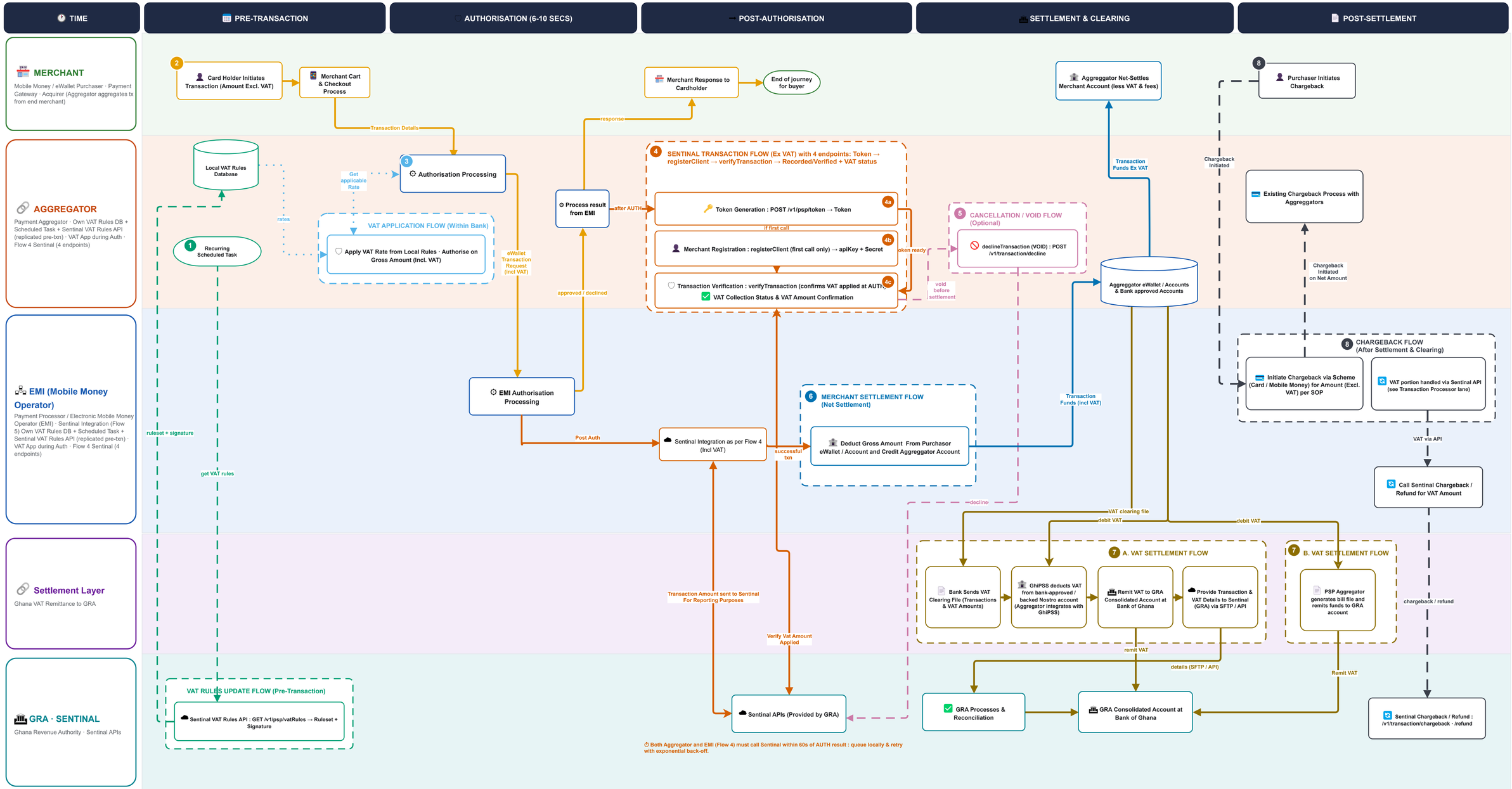
- Authorisation must complete within 6-10 seconds per Card Scheme rules.
- VAT is applied during AUTH using the bank's local VAT Rules; auth runs on the gross (Incl. VAT) amount.
- Sentinal API order: Token → registerClient (first call) → verifyTransaction; each step needs the prior credential.
- verifyTransaction 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

Appendix E — Aggregator Flow

Full-page flow — to be inserted as a separate page outside Word. This flow covers aggregators that collect payments locally under their own business details and settle offshore merchants outside the payment rail; these entities are obliged to integrate and report to Sentinal (Section 2.1).

<<Diagram on next page>>

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

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