□□ KRA GAVACONNECT

PHP SDK & Plugin Ecosystem

System Architecture & Developer Specification

Version	1.0.0-alpha
Target API	KRA GavaConnect (developer.go.ke)
PHP Requirement	PHP ^8.1
License	MIT
Packagist	`kra-php/sdk`

1. Executive Overview

The KRA GavaConnect PHP SDK is an open-source, framework-agnostic PHP library that provides clean, type-safe access to all 16+ KRA GavaConnect APIs. It removes the boilerplate of OAuth2 token management, HTTP retries, error handling, and data mapping — enabling any PHP developer in Kenya to integrate KRA tax services in under 30 minutes.

The SDK ships as a standalone Composer package (kra-php/sdk) and powers a family of first-class plugins for the most common PHP platforms used by Kenyan businesses: Laravel, WordPress, WooCommerce, Symfony, Filament, WHMCS, and Odoo.

1.1 What This Solves

- Every Kenyan developer currently hand-writes OAuth2 flows, token refresh logic, and raw HTTP calls to GavaConnect — duplicating thousands of lines of fragile boilerplate across the ecosystem.
- eTIMS compliance is becoming mandatory for all VAT-registered businesses. There is no standard PHP implementation. Businesses are non-compliant by default.
- Kenyan SaaS products (ERPs, billing systems, e-commerce) need KRA PIN verification at signup and TCC checks for supplier onboarding. Today this requires custom integrations for every product.
- Zero existing PHP SDKs exist with full API coverage, DTOs, test mocks, and framework plugins combined in one ecosystem.

1.2 Architecture Philosophy

- Framework-agnostic core — zero Laravel or Symfony dependencies in kra-php/sdk
- PSR compliance throughout — PSR-4 autoloading, PSR-7 HTTP messages, PSR-18 HTTP client, PSR-6 caching
- Dependency injection first — everything is interface-driven and mockable
- Fail loudly with typed exceptions — no silent failures, no mixed return types
- Offline-testable — full sandbox mode + Mockery-ready interfaces for unit tests without hitting real APIs

2. KRA GavaConnect API Reference

The GavaConnect platform (developer.go.ke) currently exposes 16 APIs across taxpayer verification, compliance checking, return filing, and eTIMS electronic invoicing. All APIs use OAuth2 client credentials flow except eTIMS which adds mutual TLS in production.

API Name	Endpoint	Description	Auth
PIN Validator	`GET /pin/validate`	Verify KRA PIN validity	OAuth2 Bearer
PIN by ID	`GET /pin/check-by-id`	Lookup PIN via National ID number	OAuth2 Bearer
TCC Checker	`GET /tcc/validate`	Tax Compliance Certificate status	OAuth2 Bearer
Taxpayer Obligations	`GET /taxpayer/obligations`	All tax obligations for a PIN	OAuth2 Bearer
Taxpayer Liabilities	`GET /taxpayer/liabilities`	Outstanding tax liabilities	OAuth2 Bearer
NIL Return	`POST /returns/nil`	File NIL tax return	OAuth2 Bearer
e-Slip Verify	`GET /eslip/verify`	Verify payment e-slip	OAuth2 Bearer
Excise License	`GET /excise/check`	Check excise license by number	OAuth2 Bearer
iTax Integration	`POST /itax/submit`	Submit tax data to iTax	OAuth2 Bearer
eTIMS Invoice	`POST /etims/invoice`	Submit electronic tax invoice	API Key + mTLS
eTIMS Stock IO	`POST /etims/stock`	Report stock in/out movements	API Key + mTLS
eTIMS Purchase	`POST /etims/purchase`	Register purchase transactions	API Key + mTLS
VAT Compliance	`POST /vat/file`	File VAT return (coming soon)	OAuth2 Bearer
PAYE Filing	`POST /paye/file`	File PAYE return (coming soon)	OAuth2 Bearer
Token (OAuth2)	`POST /oauth/token`	Get access token (client credentials)	Client ID + Secret
Token Refresh	`POST /oauth/refresh`	Refresh expired access token	Refresh Token

Register at developer.go.ke to obtain your Consumer Key and Consumer Secret. Use environment=sandbox for all development. Contact apisupport@kra.go.ke for production access approval.

3. PHP SDK Core Architecture

3.1 Package Structure

The SDK is organized into layered responsibilities. The client layer handles connectivity. The resource layer wraps individual API groups. DTOs provide typed return values. Exceptions provide granular error handling.

File / Path	Responsibility
`src/KraClient.php`	Core client — auth, HTTP, retry, rate limiting
`src/Auth/OAuth2Handler.php`	OAuth2 client credentials flow + token cache
`src/Auth/TokenCache.php`	PSR-6 token caching (file/Redis/Memcached)
`src/Http/HttpClient.php`	PSR-18 HTTP client adapter (Guzzle/Symfony)
`src/Http/RetryMiddleware.php`	Exponential backoff + jitter retry logic
`src/Resources/Pin.php`	PIN validation methods
`src/Resources/Tcc.php`	TCC verification methods
`src/Resources/Taxpayer.php`	Taxpayer obligations & liabilities
`src/Resources/Returns.php`	NIL and other return filing
`src/Resources/Etims.php`	eTIMS invoice, stock, purchase APIs
`src/Resources/ESlip.php`	e-Slip verification
`src/Resources/Excise.php`	Excise license checks
`src/Exceptions/KraException.php`	Base exception class
`src/Exceptions/AuthException.php`	Auth failures (token, credentials)
`src/Exceptions/ApiException.php`	API error responses (4xx, 5xx)
`src/Exceptions/RateLimitException.php`	Rate limit exceeded (429)
`src/DTOs/PinResult.php`	Typed response DTO for PIN checks
`src/DTOs/TccResult.php`	Typed response DTO for TCC checks
`src/DTOs/TaxpayerProfile.php`	Full taxpayer data DTO
`src/DTOs/EtimsInvoice.php`	eTIMS invoice request/response DTO
`src/Contracts/KraClientInterface.php`	Interface for mocking in tests
`src/Contracts/CacheInterface.php`	Pluggable cache contract
`src/Config/KraConfig.php`	Configuration value object
`tests/Unit/`	PHPUnit unit tests for all resources
`tests/Integration/`	Integration tests against sandbox
`tests/Fixtures/`	Mock API response fixtures

3.2 Authentication Flow

KraClient automatically manages the OAuth2 token lifecycle. Developers never call token endpoints directly.

```
// 1. Client instantiation
$kra = new KraClient([
    "client_id"     => env("KRA_CLIENT_ID"),
    "client_secret" => env("KRA_CLIENT_SECRET"),
    "environment"   => "sandbox",  // or "production"
    "cache"         => new RedisTokenCache($redis),
]);
```

```
// 2. First API call triggers auto token fetch
$result = $kra->pin()->validate("A000000010");
```

```
// 3. Token is cached (TTL: 3300s). Subsequent calls use cache.
// 4. On expiry, token is silently refreshed before the next request.
```

The OAuth2Handler posts to /oauth/token with grant_type=client_credentials, caches the response via the configured PSR-6 cache driver, and injects the Bearer token into every outgoing request via a Guzzle middleware.

3.3 PIN Validation — Full Code Example

```
use KraPHP\KraClient;
use KraPHP\Exceptions\ApiException;
use KraPHP\Exceptions\AuthException;
```

```
$kra = new KraClient(config("kra"));
```

```
try {
    // By KRA PIN
    $pin = $kra->pin()->validate("A000000010");
    echo $pin->taxpayerName;   // "ACME KENYA LIMITED"
    echo $pin->pinStatus;      // "ACTIVE"
    echo $pin->taxObligations; // ["VAT", "PAYE", "CIT"]
```

```
    // By National ID
    $pin2 = $kra->pin()->validateById("12345678");
    echo $pin2->kraPin;        // "A123456789B"
```

```
} catch (AuthException $e) {
    // Bad credentials or token failure
} catch (ApiException $e) {
    // KRA returned an error response
    echo $e->getKraErrorCode(); // "PIN_NOT_FOUND"
}
```

3.4 eTIMS Invoice Submission

```
$invoice = new EtimsInvoice([
    "invoiceNumber"    => "INV-2024-001",
    "buyerPin"         => "A000000020",
    "items"            => [
        ["description" => "Web Services", "qty" => 1,
         "unitPrice"   => 50000, "vatRate" => 16],
    ],
    "invoiceDate"      => "2024-11-01",
    "currency"         => "KES",
]);
```

```
$response = $kra->etims()->submitInvoice($invoice);
echo $response->controlUnit;  // "VSCU-KRA-XXXX"
echo $response->qrCode;       // base64 QR for receipt printing
echo $response->invoiceId;    // KRA-assigned unique ID
```

4. Environment Configuration

All sensitive credentials are loaded from environment variables. Never hardcode credentials in application code. The SDK reads configuration from a plain PHP array, making it compatible with any framework's environment system.

ENV Variable	Example Value	Description
`KRA_CLIENT_ID`	`your-client-id`	OAuth2 client ID from GavaConnect portal
`KRA_CLIENT_SECRET`	`your-client-secret`	OAuth2 client secret
`KRA_ENVIRONMENT`	`sandbox | production`	API environment target
`KRA_SANDBOX_BASE_URL`	`https://api-sandbox.developer.go.ke`	Sandbox base URL
`KRA_PROD_BASE_URL`	`https://api.developer.go.ke`	Production base URL
`KRA_ETIMS_SANDBOX_URL`	`https://etims-api-sbx.kra.go.ke`	eTIMS sandbox endpoint
`KRA_ETIMS_PROD_URL`	`https://etims-api.kra.go.ke/etims-api`	eTIMS production endpoint
`KRA_CACHE_DRIVER`	`file | redis | memcached`	Token cache driver
`KRA_CACHE_TTL`	`3300`	Token cache TTL in seconds
`KRA_TIMEOUT`	`30`	HTTP request timeout (seconds)
`KRA_RETRY_ATTEMPTS`	`3`	Max retry attempts on failure
`KRA_LOG_CHANNEL`	`stack`	Laravel log channel for SDK logs

4.1 Composer Installation

```
composer require kra-php/sdk
```

```
# Laravel plugin
composer require kra-php/laravel
```

```
# WordPress plugin — install via WP CLI or manual upload
wp plugin install kra-php-wordpress --activate
```

```
# WooCommerce extension
composer require kra-php/woocommerce
```

5. Plugin Ecosystem

Each plugin wraps the core SDK and adds platform-native integration points. All plugins share the same SDK under the hood, ensuring consistent behavior, error handling, and future API coverage across every platform.

Plugin	Package Name	Key Features
Laravel Package	`kra-php/laravel`	Service provider, Facade, config/kra.php, artisan commands: kra:check-pin, kra:file-nil
WordPress Plugin	`kra-php/wordpress`	Admin settings page, WooCommerce KRA PIN field at checkout, shortcodes, order tax compliance badge
WooCommerce Extension	`kra-php/woocommerce`	Auto-eTIMS invoice on order complete, customer PIN capture, compliance dashboard widget
Symfony Bundle	`kra-php/symfony`	DI container config, console commands, Twig extensions for KRA data display
Filament Plugin	`kra-php/filament`	Admin panel widgets: TCC status badge, PIN validator field, compliance dashboard
WHMCS Module	`kra-php/whmcs`	Invoice eTIMS sync, client KRA PIN field, automated compliance checking for invoices
Odoo Connector	`kra-php/odoo`	REST bridge to sync Odoo invoices to eTIMS, map Odoo tax codes to KRA obligation codes

5.1 Laravel Package Deep Dive

The Laravel package provides everything a Laravel developer expects: auto-discovery, config publishing, Facade, artisan commands, middleware, and Blade components.

```
// config/kra.php (published via artisan vendor:publish)
return [
    "client_id"     => env("KRA_CLIENT_ID"),
    "client_secret" => env("KRA_CLIENT_SECRET"),
    "environment"   => env("KRA_ENVIRONMENT", "sandbox"),
    "cache_driver"  => env("KRA_CACHE_DRIVER", "redis"),
];
```

```
// Facade usage in controllers
use KraPHP\Laravel\Facades\Kra;
```

```
$result = Kra::pin()->validate(request("kra_pin"));
$tcc    = Kra::tcc()->validate($pin, $tccNumber);
```

```
// Artisan commands
php artisan kra:check-pin A000000010
php artisan kra:file-nil --pin=A000000010 --obligation=VAT --period=2024-10
php artisan kra:sync-etims-invoices  // Batch sync pending eTIMS invoices
```

5.2 WooCommerce eTIMS Auto-Sync

When installed, the WooCommerce extension hooks into order completion and automatically submits the invoice to KRA eTIMS. Zero manual action required from the merchant.

```php
// Fires automatically on woocommerce_order_status_completed
add_action("woocommerce_order_status_completed", function($order_id) {
    $order    = wc_get_order($order_id);
    $invoice  = EtimsMapper::fromWooOrder($order);
    $response = KraPlugin::client()->etims()->submitInvoice($invoice);
    $order->update_meta_data("_kra_etims_id", $response->invoiceId);
    $order->update_meta_data("_kra_control_unit", $response->controlUnit);
    $order->save();
});
```

6. Build Roadmap & Phasing

The SDK is designed to be shipped incrementally. Phase 1 delivers a production-ready core with the highest-demand APIs. Each subsequent phase adds a new platform integration while the core SDK matures.

Phase	Timeline	Focus	Deliverables
Phase 1	Weeks 1-3	Core SDK	KraClient, OAuth2Handler, TokenCache, HttpClient, RetryMiddleware, Pin, Tcc, Taxpayer resources, DTOs, PHPUnit suite, sandbox tests
Phase 2	Weeks 4-6	Returns + eTIMS	Returns, ESlip, Excise, Etims resources, full eTIMS invoice/stock/purchase flow, integration test suite, Mockery mocks
Phase 3	Weeks 7-9	Laravel Package	ServiceProvider, Facade, config publish, artisan commands, Blade components, Laravel tests
Phase 4	Weeks 10-12	WordPress + WooCommerce	Admin settings, checkout PIN field, shortcodes, WooCommerce eTIMS auto-sync on order complete, compliance badge
Phase 5	Weeks 13-15	Symfony + Filament	Bundle config, DI wiring, console commands, Twig extensions, Filament widgets and panel
Phase 6	Weeks 16-18	WHMCS + Odoo + Docs	WHMCS module, Odoo REST bridge, full docs site (Docusaurus), live sandbox playground, Packagist publish

6.1 Testing Strategy

- Unit tests: PHPUnit 11 + Mockery. Every resource class tested in isolation using the KraClientInterface mock. No HTTP calls.
- Integration tests: Real sandbox API calls gated behind a GAVACONNECT_SANDBOX_ENABLED=true env flag. Run in CI nightly, not on every PR.
- Contract tests: Pact-based consumer-driven contract tests to detect API changes before they break production integrations.
- Platform tests: Each plugin has its own test suite using the platform's test framework (Laravel's Orchestra Testbench, WP_Mock for WordPress, etc.)
- Coverage target: 90% line coverage for core SDK. 80% for plugins.

6.2 CI/CD Pipeline (GitHub Actions)

```
# .github/workflows/test.yml
on: [push, pull_request]
matrix:
  php: ["8.1", "8.2", "8.3"]
steps:
  - composer install
  - vendor/bin/phpstan analyse  # Static analysis (level 8)
  - vendor/bin/phpcs            # PSR-12 code style
  - vendor/bin/phpunit          # Unit test suite
  - vendor/bin/infection        # Mutation testing score
```

7. Security Architecture

7.1 Credential Handling

- Client credentials are never logged, echoed, or included in exceptions. The KraConfig object implements __debugInfo() to scrub secrets from var_dump() output.
- Token storage: tokens are stored encrypted at rest when using the Redis driver. The encryption key is derived from APP_KEY (Laravel) or a configurable KRA_ENCRYPT_KEY.
- mTLS for eTIMS production: the SDK supports loading client certificates from the filesystem path or from an AWS Secrets Manager ARN.

7.2 Rate Limiting & Backoff

- The RetryMiddleware implements exponential backoff with jitter on 429 (Too Many Requests) and 5xx responses.
- Default: 3 retries. Base delay: 500ms. Max delay: 10s. Jitter: ±20%.
- The SDK tracks API call counts per minute in the cache layer and proactively throttles before hitting KRA limits.
- A CircuitBreaker can be enabled to halt requests after N consecutive failures, preventing cascade failures in high-throughput applications.

7.3 Data Privacy

- National ID numbers and KRA PINs are PII. The SDK never logs request bodies by default. Enable debug logging only in non-production environments.
- All eTIMS invoice data containing buyer PINs is transmitted over TLS 1.2+ minimum. The SDK rejects connections to endpoints without valid TLS certificates.
- The WordPress plugin stores KRA credentials in wp_options with values encrypted using WordPress's native encryption layer.

8. Open Source Monetization Strategy

The SDK core (kra-php/sdk) is MIT licensed and free forever. Revenue is generated through the ecosystem built around it.

8.1 Revenue Streams

Stream	Model	Target Market
Premium Plugins	One-time purchase or annual license	WHMCS, Odoo, Sage, QuickBooks connectors — complex integrations billed at KES 15,000-50,000/year
eTIMS Managed Service	KES 2,000-8,000/month SaaS	SMEs who want eTIMS compliance without hosting: hosted API relay, dashboard, invoice history, retry queue
Enterprise Support	Annual retainer KES 120,000+	Banks, telcos, large ERP deployments needing SLA, dedicated support channel, security reviews
Integration Consulting	Project-based	Custom GavaConnect integrations for companies who want the SDK customized for their stack
Developer Portal	Freemium API proxy	Sandbox relay with higher rate limits, test data seeding, and team credential management for KES 500/month

8.2 Community Growth

- Launch at a KRA API Masterclass event — KRA actively promotes community developers building on GavaConnect.
- Submit to awesome-kenya-developer-tools and Kenyan tech communities (Nairobi Dev, AfricaTechies).
- Write the definitive KRA integration guide on dev.to — this will be the #1 search result for "KRA PHP API integration" within weeks.
- Offer free integration review for the first 10 companies who adopt the SDK — generates case studies and real-world feedback.
- Establish as the official community SDK by engaging KRA's developer relations team at apisupport@kra.go.ke.

9. Developer Quick Start

A developer with valid GavaConnect sandbox credentials should be making real API calls within 5 minutes of running composer require.

Step 1 — Install

```
composer require kra-php/sdk
```

Step 2 — Configure

```
# .env
KRA_CLIENT_ID=your-sandbox-client-id
KRA_CLIENT_SECRET=your-sandbox-client-secret
KRA_ENVIRONMENT=sandbox
```

Step 3 — Make Your First Call

```
<?php
require "vendor/autoload.php";
```

```
use KraPHP\KraClient;
```

```
$kra = new KraClient([
    "client_id"     => getenv("KRA_CLIENT_ID"),
    "client_secret" => getenv("KRA_CLIENT_SECRET"),
    "environment"   => getenv("KRA_ENVIRONMENT"),
]);
```

```
// Validate a KRA PIN
$result = $kra->pin()->validate("A000000010");
print_r($result->toArray());
```

For full documentation, visit the project README on GitHub or the hosted docs site. For sandbox credentials, register at developer.go.ke. For API support, contact apisupport@kra.go.ke.

Built for Kenya. Open to the world.

github.com/kra-php/sdk | developer.go.ke | apisupport@kra.go.ke