OPTISLEEK

TECHNICAL PROJECT OVERVIEW

WooCommerce intelligence integration

A code-based overview of the Optisleek WordPress plugin: purpose, architecture, capabilities, interfaces, operations, and current implementation status.

Plugin version 1.0.4
Prepared from the repository source

29 September 2026
FDN Group

01 · EXECUTIVE SUMMARY

What the project does

Optisleek connects a WooCommerce store to a remote optimization platform while keeping WordPress responsible for storefront rendering, catalog access, attribution, and merchandising order.

In one sentence: the plugin exports catalog data, captures shopping behavior and purchases, renders externally powered recommendation experiences, and applies product-ordering rules returned by the Optisleek service.

1.0.4	6.7+	7.4+	WC
Current plugin version	Minimum WordPress version	Minimum PHP version	WooCommerce is required

1

Observe

Front-end scripts collect product views, engagement, selected variations, cart context, and order outcomes. Recommendation and landing-page identifiers are retained through WooCommerce sessions and order metadata.

2

Exchange

Catalog entities—brands, categories, attribute definitions, attribute values, products, variations, prices, stock, and images—are exposed for synchronization with the remote service.

3

Recommend

Product pages can display “Related Products” and “Bought Together” modules powered by the remote `products.js` SDK, with a local mock mode for integration testing.

4

Merchandise

Category-specific product sequences are stored locally and applied to WooCommerce catalog queries. Rules can be managed in WordPress, through the REST API, or synchronized on a schedule.

Boundary of responsibility. This repository contains the WordPress integration layer. The recommendation and optimization algorithms themselves are provided by the external Optisleek platform rather than implemented inside the plugin.

02 · SYSTEM ARCHITECTURE

How information moves

The integration is bidirectional: WooCommerce sends catalog and behavioral context outward; Optisleek returns presentation assets and merchandising decisions.

WooCommerce store

Products, variations, categories, attributes, shoppers, carts, and orders.

↔

Optisleek plugin

WordPress hooks, REST/export interfaces, sessions, widget anchors, and sorting table.

↔

Optisleek platform

Tracking endpoints, widget SDK, recommendation content, and ranked product lists.

Catalog sync	WordPress reads WooCommerce records → serializes store entities → publishes them through REST and a legacy rewrite endpoint → remote tooling imports the data.
Behavior	Storefront hooks inject tracking code → browser events identify product/variation interaction → identifiers are persisted in the WooCommerce session → purchase events are reported at checkout.
Recommendations	The plugin adds a target element to the product page → loads the remote widget SDK once → initializes each enabled widget with store, pixel, and product context.
Sorting	Twice-daily WP-Cron requests ranked products → rules are stored in `{prefix}_optisleek_sorting` → a WooCommerce query filter prioritizes the configured product IDs.
Attribution	Recommendation and landing-page GUIDs follow the shopper through session/cookie state → are captured on the order → support client-side reporting plus a server-side fallback.

Runtime model

Admin
Settings, notices, documentation, widget controls, sync logs, and sorting-rule management.

Storefront
Tracking, product widgets, landing-page shortcode, attribution, and catalog ordering.

Background
WP-Cron rule synchronization and delayed purchase-reporting fallback.

03 · COMPONENTS

Codebase map

The plugin is organized as a lightweight PHP module set with WordPress hooks, plus CSS and JavaScript assets for its admin and storefront surfaces.

COMPONENT	RESPONSIBILITY	STATUS
`optisleek.php`	Plugin metadata, activation/deactivation hooks, and bootstrap.	Active
`class-optisleek.php`	Loads modules and registers admin, storefront, widget, sorting, landing-page, and REST behavior.	Active
`class-admin-page.php`	Five admin areas: general settings, product widgets, landing pages, sorting, and documentation.	Active
`class-product-export.php`	Builds the WooCommerce catalog payload and serves the alternative `/optisleek/api/` interface.	Active
`class-rest-api.php`	Catalog, sorting CRUD, and remote cron-trigger routes with token checks where configured.	Active
`tracking-pixelV2.php`	Detailed browser tracking, engagement, GUID attribution, checkout reporting, and fallback delivery.	Active
`class-tracking-core.php`	Loads the primary remote tracking script when the integration is enabled and configured.	Active
`class-product-widget-base.php`	Shared widget lifecycle, SDK loading, target placement, configuration, and test mode.	Active
`class-related-products-widget.php`	Configuration for the Related Products experience.	Active
`class-bought-together-widget.php`	Configuration for the Bought Together experience.	Active
`optisleek-landing-pages.php`	Landing-page shortcode, widget bootstrapping, URL attribution, session/cookie state.	Active
`class-sorting.php`	Rule persistence, scheduled synchronization, logging, and SQL-level WooCommerce ordering.	Active
`class-attribute-erp.php`	Planned ERP code fields for WooCommerce attribute terms. Handlers remain placeholders and the module is not bootstrapped.	Dormant

Dependencies. Runtime behavior uses WordPress and WooCommerce APIs only. Composer currently declares WordPress 7.1 and WooCommerce 11 stubs as development dependencies; they are not runtime libraries.

04 · CAPABILITIES

Functional scope

Catalog export

- Brands and product categories
- Global attribute templates and values
- Simple and variable products
- Images, prices, availability, and relationships
- Optional pagination up to 500 records per request
- Incremental selection through `modified_after`

Behavior and conversion tracking

- Product-page activity and variation changes
- Recommendation engagement identifiers
- Landing-page attribution identifiers
- Order capture for classic and Store API checkout
- Client confirmation and scheduled server fallback

Recommendation widgets

- Related Products
- Bought Together
- Default WooCommerce insertion hooks
- Custom CSS selectors for theme builders
- One shared SDK load for active widgets
- Local mock/test mode

Landing pages

- `[optisleek_landing_page]` shortcode
- Per-shortcode token, URL, and pixel overrides
- Elementor preview accommodations
- Session storage with cookie fallback
- Attribution passed into purchase tracking

Product sorting

- Category-specific ordered product lists
- REST and admin CRUD controls
- Twice-daily synchronization
- Manual synchronization and logs
- “Recommended (Optisleek)” catalog option
- Stable fallback ordering for unranked products

Administration

- Connection URL, pixel code, API token, and shop ID
- Color and size attribute mapping
- Feature-level enable/disable switches
- Configuration warnings and welcome screen
- Bundled documentation browser
- Nonce-protected AJAX management actions

Operational dependency. Live tracking, widgets, landing-page results, and sorting synchronization depend on a reachable and correctly configured Optisleek service URL. Storefront features fail independently when their required configuration is missing.

05 · INTERFACES AND CONFIGURATION

Integration surface

REST routes

METHOD	ROUTE	PURPOSE / ACCESS
GET	`/wp-json/optisleek/products`	Catalog payload. Public by design; supports `page`, `per_page`, and `modified_after`.
GET · POST	`/wp-json/optisleek/sorting`	List or create sorting rules. Requires the configured API token.
PUT · PATCH · DELETE	`/wp-json/optisleek/sorting/{id}`	Update or remove one sorting rule. Requires the configured API token.
GET	`/wp-json/optisleek/cron-sync`	Trigger a sorting synchronization. Requires the separate cron secret.

Alternative catalog interface

`/optisleek/api/?sync={resource}` serves `all-json`, `brands`, `categories`, `attributes-templates`, `attributes`, or `products`. Product calls accept `pg`, `per_page`, and `modified_after`.

Primary configuration

SETTING	ROLE
`optisleek_url`	Base URL of the remote Optisleek service.
`optisleek_code`	General tracking/pixel identifier and widget fallback identifier.
`api_token`	Authenticates sorting APIs and remote synchronization requests.
`optisleek_shop_id`	Identifies the WooCommerce shop to the remote service.
`optisleek_enabled`	Master switch for front-end integration behavior.
`optisleek_color_attribute` `optisleek_size_attribute`	Maps WooCommerce attributes to the color and size concepts used in exports and tracking.

> **Authentication behavior.** Sorting routes accept the token through `X-Api-Token` or an `api_token` parameter. The cron route uses a distinct generated secret. WordPress admin operations are capability- and nonce-protected.

06 · DEPLOYMENT AND OPERATION

How to run the integration

Installation baseline

- Use WordPress 6.7 or newer.
- Use PHP 7.4 or newer.
- Install and activate WooCommerce.
- Place the plugin in `wp-content/plugins/optisleek`.
- Activate Optisleek in WordPress.

Required configuration

- Enter the remote service URL.
- Enter the Optisleek code/pixel ID.
- Enter the API token and shop ID.
- Select store color and size attributes.
- Enable the master integration switch.

Widget rollout

1. Enable test mode and the desired widget.
2. Confirm placement on a product page.
3. If the theme omits the standard WooCommerce hook, configure a CSS selector.
4. Disable test mode only after the live SDK URL and pixel are validated.

Sorting rollout

1. Confirm the remote service returns ranked product groups.
2. Run a manual synchronization and review its log.
3. Inspect rules before enabling storefront sorting.
4. Confirm WP-Cron runs reliably in the hosting environment.

Operational checks

AREA	CHECK	EXPECTED RESULT
Catalog	Request a small paginated export.	Valid JSON with store entities and published products.
Tracking	View a product, choose a variation, add to cart, and place a test order.	Browser requests complete and order GUID metadata is captured when applicable.
Widgets	Open a product with test mode enabled.	Mock recommendations render exactly once at the intended location.
Landing page	Render the shortcode with a token and visit an attributed URL.	The module loads and attribution persists through checkout.
Sorting	Sync, enable a rule, and open its category.	Ranked products appear first; remaining products retain stable ordering.

07 · CURRENT STATUS

Strengths and next priorities

The integration covers the complete operational loop, but several engineering improvements would reduce risk and make future releases easier to maintain.

Established strengths

- Clear WordPress/WooCommerce lifecycle integration
- Centralized widget base class avoids duplicate SDK loads
- Pagination and incremental catalog synchronization
- Separate API and cron secrets for protected routes
- Stable sorting fallback for products outside a ranking rule
- Attribution survives checkout and has a delivery fallback
- Admin notices expose missing settings and unsafe test mode

Known implementation gaps

- The ERP attribute module contains placeholder methods and is not loaded.
- No automated test suite or continuous-integration configuration is present.
- Tracking logic is concentrated in one large procedural file.
- REST routes do not use a versioned namespace.
- The project has no uninstall cleanup routine for options or its custom table.
- The declared localization domain has no language catalog in this repository.

Recommended engineering priorities

#	PRIORITY	RECOMMENDATION
1	Transport security	Remove `sslverify => false` from sorting synchronization and resolve any certificate-chain issue at the environment level.
2	API hardening	Confirm public catalog exposure is intentional, prefer header-only credentials, use POST for state-changing cron triggers, add argument schemas, and rate-limit external calls.
3	Automated verification	Add unit/integration coverage for export payloads, REST authorization, attribution state, checkout reporting, and SQL ordering.
4	Maintainability	Split `tracking-pixelV2.php` into focused services and centralize remote HTTP behavior, error handling, and timeouts.
5	Data governance	Document retention, consent, cookie behavior, exported fields, and the handling of order-linked identifiers for production operators.
6	Release hygiene	Add an uninstall policy, version API routes, package only required files, and complete or remove the dormant ERP attribute feature.

Review note. These priorities are based on static inspection of the repository. They are not a penetration test, privacy assessment, or verification against a running WordPress installation.

08 · REFERENCE

Repository at a glance

PHP modules

`includes/`

Administration, export, REST, tracking, widgets, landing pages, sorting, and the dormant attribute ERP scaffold.

Browser assets

`assets/css/` · `assets/js/`

Admin styling and behavior, storefront widget styling, and a local recommendation-widget mock.

Documentation

`docs/`

Bundled customer-facing documents displayed in WordPress. Files shipped here are publicly addressable if their URL is known.

Development metadata

`composer.json` · `composer.lock`

IDE/static-analysis stubs for current WordPress and WooCommerce APIs.

Project assessment

Optisleek v1.0.4 is a feature-rich WooCommerce connector with a coherent commercial purpose and a broad production surface. Its strongest qualities are the end-to-end data loop and practical storefront integration. The best next investment is hardening the network/API boundary and adding automated coverage around checkout attribution, catalog synchronization, and sorting.

DOCUMENT BASIS

Prepared from the repository contents available on 29 September 2026. No live WordPress instance, production configuration, or remote Optisleek service was exercised.

SCOPE

Technical overview rather than exhaustive API specification, compliance opinion, performance benchmark, or security audit.

OPTISLEEK

Infinite optimization for ecommerce