

Atlas Headless WordPress API – Custom REST API & Headless CMS

Plugin Project Documentation

Architecture · Features · Roadmap

Version	Status	Author	Last Updated
1.0.0	Phase 1 — Active	You (Lead Dev)	16 April 2026

This document serves as the canonical reference for the Atlas Headless WordPress API – Custom REST API & Headless CMS plugin. It covers the project overview, technical architecture, Phase 1 feature specifications, known challenges, and the complete future roadmap. Keep this document updated as the plugin evolves.

Tag Line: Turn WordPress into a controlled Headless CMS with a custom API layer.

1. Project Overview

1.1 Plugin Name & Identity

Property	Value
Plugin Name	Atlas Headless WordPress API – Custom REST API & Headless CMS
Slug	atlas-headless-wordpress
Namespace	atlas-headless-wordpress/v1
Min WordPress	6.0+
Min PHP	8.0+
License	GPL-2.0+
API Style	REST (Phase 1) — GraphQL extension planned Phase 3

1.2 Problem Statement

WordPress ships with a frontend theme system and a default REST API (`/wp-json/wp/v2/`) that exposes too much information for headless deployments, has inconsistent response shapes, and provides no access-control layer for per-content-type filtering. Developers going headless must cobble together multiple plugins, none of which share a unified settings interface.

1.3 Solution

Atlas Headless WordPress API transforms WordPress into a clean, controlled API backend by:

- Disabling the WordPress frontend theme layer entirely
- Selectively removing default WP REST API routes without breaking the Block Editor
- Registering a clean, versioned custom REST namespace
- Providing granular per-content-type access controls (All / Only These / Exclude These)
- Resolving ACF and Secure Custom Fields data natively in every response
- Exposing Menus, Media, CPTs, Taxonomies, and navigation data in developer-friendly shapes
- Offering a unified admin settings panel to manage all of the above

1.4 Existing Landscape

No single plugin achieves what Atlas Headless WordPress API (AHWA) does. The closest alternatives and their gaps:

Plugin	What it does	Gap vs Atlas Headless WordPress API – Custom REST API & Headless CMS
WPGraphQL	GraphQL API	Doesn't disable REST; no settings UI; no ACF menu unification
Headless Mode (WP Engine)	Disables frontend	No custom API layer; no filtering controls
ACF to REST API	Exposes ACF in REST	Single-purpose; no settings; exposes all ACF without control
Faust.js (WP Engine)	Headless framework	Opinionated; Next.js-locked; not a standalone plugin
JWT Authentication	Adds JWT to REST	Auth only; no API management

Full Name: Atlas Headless WordPress API

Short Name: Atlas WP

Internal Prefix (code): atlas_ or ahwp_

2. Technical Architecture

2.1 Directory Structure

```
atlas-headless-wordpress/
|
|— atlas-headless-wordpress.php      # Bootstrap (entry point)
|— uninstall.php
|
|— /_init
|   |— app.php                      # App container init
|   |— hooks.php                   # Central hook registration
|
|— /config
|   |— app.php                      # Global config
|   |— api.php                     # API config (namespace, version)
|   |— auth.php                    # Auth config
|
|— /src
|   |— /Core
|       |— Application.php          # Main app class
|       |— Container.php           # Dependency injection container
|       |— ServiceProvider.php      # Base provider
|   |
|   |— /API
|       |— APIManager.php          # Registers all routes
|       |— RouteRegistrar.php
|       |— Middleware.php
|   |
|   |— /Http
|       |— Request.php
|       |— Response.php
|       |— Controllers/
|           |— BaseController.php
|           |— PostController.php
|           |— PageController.php
|           |— CPTController.php
|           |— TaxonomyController.php
|           |— MediaController.php
|           |— MenuController.php
|   |
|   |— /Services
```

```

├── ContentService.php
├── TaxonomyService.php
├── MediaService.php
├── MenuService.php
├── FieldResolver.php # ACF/SCF
├── /Repositories
│   ├── PostRepository.php
│   ├── TaxonomyRepository.php
│   └── MediaRepository.php
├── /Domain
│   ├── Entities/
│   │   ├── PostEntity.php
│   │   ├── TermEntity.php
│   │   └── MediaEntity.php
├── /Support
│   ├── Helpers.php
│   └── Formatter.php
├── /Features
│   ├── FrontendDisabler.php
│   ├── RestRestrictor.php
│   └── AccessControl.php
├── /Auth
│   ├── APIKeyAuth.php
│   └── JWTAuth.php (Phase 2)
├── /Settings
│   ├── SettingsManager.php
│   └── SettingsSchema.php
├── /Admin
│   ├── AdminMenu.php
│   ├── Pages/
│   │   └── SettingsPage.php
├── /resources
│   ├── views/
│   ├── js/
│   └── css/

```

```
|  
| — /routes  
|   | — api.php                # Central route definitions  
|  
| — /tests (Phase 3+)
```

2.2 Architecture Pattern (Important)

Layered Architecture

Client Request
↓
Route (routes/api.php)
↓
Controller
↓
Service
↓
Repository
↓
WordPress Core

API Namespace & Versioning

All custom endpoints live under:

```
GET / wp-json/atlas/v1/
```

The namespace "whc" (Atlas Headless WordPress API – Custom REST API & Headless CMS) is configurable in settings. Versioning is baked in from day one — /v2/ can be added in Phase 3 without breaking existing consumers.

2.3 Authentication Strategy (Phase 1)

Phase 1 uses API Key authentication — lightweight, no external dependency, works for server-to-server headless setups:

- API keys are generated in the plugin settings panel
- Keys are passed via the X-WHC-API-Key request header
- Public endpoints (configurable) can bypass authentication entirely
- Phase 2 will add JWT Bearer token support for user-context requests

2.4 Response Envelope

Every endpoint returns a consistent JSON envelope:

```
{  
  "success": true,
```

```
"data": { ... },
"meta": {
  "total": 42,
  "total_pages": 5,
  "current_page": 1,
  "per_page": 10
},
"error": null
}
```

3. Phase 1 — Feature Specifications

3.1 Frontend Disabler

Hooks used to fully disable the WordPress frontend without affecting wp-admin or the Block Editor REST calls:

Hook	Purpose
template_redirect	Intercept all frontend requests and return 410 Gone or 200 with custom message
wp_enqueue_scripts	Dequeue all theme scripts and styles
show_admin_bar	Remove admin bar on any surviving frontend output
rest_endpoints (filter)	Surgically remove default WP REST routes while preserving internal Gutenberg routes

IMPORTANT: The following default REST routes are preserved to keep Gutenberg functional:

- /wp/v2/block-renderer
- /wp/v2/settings (internal)
- /wp/v2/types (used for CPT block support)
- /wp/v2/block-directory

3.2 API Endpoints — Phase 1

3.2.1 Posts

Route	Description
GET /atlas/v1/posts	List posts with pagination, filtering by category, tag, author, date range
GET /atlas/v1/posts/{id}	Single post by ID — includes ACF/SCF fields, featured image, categories, tags, author
GET /atlas/v1/posts/slug/{slug}	Single post by slug

3.2.2 Pages

Route	Description
GET /atlas/v1/pages	List pages — supports hierarchical tree format param
GET /atlas/v1/pages/{id}	Single page by ID with full ACF/SCF resolution

GET /atlas/v1/pages/slug/{slug}	Single page by slug — supports nested slugs (parent/child)
---------------------------------	--

3.2.3 Custom Post Types

Route	Description
GET /atlas/v1/cpt	List all registered public CPTs (name, slug, labels, supports)
GET /atlas/v1/cpt/{post_type}	List items of a specific CPT with filtering
GET /atlas/v1/cpt/{post_type}/{id}	Single CPT item with full ACF/SCF resolution

3.2.4 Categories & Taxonomies

Route	Description
GET /atlas/v1/categories	List categories — hierarchical tree or flat list (param)
GET /atlas/v1/categories/{id}	Single category with ACF/SCF fields and post count
GET /atlas/v1/taxonomies	List all registered public taxonomies
GET /atlas/v1/taxonomies/{taxonomy}	List all terms for a given taxonomy with ACF/SCF
GET /atlas/v1/taxonomies/{taxonomy}/{id}	Single term with full field resolution

3.2.5 Media

Route	Description
GET /atlas/v1/media	List media — filter by type (image/video/pdf), mime type, date
GET /atlas/v1/media/{id}	Single media item with all sizes, alt text, ACF fields, srcset data

3.2.6 Menus

Route	Description
GET /atlas/v1/menus	List all registered nav menus with location slugs
GET /atlas/v1/menus/{id}	Full menu tree by menu ID — parent/child hierarchy resolved
GET /atlas/v1/menus/location/{location}	Menu by theme location slug (e.g. primary, footer)

3.3 ACF & Secure Custom Fields Resolution

The FieldResolver class handles all ACF/SCF data. Key rules:

- All ACF/SCF fields are included in a "fields" key on every response object
- Flat value types (text, number, select, checkbox, etc.) are returned as-is
- Relational fields (post_object, relationship) resolve to full objects (ID, title, slug, permalink) — not raw IDs
- Image/file fields return full URL, title, alt text, width, height, and all registered sizes
- Repeater fields are returned as arrays of field-keyed objects
- Flexible content fields are returned as layout-typed arrays
- Gallery fields return arrays of image objects (same shape as media endpoint)
- Group fields are returned as nested objects

NOTE: To prevent N+1 query issues, relational field resolution depth is limited to 1 level in Phase 1. Deep nesting will be addressed with a dataloader pattern in Phase 2.

3.4 Access Control Settings (Per Content Type)

Each content type in the settings panel offers three modes:

Mode	Setting Key	Behaviour
All	access = "all"	All published items returned — default behaviour
Only These	access = "include"	Whitelist: only selected IDs/slugs are exposed via the API
Exclude These	access = "exclude"	Blacklist: all items returned except selected IDs/slugs

Selection UI uses an async searchable post selector (Select2) to handle large datasets. Settings are stored in wp_options as serialized arrays per content type.

4. Admin Settings Panel

The settings panel is accessible at WP Admin → Settings → Atlas Headless WordPress API – Custom REST API & Headless CMS. It is built using native WordPress Settings API + Options API with a React-powered component for the post selectors.

4.1 Settings Tabs (Phase 1)

Tab	Options
General	Enable/disable plugin, API namespace slug, frontend disable toggle, response format
Authentication	Generate/revoke API keys, set public vs authenticated endpoints per route
Content	Per-type access controls: Posts, Pages, each CPT, all Taxonomies, Media, Menus
Fields	ACF/SCF enable toggle, relational field depth (1 or 2), field name overrides
Status	Read-only: active endpoints list, API key usage logs, last request timestamps

4.2 Settings Storage Schema

```
// wp_options key: whc_settings
{
  "general": { "enabled": true, "namespace": "whc", "disable_frontend": true },
  "auth": { "require_key": true, "public_routes": ["/atlas/v1/posts"] },
  "content": {
    "posts": { "access": "all", "ids": [] },
    "pages": { "access": "include", "ids": [1,2,5] },
    "product": { "access": "exclude", "ids": [99] },
    "media": { "access": "all", "ids": [] }
  },
  "fields": { "acf_enabled": true, "depth": 1 }
}
```

5. Complexity & Known Challenges

Challenge	Severity	Mitigation
Surgical REST route removal without breaking Gutenberg	HIGH	Maintain explicit whitelist of Gutenberg-required routes; test against full block editor on every build
ACF relational field N+1 queries	HIGH	Limit depth to 1 in Phase 1; implement dataloader/batch pattern in Phase 2
Menu tree reconstruction from flat WP structure	MEDIUM	Build recursive walker utility in Phase 1 — well-understood algorithm
Post selector UI performance with 10k+ posts	MEDIUM	Use async Select2 with server-side search endpoint; paginate at 20 items
30+ ACF field types — coverage gaps	MEDIUM	Phase 1 covers 18 most common types; exotic types (clone, accordion) documented as known gaps
Multisite compatibility	LOW	Explicitly out of scope for Phase 1; documented in README
Cache invalidation for API consumers	LOW	Phase 2 feature — webhook/ping on post save actions

6. Future Phases Roadmap

Phase 2 — Authentication, Performance & Webhooks

Target: 6–8 weeks after Phase 1 release.

- JWT Bearer token authentication for user-context requests (login, profile, protected content)
- Role-based access control — expose different content to different API consumers
- Relational field dataloader to eliminate N+1 queries
- ACF depth setting up to 3 levels of nested resolution
- Webhook system — ping configured URLs on post publish, update, delete
- WooCommerce support — products, variations, orders (if WC active)
- Response caching layer with configurable TTL (transients or Redis if available)
- Rate limiting per API key
- Per-field ACF/SCF visibility control (expose/hide individual fields)

Phase 3 — GraphQL Layer

Target: 3–4 months after Phase 1.

- GraphQL endpoint at `/wp-json/atlas/graphql`
- Extends WPGraphQL if present, or runs standalone schema
- All Phase 1 + Phase 2 content types available as GraphQL types
- Fragments for ACF field groups — auto-generated from ACF field group config
- Subscriptions (WebSocket) for real-time content updates
- GraphQL persisted queries for performance

Phase 4 — Developer Experience & Ecosystem

Target: 6 months after Phase 1.

- OpenAPI 3.0 spec auto-generated from registered endpoints — live Swagger UI in admin
- WordPress CLI commands: `wp whc flush-cache`, `wp whc list-endpoints`, `wp whc test-key`
- SDK packages: JavaScript/TypeScript client, PHP client
- Preview mode — draft content API with time-limited preview tokens
- i18n/multilingual support — WPML and Polylang integration
- Multisite support with per-site settings inheritance
- Plugin compatibility layer — Gravity Forms, WPForms, ACF Blocks

Phase 5 — Analytics & Monitoring

Target: Post Phase 4.

- Built-in API request analytics dashboard in wp-admin
- Per-endpoint latency tracking

- API key usage stats — requests per day, top endpoints, error rates
- Alerting — email/Slack notification when error rate exceeds threshold
- Content delivery insights — which posts/pages are most requested via API

7. Phase 1 — Build Plan & Milestones

Week	Milestone	Deliverables
1	Plugin scaffold & architecture	Directory structure, bootstrap file, autoloader, settings schema, REST namespace registration
2	Frontend disabler + REST pruning	Theme disabler, REST endpoint filter with Gutenberg whitelist, API key auth middleware
3	Core content endpoints	Posts, Pages, CPT endpoints with filtering, pagination, and response envelope
4	Taxonomies, Media, Menus	Categories, all taxonomies, media endpoint, menu tree builder
5	ACF / SCF resolver	FieldResolver class, all 18 field types, relational resolution, gallery handling
6	Admin settings panel	All 5 tabs, Select2 async selectors, API key management UI, settings save/read
7	Testing & hardening	PHPUnit tests, Postman collection, Gutenberg compatibility check, edge case hardening
8	Docs & release	README, inline PHPDoc, changelog, GitHub release, plugin zip

8. Decision Log

Record key architectural and product decisions here for future reference.

Date	Decision	Rationale	Alternatives Considered
16 Apr 2026	REST for Phase 1 (not GraphQL)	Native to WP, no extra deps, every frontend dev knows it, easier to version & cache	GraphQL (WPGraphQL) — deferred to Phase 3
16 Apr 2026	API Key auth for Phase 1	No external dependency, sufficient for server-to-server headless setups	JWT (Phase 2), Application Passwords (limited, not revocable per-key)
16 Apr 2026	ACF depth limited to 1 level	Prevents N+1 query explosion; manageable in Phase 1 scope	Unlimited depth (performance risk), no relational resolution (DX poor)
16 Apr 2026	Multisite excluded from Phase 1	Adds significant namespace and options complexity; small user base	Include from start (scope creep risk)

9. Changelog

v1.0.0 — Phase 1 Release (TBD)

- Initial release
- Frontend disabler with Gutenberg-safe REST route pruning
- Custom REST namespace: /wp-json/atlas/v1/
- Endpoints: Posts, Pages, CPTs, Categories, Taxonomies, Media, Menus
- ACF/SCF field resolution for 18 field types
- API key authentication
- Admin settings panel — 5 tabs
- Per-content-type access control: All / Only These / Exclude These

This document is a living reference. Update the Decision Log and Changelog sections with every significant change. Questions or additions? Add a comment inline and sync with the lead developer.