Overview

What Waypoint Optimizer does

Waypoint Optimizer is designed for organisations whose technicians **travel to different locations to resolve tickets**. The technician selects the tickets to visit in the mobile app, and the system calculates the most efficient order of stops. From that moment on, every journey and every intervention is measured and stored: how far the technician actually travelled, how long each trip took, how long the work at each site lasted, and how much of that is billable.

The result is a field operation that can be planned, audited and invoiced with real data instead of estimates.

Key benefits

- **Optimised route planning:** select the tickets and the system calculates the most efficient order of stops using the Google Routes API.
- **Automatic time tracking:** travel time and on-site working time are recorded separately, with no paperwork for the technician.
- **Real distance vs. estimate:** every stop stores both what Google predicted and what the technician actually drove, so deviations are visible.
- **Live visibility for supervisors:** follow the team's position and active routes from the GLPI web interface.
- **Full audit trail:** every route is versioned. Changes made on the road never overwrite the original plan.
- **Data export:** download all journey data as CSV, with per-technician and per-day totals, for invoicing or analysis.

How it works: the model

Understanding four concepts is enough to use the plugin correctly.

2.1 A route contains several tickets

A **route** is a day's itinerary: one technician, several stops. Each **stop** corresponds to one GLPI ticket that has a location assigned. The plugin decides the best order in which to visit them.

A route is not tied to a single ticket — it groups all the tickets the technician plans to visit in that journey.

2.2 Two different timers

The plugin measures two things that must not be confused:

- **Waypoint (travel):** the trip from where the technician is to the ticket's location. Starts when the technician taps *Start* before setting off, and stops on arrival.
- **ActualTime (work):** the time spent actually resolving the ticket on site. It starts **automatically** the moment the technician stops the travel timer, and is recorded against the ticket through the ActualTime plugin.

> **NOTE**
> The two never run at the same time: stopping the travel timer is what starts the work timer.

2.3 Route versions

A route can be modified while it is in progress — adding a stop, removing one, or reordering. Each modification creates a **new version** of the route instead of overwriting it. The web interface keeps the complete version history, so it is always possible to see what was originally planned and what actually happened.

2.4 Four distance and time metrics

Each stop stores several measurements. They answer different questions and are **not interchangeable** — this is the part most worth understanding before reading any report.

Metric	What it measures	Where it comes from
Actual	What the technician really travelled on that leg.	Measured between the GPS position where the trip started and where it ended.
Leg planned	What Google predicted for *that single leg*, when the trip started.	Google, calculated at the moment the technician sets off.
Route planned	What Google predicted for the *whole optimised route*. Identical for every stop of the same route.	Google, calculated once when the route is created.
Billing	What is charged to the customer for the trip. **Not a measurement of the journey.**	Distance from the base to the ticket (doubled if return-trip billing is enabled).

Why *Actual* and *Leg planned* can differ a lot: the estimate assumes a vehicle following the road network from the previous point, while the real measurement reflects what actually happened — a short walk between two nearby sites, a detour, or traffic. A large gap is information, not necessarily an error.

Why *Billing* is separate: it always measures base → ticket, regardless of the order in which stops were visited. This keeps invoicing predictable and independent of the route the technician happened to follow that day.

Requirements

Before installing, check every point in this list. The plugin will not work if any of them is missing.

- **GLPI 10.0.0 – 10.1.0.**
- **Required plugins:** `gappextended` (2.21.0 or later) and `actualtime`. Both must be installed *and enabled*.
- **Google API key** with the Routes API and Distance Matrix API enabled. Without a valid key **no route can be calculated** and distances will be empty.
- **The mobile app,** installed on the technicians' devices. Route creation and execution happen in the app; the GLPI web interface is used for configuration, supervision and reporting.
- **Technician accounts must be central users** (not self-service) for the Waypoint section to appear in the app.
- **Location permissions** granted to the app, including background location, so tracking continues while the phone is locked.

> **IMPORTANT**
> **Waypoint depends on `gappextended`.** If `gappextended` is disabled, Waypoint is disabled with it — a cascading failure worth remembering when troubleshooting.