Scheduling Recurring Visits with Calendar Dates, Blackouts and Depots

MCP Server · https://www.easyterritory.com/documentation/ezt-mcp/scheduling-dates-blackouts-depots/

`schedule_visits` turns each account's service cadence into a day-by-day plan. Every day is split into tight geographic work clusters ("buckets"), and each bucket is one technician-day kept under an hour cap you set. This page covers the calendar features: start dates, blackouts, fixed appointments, due dates, and weekday rules. It also covers anchoring each day to a technician's home and a shared depot.

> Scheduling decides **which day** each visit happens. Routing (`calculate_route`) decides **what order** to drive one day's stops. Use both.

What you need

Input	Tell the agent	Notes
Accounts with a cadence column	"Visit frequency is in `service_interval_days`"	Declare it as the visit-frequency column when you load the accounts. Say whether the number is *days between visits* or *visits across the horizon*, and which values mean "never visit" (by default 0, empty, or null).
Onsite (dwell) time	"45 minutes per visit", or "use the `onsite_minutes` column"	Required. The server never assumes a default.
Daily capacity	"8-hour days, hard cap", or "aim for 7 hours, ±10%"	Required. Use a hard ceiling (`not_to_exceed`) or a target band.
Horizon	"Two weeks", or "derive it from the longest cadence"	Optional. Defaults to the longest cadence.

Input	Tell the agent	Notes
Start date	"Day 1 is Monday, November 2, 2026"	Required whenever you use dates. Without a start date, days are numbered 1, 2, 3... instead of being calendar dates. The server won't assume today.

Before building a visit schedule, you can ask for an estimate of the daily hours and number of people needed. If the schedule can't fit, you get a capacity plan with suggested fixes, such as more people per day or a longer workday.

Calendar rules (`visit_constraints`)

Each rule can apply to one account, a list of accounts, or all accounts (`*`). Mix as many as you need.

Rule	What it means	Ways to state it	Example
Blackout (or no-service window)	The account can't be visited	A single date, a list of dates, a date range, a range that **repeats every year**, or weekdays that repeat **every week**	"No visits to store 337 from Nov 9 to 13 (inventory)." "Never on Veterans Day, every year." "Never on Sundays."
Fixed	A hard appointment on one date	One date	"Inspection at site 184 on Nov 5."
Due	Must be visited by a date, optionally not before another date	Due date, optional earliest date; *penalized* (default) or *hard*	"Site 103 between Nov 10 and Nov 12, hard deadline."
Anchor	Fixes when the recurring pattern lands	A date or a weekday	"Site 1383 is always serviced on Tuesdays."
Allowed weekdays	Days the account can be visited	A list of weekdays	"Weekdays only for site 213."

How the rules are handled:

- **Blackouts are treated as hard rules.**
- **A fixed appointment counts toward the cadence.** It replaces a regular visit; it isn't added on

top.
- **Penalized due dates** are scheduled as early as possible. Any miss is listed by account with the number of days late. Mark a deadline *hard* if it can't slip, for example for a regulatory inspection.
- **Conflicts stop the run.** If rules can't all be met (say, a fixed date inside a blackout), the schedule fails with a date-conflict error instead of quietly dropping a rule. Decide which rule moves, then run it again.

Seasonal windows. To allow visits only in certain months (for example, "spring and fall only"), black out the other months with a range that repeats every year. A due date with an earliest date also works for a one-off window such as "between March and June 2027".

Anchoring days to technicians and a depot

Without technicians, buckets are unanchored day-clusters, and you can cap how many run per day. With technicians:

- Each technician's home or start location becomes the start of their buckets.
- Add a **pre-service stop**, such as a parts depot with its own onsite minutes, that every occupied day visits before the first account.
- Choose how the day ends: back home (loop) or at a set end point.
- Travel from home to the depot to the first account is included in that day's hours, so the cap reflects the real day.

Tell the agent: "*Use the three technicians in the* `technicians` *layer. Each day starts at home, stops at the Central Supply depot for 15 minutes, then returns home.*"

Worked example (sample data)

A two-week HVAC preventive-maintenance book in Houston:

- 280 accounts, mostly every 14 days, a few weekly, and one on hold
- 3 technicians and one shared depot (15 minutes)
- 45 minutes onsite and an 8-hour hard cap
- Day 1 is Monday, November 2, 2026

- Rules: an inventory blackout, a yearly Veterans Day blackout, no Sundays for one site, no Saturdays for another, one fixed inspection, one hard due window, one Tuesday anchor, and one weekdays-only site

The result:
- 42 buckets (3 technicians × 14 days), each holding 5–8 visits
- 279 accounts scheduled; the on-hold account was reported as excluded

Routing the three Day 1 buckets (home → depot → that day's calls → home) on Azure Maps gave 6.7 to 7.6 hours per technician, including onsite time.

In the same sample, a 300-account version didn't fit under 8 hours once every technician's depot stop was counted. That is the kind of capacity answer the tool is for.

What you get back

- **Per-bucket facts:** day, date (when you gave a start date), technician (when anchored), visits, and estimated hours.
- **A visits layer** (`<accounts>_visits`): one point per scheduled visit, with `schedule_day`, `schedule_date`, and `bucket_id`. An account visited twice appears twice.
- **A distribution summary:** hour spread across buckets, horizon fit, missed due dates, and excluded accounts.
- **A downloadable file** of every visit row.

Follow-ups that work in one step:
- "Color the visits by day."
- "Route bucket d01-b1 as a loop from Marcus's home through the depot." The agent filters the visits layer by `bucket_id`.

Schedule hours vs route hours

Bucket hours are a planning estimate: straight-line travel between that day's stops plus onsite time. When you route a day, `analyze_routes` reports road drive time, and the two numbers will differ. Use bucket hours to size the plan and route hours for the day itself. Never add them together.

Changing a schedule

Re-run with the same visits-layer name to replace a schedule, or use a new name to keep two side by side for comparison. There is no move-one-visit tool. Change the inputs and re-run.

What it doesn't do (yet)

- It doesn't decide how many technicians you need, or where to base them. Test alternatives by re-running with different technician lists or daily caps. Use `cluster_points` or `auto_build` with a workload target to estimate headcount.
- It doesn't model overnight travel or hotel relocation.
- It doesn't export to a specific work-order system. Use the visit file or GeoJSON.

Next steps

- Routing and Managing Multiple Routes
- How EZT Measures Travel
- Tools Reference → `schedule_visits`, `calculate_route`, `analyze_routes`