MCP Server Tools Reference

MCP Server · https://www.easyterritory.com/documentation/ezt-mcp/mcp-server-tools-reference/

The MCP Server exposes **58 customer-facing tools**. Confirmed against the public tool list at publish time. Tools are organized into two tiers: **Tier 1 goal tools** (prefer these — they accomplish a complete user goal in one call) and **Tier 2 primitives** (compose for edge cases and lower-level operations).

Call `get_guidance` at the start of a session. It returns the operating guide — the cross-tool workflow order and the shared rules every tool follows — plus `server_version` and a `seat` block for your key with your `plan` and the key's expiry (`key_expires_at`). When your key has an enforced quota, the seat block also carries `quota_remaining` , the remaining allowance for geocoding, routing, and isochrone requests. There is no session token or handle to pass: every call authenticates with your API key alone.

A Territory Solution (TS) is a self-contained **GeoJSON FeatureCollection**. That is the only project format. Save with `export_geojson` and reopen with `import_geojson` . There is no compact TS v2 format and no `export_ts` tool — a leftover compact body is rejected as invalid.

Every tool that does work returns the same envelope: `ok` , a `result` when the call succeeded, an `error` when it did not, and `warnings` for anything worth seeing before acting on the result. An error carries a code, a message, and usually `blocked_by` and a `do_this_next` naming the tool to call instead. Read the envelope, not the HTTP status: a call that fails still answers 200 with `ok: false` . The two task-status tools are the exception — `tasks_get` and `tasks_cancel` return the task status document itself.

Session Entry and Routing

Tool	Tier	Description

`get_guidance`	1	Returns the operating guide — cross-tool workflow order and shared rules — plus server_version and the caller's seat (plan and key expiry); issued keys on an enforced quota also carry remaining quota. Read-only. Call at the start of a session.
`discover_intent`	1	Deterministic router that classifies a user request into a workflow category — returns intent, tool order, clarifying questions, and inlined guidance.
`workflow_advisor`	1	Validated next-action advisor for multi-step workflows. Pass intent + state, get the correct next tool call. Re-call after each step until done.
`ezt`	1	Plain-language orchestrator for one-step completion when the granular tool is unclear. Prefer granular tools when you know the step.

Map and Data

Tool	Tier	Description
`get_map_visualization`	1	Opens or reuses the Map Component for a Territory Solution. Returns a shareable map_url. First step of all non-headless work. Omit arguments to reuse the current per-user workspace; pass `new_project=true` to wipe it in place and start blank (same session URL).
`ensure_map_viewer`	1	Confirms a live viewer is connected before pushing compute or selection state. Poll until connected.
`show_map_overlay`	1	Shows or adds a data layer on the map — ZIPs, counties, accounts, territories, or a route already drawn by calculate_route. Takes natural language user_request. Prefer this over configure_map for show/add asks.
`configure_map`	2	Patches map config: loaded part layers, active TAL, point classifications (color/size/shape), presentation, and optional camera (`center` + `zoom`). Also handles point symbology via point_layer_classifications. Omitting camera fields leaves the viewport alone.
`set_map_state`	2	Low-level MC mode, active TAL, or session-only camera jump. Does not write the TS. Prefer configure_map for durable config.

`delete_point_layer`	2	Removes one point layer (for example a stale account or contractor layer) from the Territory Solution and map.
`geocode_address`	1	Geocode-only pass: turns staged addresses into coordinates with match confidence, without loading a map layer. Stage the addresses first with `request_account_upload`, then pass its `upload_handle`; the handle is not claimed, so `ingest_accounts` can load the same rows afterward. Accepts address, city, state, and postal columns, and maps common source headers. Cached addresses are free; billable lookups count against a monthly per-key quota. Up to 10,000 rows per call. The result is a summary, 10 sample rows, the first 50 failures, and a downloadable artifact holding every geocoded row — not the rows inline.
`ingest_accounts`	1	Account ingest: loads staged account rows as a point layer on the Territory Solution, geocoding rows without coordinates. Rows arrive only as an `accounts_handle` from `request_account_upload`, which stages them from the user's file, `csv_text`, or rows (100,000-row cap) and suggests these arguments. Declares the metric, workload, dwell, visit-frequency, grouping, display, and search columns; undeclared columns are dropped. Use `layer_role: contractors` for a contractor layer with trade, tier, capacity, and service-ZIP columns (used by `analyze_network_coverage`).
`request_account_upload`	2	The one way account or address rows enter the server. Stages them for `ingest_accounts` or `geocode_address` and returns `upload_handle`, the CSV headers, and suggested ingest arguments with a geocode plan. Pass that `upload_handle` as `accounts_handle` to `ingest_accounts`, or as `upload_handle` to `geocode_address`. Four ways in: an `upload_url` the user opens in a browser to drop a CSV or .xlsx, which needs no API key (`result.share_with_user`) and which a shell can POST to instead; `csv_file`; `csv_text` up to 25 MB per call; or row chunks appended with the handle. The browser page accepts files up to 25 MB and previews them, with warnings, before anything is written. One handle holds up to 100,000 rows, is single-use, and expires in 1 hour.

`list_dataverse_views`	2	Microsoft Copilot only (Entra ID sign-in). Lists the Dynamics 365 environments, then the account views the signed-in user can read, with a `point_readiness` sample (coordinates or address-only). Other clients get `DATAVERSE_NOT_AVAILABLE`; use `request_account_upload` instead.
`import_dataverse_view`	2	Microsoft Copilot only. Reads one account view server to server, as the signed-in user, and stages it like an account upload (`upload_handle`, preview, `suggested_ingest`). No rows pass through the agent. Adds the account id, name, and address columns. Up to 100,000 accounts per view. Confirm column roles, then call `ingest_accounts`.
`publish_to_dataverse`	2	Microsoft Copilot (Entra ID sign-in). Writes results stamped on a point layer, such as territory, route, schedule, or group, back to the matching Dynamics 365 account records. The first call previews the changes and the account fields to fill; nothing is written until you confirm.
`list_part_layers`	2	Lists the boundary layers you can build territories from — ZIPs, counties, Canadian FSAs, and each active country's postal or administrative areas — with the part count, ID format, and example parts for each. Ask for one layer by name to see its coverage and data vintage, or narrow the list to one country. Read-only.
`query_parts`	2	Retrieves part metadata only (no geometry). Takes exactly one of `filter` (e.g. `{state_abbr: "TX"}`) or `part_ids`; neither or both is rejected. A `filter.territory` of `{tal_id, territory_id}` with `ts_handle` or `map_session_id` lists one territory's parts. Filter, enrich, or count parts before a build.

You can switch the map between a dark and a light look, and the change shows up on the live map right away.

Territory Building

Tool	Tier	Description

`auto_build`	1	Builds N balanced territories from loaded accounts. Modes: fixed_territory_count, fixed_workload_target, scoped_split. Objective blends workload and metrics via bias weights. Named-state builds use `part_scope=explicit` plus `part_filter`.
`account_build`	1	Builds territories that mirror an existing CRM grouping field (rep name, territory name, territory code). Conflicts resolved by plurality or configurable policy.
`cluster_points`	1	Balanced point grouping with the same build_mode, objective, dwell, and visit-frequency as auto_build. Output is a group_field on points (no TAL, no geography dissolve). Add seed_point_layer to anchor groups to start locations.
`direct_build`	1	Creates a TAL from known part-to-territory assignments — a ZIP-to-territory sheet or file, including a three-level region/district/territory hierarchy. Assignments arrive only as an `assignments_handle` staged by `request_assignment_upload`; there is no inline assignment list. Duplicate part IDs follow `duplicate_part_policy` (`keep_first` by default).
`request_assignment_upload`	2	The one way part-to-territory assignment rows enter the server. Stages a ZIP-to-territory spreadsheet (Postal Code plus Territory/Region/Division) from a browser upload link, `csv_file`, `csv_text`, or appended row chunks, and returns the handle that `direct_build` takes as `assignments_handle`. Legacy column names map to `part_id` and `territory_path` automatically. A file of locations rather than assignments is redirected to `request_account_upload`. Single-use; expires after 1 hour.
`isochrone_build`	1	Generates drive-time or drive-distance areas from one or more origins. Origins times budget bands in one call. A reachability area, not a balanced partition. Limits per call: 25 origins, 5 bands, up to 6 hours or 500 km.

`seed_build`	1	Grows one territory outward from one seed location until a location count or a declared metric sum is as close as possible. Appends a leaf to an existing layer, or starts a new layer. Does not overlap territories already on that layer. One seed per call. Geocode an address first.

Territory Management

Tool	Tier	Description
`realign`	1	Moves parts between leaf territories within one TAL. Supports individual moves, remove_parts, and remove_empty_territories. Followed by geometry re-dissolve and repair.
`request_part_selection`	2	Puts the open map into part-selection mode for a realign, a new territory, or a plain `return_list`. `map_session_id` selects on the open map; `ts_handle` opens a stored Territory Solution first. The person selects parts on the map and the commit happens there, not in chat; the agent polls `get_part_selection` until it reports `committed`.
`get_part_selection`	2	Polls or retrieves committed part IDs after a selection task. Poll until status=committed, then proceed.
`get_map_selection`	2	Reads the latest committed MC selection — use when the user started selection from the legend without a prior request_part_selection.
`create_territory_from_parts`	2	Creates or updates one leaf territory from committed part IDs. Supports conflict_policy=move_from_existing to take parts from another territory.
`delete_territory`	1	Deletes one leaf territory from a TAL. Collects parts and runs Realign remove_parts internally. To wipe an entire TAL, use delete_tal.
`delete_tal`	1	Wipes one entire TAL from the TS and open map in one synchronous call. Preserves points, part overlays, and routes. Never call delete_territory N times to clear a layer.

Restructure (Split / Merge / Rebalance)

Tool	Tier	Description
`territory_split`	1	Splits one oversized territory with minimal disruption to the rest of the alignment. Distinct from auto_build scoped_split (balanced from scratch).
`territory_merge`	1	Merges two adjacent territories and rebalances with minimal disruption.
`territory_rebalance`	1	Adjusts boundaries to restore balance after data changes. Source assignment is the non-regression baseline — no change if no candidate improves the objective.

Analysis

Tool	Tier	Description
`analyze`	1	Computes balance, workload, and compactness metrics for territories. Re-run after any TAL or point change. Supports cross-TAL comparison and hypothetical moves. Prefer `analysis_panel=single` when a map is open. Accepts a completed analyze `task_id` on follow-up display.
`analyze_routes`	2	Returns per-route facts (drive hours, dwell hours, route workload, stop coordinates) for routes already drawn by calculate_route. Facts only — no capacity or ranking. Never sum route hours with territory workload.
`analyze_network_coverage`	1	Service Network Coverage. For one trade per run (optional tier filter), counts how many qualified contractors can reach each demand point by a drive-time limit or declared service ZIPs, with an optional road-network check using drive-time areas (up to 25 contractors per call). Returns color-coded coverage and gap layers, coverage percentage, single-contractor counts, gap priority, and per-contractor reach. Read-only: never changes source layers. See Service Network Coverage.

`points_in_territory`	1	Lists every point inside one, several, or all territories or drive-time areas, with full properties, counts, and metric totals. Default list size 1,000, maximum 10,000 per territory; counts and totals always cover every point (check `truncated`). point_filter (e.g. {tier: "Gold"}) filters points by attribute before counting and listing, and overlaps lists the points falling in more than one territory.
`load_analysis_panel`	2	Renders Analyze JSON in the Map Component bottom dock. Modes: single, compare, diagnostic, executive. Accepts a completed analyze task_id. Session-only display; does not mutate the TS.

Routing and Scheduling

Tool	Tier	Description
`calculate_route`	1	Drives a known list of stops in the best sequence. Ordered stop_sets, stop-priority banding, and a route type you choose: circuit, tour (needs an end stop), or open_tour. Returns road distances, drive times, and map geometry. Reuse an existing route_id to replace that route; a new route_id adds another.
`schedule_visits`	1	Expands recurring account cadence across a repeating horizon into daily work clusters under a technician-day workload cap. No TAL. Pairs with calculate_route for per-day routing. Calendar rules (`visit_constraints`): start date, blackout dates and ranges (including yearly and weekly repeats), fixed dates, due dates, anchors, and allowed weekdays; conflicts are reported, not dropped. Technician and depot anchoring: `contractors` (each technician's start), `pre_service_stops` (shared depot), and `end`; depot travel counts against the day's hours. See Scheduling with Dates, Blackouts and Depots.
`delete_route`	2	Drops a whole route from the TS and map. To change stops, call calculate_route again with the same route_id and the revised stop list.

Import and Export

Tool	Tier	Description

`export_geojson`	2	The project save. Async job: materializes the Territory Solution as a self-contained GeoJSON FeatureCollection with download_url and a share link that opens without a key for one hour. Omit filters for the full project. Gzip is compression, not a second format.
`import_geojson`	2	The project reopen. Opens a GeoJSON FeatureCollection — points, or an EZT-exported territory file, plain or gzip/zip — as a Territory Solution and returns a ts_handle. Files of real size move by reference: call with no file argument for an upload_url, then poll with upload_handle until received. Inline `geojson`, `geojson_text`, and `geojson_gzip` are capped at 256 KB; anything larger returns `INLINE_PAYLOAD_TOO_LARGE` with the `upload_url` call to make instead. Arbitrary territory polygons need part_layer + overlay_policy='centroid_within' to assign parts by centroid. Compact TS v2 is not accepted. A Territory Designer project comes in through import_designer_project instead (up to 250,000 point features); see Pulling and Pushing Territory Designer Projects.
`link_designer`	2	Links the Designer account to the MCP Server once. Returns a link the user opens to paste their Designer agent token; the token is stored encrypted server-side and never shared in chat. Poll until status is linked.
`push_to_designer`	2	Saves a territory solution as a Designer project — a new project when no project_id is given (ask for the label first), an in-place update when one is. Updates add and change only, skip locked markup, and never delete; a point layer already in the project has its rows replaced by the pushed rows, which is the irreversible part. Cannot be undone: confirm before running one.
`list_designer_projects`	2	Lists the Designer projects that user can see — project_id, label, tags, saved_at, is_mine, layers. Optional search, tag filter, mine-only.
`import_designer_project`	2	Pulls an existing Designer project back as a territory solution you can keep working on. Hand-drawn polygons with no part IDs cannot be imported; option to import with those excluded.

Delegation

Tool	Tier	Description
`extract_tal_branch`	2	Extracts a regional subtree for bounded delegate editing. Senior planner sends a branch to a delegate.
`reintegrate_branch`	2	Merges an approved proposal TS back into master. Sequential merges only; STALE_TS_REVISION if master moved.

Async Task Management

Tool	Tier	Description
`tasks_get`	2	Polls status of an async task. Follow next_action exactly: sleep_and_poll → one more poll; consume_result → call tasks_result once. status=cancelled is a user stop — do not resubmit unless asked.
`tasks_result`	2	Consumes the terminal result of a completed task. Call exactly once after tasks_get returns completed. Returns handle-only result (never inline TS).
`tasks_cancel`	2	Cooperatively cancels a running async task. Confirm TS unchanged or partial per the originating tool contract. Do not treat cancel as a retry signal.

Knowledge and Feedback

Tool	Tier	Description
`ep_search`	2	Semantic search over the EasyTerritory ExpertPack (EMEP) for workflow guidance, concepts, and common mistakes.
`ep_list_topics`	2	Lists EMEP topics by type (concept, workflow, interface, troubleshooting, decision).
`ep_graph_traverse`	2	Walks related EMEP guidance from a topic file.
`submit_feedback`	2	Logs a quality issue when workflow is blocked, partial, or a workaround was required. Category, severity, and context captured.

Tool Tiers Explained

Tier 1 — Goal Tools

Tier 1 tools accomplish a complete user goal in one call. Prefer these. Each links to an owning workflow that defines prerequisites, next steps, and common mistakes. Tier 1 tools include routing (`get_guidance`, `discover_intent`, `workflow_advisor`, `ezt`), map and data (`get_map_visualization`, `ensure_map_viewer`, `show_map_overlay`, `geocode_address`, `ingest_accounts`), building (`auto_build`, `account_build`, `cluster_points`, `direct_build`, `isochrone_build`, `seed_build`), management (`realign`, `delete_territory`, `delete_tal`), analysis (`analyze`), routing (`calculate_route`), and scheduling (`schedule_visits`). Restructure tools (`territory_split`, `territory_merge`, `territory_rebalance`) are goal-level for those specific workflows.

Tier 2 — Primitives

Tier 2 tools compose for edge cases and lower-level operations. Use them to support a Tier 1 goal, not in place of one. Includes map configuration (`configure_map`, `set_map_state`), data staging (`request_account_upload`, `request_assignment_upload`, `list_part_layers`, `query_parts`), selection (`request_part_selection`, `get_part_selection`, `get_map_selection`, `create_territory_from_parts`), analysis display (`analyze_routes`, `load_analysis_panel`), route deletion (`delete_route`), import/export (`export_geojson`, `import_geojson`), Designer roundtrip (`link_designer`, `push_to_designer`, `list_designer_projects`, `import_designer_project`), delegation (`extract_tal_branch`, `reintegrate_branch`), async tasks (`tasks_get`, `tasks_result`, `tasks_cancel`), knowledge (`ep_search`, `ep_list_topics`, `ep_graph_traverse`), and feedback (`submit_feedback`).

Build Tool Disambiguation

These tools are not interchangeable:

- **auto_build** — balanced partition from points into geographic territories (N territories or a workload target)
- **account_build** — group by existing CRM attribute (rep name, territory code)
- **cluster_points** — balanced grouping column on points, no TAL
- **direct_build** — known part-to-territory file

- **schedule_visits** — recurring cadence, which day each visit happens (only one with a time dimension)
- **seed_build** — one territory grown from one location until a count or metric total is met
- **isochrone_build** — reachable area around fixed origins (a budget, not an objective — divides nothing, balances nothing)
- **calculate_route** — drive order for a known list of stops (creates no territories)
- **analyze_network_coverage** — reports contractor coverage and gaps for one trade (creates no territories, assigns nothing)
- **points_in_territory** — lists what is already inside a territory or drive-time area (builds nothing)

See also: Connecting an OAuth-Only Agent, Getting Started, Connecting Your AI Agent, Territory Building with AI, Territory Management & Realignment, Routing & Scheduling, FAQ.