

livespec

Superficie completa de tools MCP — v0.31.4. 45 tools: 28 en el núcleo siempre visible, 12 en el plugin **livespec-spec** (mutación de Specs) y 5 en el plugin **livespec-docs** (documentación y Explorer). Los plugins registran al boot pero **PluginVisibilityMiddleware** los oculta de *tools/list* hasta que el workspace los justifica — o hasta que se fuerza con `LIVESPEC_PLUGINS`. Todo tool exige workspace absoluto.

Indexado (1)

Construye y refresca todo el estado. Incremental por hash de contenido (xxh3); respeta .gitignore (raíz y anidados, con negaciones) y [index] de .livespec.toml.

```
index_project(force=False,
watch=False, explorer=False)
```

Recorre el workspace, parsea, persiste símbolos y aristas, reconstruye los chunks de búsqueda y auto-escanea las anotaciones @spec:. Es la única puerta de entrada al índice.

Búsqueda (1)

FTS5 sobre chunks AST-aware. Los vectores densos se removieron en v0.29: keyword search es el único carril.

```
search(query,
scope='all' | 'code' | 'specs',
limit=20)
```

Recupera código o Specs que **hablan de** algo cuando no hay un nombre exacto que buscar; divide snake_case en tokens OR.

Code intelligence (14)

Las preguntas que un agente hace de verdad sobre un repo ajeno. Todos aceptan limit / cursor / summary_only donde el payload puede crecer sin techo; los conteos siempre son exactos.

<code>find_symbol(query, kind, limit)</code>	Lookup de símbolo agnóstico al separador (punto, slash, doble dos-puntos): el primer salto desde un nombre a medias.
<code>get_symbol_source(qname)</code>	Devuelve solo el cuerpo del símbolo — más liviano que traer el archivo o la metadata completa.
<code>quick_orient(qname)</code>	Primer contacto canónico: metadata, lead del docstring, top-5 callers y callees por PageRank, Specs enlazadas y flag de entry point en una sola llamada.
<code>who_calls(qname, max_depth=1)</code>	Cono hacia atrás. Agrega route_callers cuando el símbolo es un endpoint HTTP que un frontend consume por ruta (cross-repo vía group_db).
<code>who_does_this_call(qname, max_depth=1)</code>	Cono hacia adelante. Agrega invokes_endpoints: las rutas de backend que este símbolo golpea vía fetch / axios / requests.
<code>analyze_impact(target_type, target, max_depth)</code>	Radio de impacto de un símbolo, archivo o Spec — incluye cascada por el grafo Spec→Spec. Con max_depth=1 cubre el viejo 'find references'.
<code>git_diff_impact(base_ref, head_ref, max_depth=5)</code>	Punto de entrada de revisión de PR: archivos cambiados → llamadores impactados → Specs afectadas → tests sugeridos.
<code>get_project_overview(include_infra structure=False)</code>	Los símbolos más centrales por PageRank, con el ruido de infraestructura filtrado por defecto.

<code>find_dead_code(include_infrastructure=False, corroborate_with=None)</code>	Candidatos sin llamadores ni links de Spec. Reporta <code>filtered_out</code> por flag; <code>corroborate_with</code> descarta los que un segundo extractor todavía ve referenciados.
<code>find_orphan_tests(max_depth=10, corroborate_with=None)</code>	Tests cuyo cono hacia adelante nunca alcanza un símbolo de producción — es decir, que probablemente no prueban nada.
<code>find_legacy_flows(project=None, include_infra_routes=False)</code>	Flujos HTTP probablemente muertos: servidores sin cliente indexado y clientes sin servidor que matchee. Mejor con <code>group_db</code> . Grafo, no tráfico.
<code>find_endpoints(framework=None)</code>	Entry points por framework: decoradores, routing por filesystem, bases de CBV de Django y routing call-style (Express, Hono). Devuelve <code>http_method</code> / <code>http_path</code> donde se puede.
<code>grep_in_indexed_files(pattern, path_glob, kind, limit=50)</code>	Grep acotado a los archivos que están en el índice — evita <code>node_modules</code> y <code>.venv</code> sin configurar <code>excludes</code> a mano.
<code>audit_coverage()</code>	Reporte de cobertura Spec: módulos sin Spec, cubiertos transitivamente, huérfanos o en lenguajes sin extractor; Specs sin implementación; y cobertura de tests derivada del cono de llamadas.

Spec agentic — consulta y bootstrap (5)

La mitad de lectura de la trazabilidad: siempre visible, porque son preguntas, no ceremonia.

<code>list_specs(status, module, priority, kind, has_implementation)</code>	Superficie de descubrimiento: qué Specs existen y cuáles tienen (o no) implementación.
<code>get_spec_implementation(spec_id)</code>	Responde '¿qué código implementa auth-user-login?' — con escenarios, símbolos enlazados por escenario y flag <code>verified</code> .
<code>propose_specs_from_codebase(module_depth=2, ...)</code>	Descubrimiento heurístico sobre un repo sin Specs: agrupa por módulo y PageRank (o por comunidad detectada con <code>community_graph</code>) y propone candidatos con título humanizado.
<code>bulk_link_spec_symbols(mappings)</code>	Enlaza N pares (<code>spec_id</code> , símbolo) en una transacción. Escape hatch para archivos donde la anotación en fuente no llega: configs, SQL, YAML. Idempotente.
<code>import_specs_from_markdown(path, fmt='openspec')</code>	Crea o actualiza Specs desde markdown OpenSpec — un archivo o un árbol entero. Siempre visible para poder arrancar brownfield sin tocar env vars.

Interop OpenSpec (5)

Round-trip completo con el formato de Fission-AI: `livespec` no reemplaza la autoría de specs, la enlaza al código.

<code>sync_openspec(openspec_dir=None)</code>	Importa el árbol OpenSpec entero de una: requirements canónicos de specs/ más cada change en changes/ y archive/.
<code>export_openspec(out_dir='openspec', include_changes=True)</code>	El inverso: escribe la DB de vuelta como specs//spec.md. Cierra el round-trip.
<code>validate_openspec(strict=False)</code>	Espeja <code>`openspec validate --strict`</code> ; el chequeo que carga peso es que todo requirement tenga al menos un escenario.

<code>list_spec_changes(status=None)</code>	Lista las propuestas de cambio con su estado (proposed / archived).
<code>get_spec_change(name)</code>	Inspecciona una propuesta: prosa de proposal / design / tasks más los deltas ADD / MODIFY / REMOVE / RENAME.

Higiene y descubribilidad (2)

Dos tools que existen porque lo invisible es peor que lo ausente.

<code>scan_annotation_verbs(sample_per_group=10)</code>	Saca a la luz comentarios con forma de anotación (@word:) que el matcher NO va a enlazar, separando verbo no reconocido (con did_you_mean) de payload no linkeable.
<code>get_cross_repo_guide()</code>	El how-to de polyrepo (group_db, Specs xrepo-*, Flow Explorer, trampas) más un snapshot vivo del grupo, expuesto como tool porque algunos hosts cachean un resources/list viejo.

Plugin livespec-spec — mutación de Specs (12)

Lo que ejecuta un operador, no lo que pregunta un agente. Visible cuando el workspace ya tiene filas spec, o con LIVESPEC_PLUGINS=spec.

<code>create_spec(title, ...)</code>	Crea una Spec nueva en el store.
<code>update_spec(spec_id, ...)</code>	Modifica campos de una Spec existente (estado, prioridad, kind, módulo, texto).
<code>delete_spec(spec_id)</code>	Borra la Spec y, en cascada, sus links a símbolos.
<code>link_spec_symbol(spec_id, symbol_qname, relation, ...)</code>	Enlaza o desenlaza un par Spec↔símbolo, con relación, confianza y origen.
<code>link_scenario_symbol(spec_id, scenario_name, symbol_qname, ...)</code>	Trazabilidad a nivel escenario: ata código o tests a un `#### Scenario:` puntual, más fino que el requirement entero.
<code>link_spec_dependency(parent, child, kind='requires')</code>	Arista del grafo Spec→Spec (requires / extends / conflicts); los ciclos se rechazan en el insert.
<code>unlink_spec_dependency(parent, child)</code>	Quita una arista del grafo Spec→Spec.
<code>get_spec_dependency_graph(spec_id, ...)</code>	Camina el grafo Spec→Spec: qué depende de esto, transitivamente.
<code>scan_spec_annotations()</code>	Matcher de dos niveles sobre las anotaciones @spec: del código; corre solo tras cada index_project. Los ids que no existen vuelven como unknown_annotation_ids en vez de perderse.
<code>scan_docstrings_for_spec_hints()</code>	Propone candidatos a Spec desde los docstrings existentes (primera oración, verbo inicial) y devuelve el histograma de verbos dominantes.
<code>apply_spec_change(name, dry_run=False)</code>	Pliega los deltas de un change sobre el set canónico (upsert, deprecate, y RENAMED mueve la trazabilidad al nombre nuevo). Con dry_run devuelve plan y warnings sin mutar.
<code>archive_spec_change(name)</code>	Marca el change como archivado, cerrando su ciclo de vida.

Plugin livespec-docs — docs y Explorer (5)

Ceremonia de tier humano. Visible con filas doc, con un bundle .mcp-docs/explorer/ en disco, o con LIVESPEC_PLUGINS=docs.

<code>generate_docs(target_type, identifier, content=None, ...)</code>	Genera documentación en tres modos (caller_supplied, sampling, needs_caller_content): funciona tanto en Claude Code como en hosts con sampling.
<code>list_docs(target_type, only_stale=False)</code>	Lista docs o, con only_stale, los que driftearon — el drift dispara por body_hash O signature_hash.
<code>export_documentation(format, out_subdir)</code>	Vuelca la documentación a markdown o JSON.
<code>export_explorer(base=None, head=None, generated_at=None)</code>	Escribe el bundle estático del Spec Explorer (.mcp-docs/explorer/): vista tipo Swagger por Spec, Try-it HTTP y playground MCP al servirlo.
<code>export_flow_explorer(...)</code>	Bundle companion del Flow Explorer: los flujos HTTP cross-repo dibujados de cliente a handler.

Resources

project://overview · project://index/status · project://specs · project://specs/{spec_id} · project://files/{path*} · project://symbols/{qname*} · project://cross-repo · doc://symbol/{qname*} · doc://spec/{spec_id} · code://symbol/{qname*}

Prompts (slash commands)

agent_playbook (la guía principal) · openspec_workflow · onboard_project · analyze_change_impact · audit_spec_coverage · extract_specs_from_module · document_undocumented_symbols · refresh_stale_docs · explain_symbol

Contratos que no se rompen

workspace absoluto obligatorio en toda llamada (sin fallback a cwd ni env) · errores solo vía mcp_error(): {error, isError, did_you_mean?, hint?} · paginación (limit / cursor / summary_only) en agregadores, con conteos siempre exactos · migraciones append-only · el resolver de refs es INSERT OR IGNORE, nunca DELETE.