Package: tidybrregmcp (via r-universe)

November 3, 2026

Title Model Context Protocol Server Derived from 'tidybrreg'

Version 0.6.0

Description Exposes the exported functions of 'tidybrreg' as Model Context Protocol (MCP) tools. The tool catalogue, every tool's title, description, argument documentation and input schema are derived at run time from the installed 'tidybrreg' namespace and its help pages, so the server changes exactly when 'tidybrreg' changes and adds no interpretation of the register's data of its own: tools return the tibbles 'tidybrreg' returns, with its column names, its codes, its attributes and its typed conditions. Serves over standard streams or streamable HTTP.

License MIT + file LICENSE

URL https://sondreskarsten.github.io/tidybrreg/, https://github.com/sondreskarsten/tidybrreg

BugReports https://github.com/sondreskarsten/tidybrreg/issues

Depends R (>= 4.1.0)

Imports cli, jsonlite, rlang, tibble, tidybrreg (>= 0.6.0), tools, utils

Suggests callr, curl, httpuv, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.1

Config/pak/sysreqs libssl-dev

Repository https://sondreskarsten.r-universe.dev

Date/Publication 2026-11-02 18:57:27 UTC

RemoteUrl https://github.com/sondreskarsten/tidybrreg

RemoteRef HEAD

RemoteSha 55e87e08a95137ad97098a794e327be25ac6aa01

RemoteSubdir tidybrregmcp

Contents

mcp_handle 2
mcp_manifest 3
mcp_serve 4
mcp_server 5
mcp_tools 5

Index **7**

`mcp_handle` *Handle one JSON-RPC message*

Description

Pure dispatcher shared by every transport. Requests return a response object; notifications return `NULL`. Tool execution errors are reported inside the result (`isError`), protocol errors as JSON-RPC errors.

Usage

```
mcp_handle(srv, msg)
```

Arguments

`srv`	Server state from `mcp_server()`.
`msg`	A decoded JSON-RPC message (named list).

Value

A named list (the response) or `NULL`.

Examples

```
srv <- mcp_server()
mcp_handle(srv, list(jsonrpc = "2.0", id = 1, method = "ping"))
```

`mcp_manifest` *Write the tidybrreg MCP manifest*

Description

Serialises everything a server needs to expose tidybrreg over the Model Context Protocol without running R: the tool catalogue of `mcp_tools()` (titles, descriptions, input and output schemas), the machine-readable endpoint contract shipped by tidybrreg (`inst/extdata/endpoints.json`, tested against the functions), the bundled dictionaries with their vintages, every help topic as Markdown, and the server instructions. The output is deterministic for a given installed tidybrreg (no timestamps), so the Node implementation in `tidybrreg-mcp-node/` commits it and CI fails when regenerating it changes anything: the non-R server changes exactly when tidybrreg does.

Usage

```
mcp_manifest(
  path = "tidybrreg-mcp-manifest.json",
  package = "tidybrreg",
  exclude = character()
)
```

Arguments

`path`	File to write. `NULL` returns the manifest as a list without writing.
`package`	Package whose exports become tools. Only `"tidybrreg"` is supported; the argument exists so the derivation is visibly generic.
`exclude`	Character vector of export names not to expose. Defaults to the comma-separated `TIDYBRREG_MCP_EXCLUDE` environment variable.

Value

The manifest as a list, invisibly when `path` is given.

Examples

```
m <- mcp_manifest(NULL)
names(m)
length(m$tools)
```

`mcp_serve` *Serve tidybrreg over the Model Context Protocol*

Description

Starts an MCP server whose tools are tidybrreg's exported functions (see `mcp_tools()`). Two transports are available.

Usage

```
mcp_serve(
  transport = c("stdio", "http"),
  host = "0.0.0.0",
  port = as.integer(Sys.getenv("PORT", "8080")),
  token = Sys.getenv("MCP_BEARER_TOKEN", ""),
  srv = NULL
)
```

Arguments

`transport`	`"stdio"` (default) or `"http"`.
`host`, `port`	Bind address for the HTTP transport. `port` defaults to the `PORT` environment variable (as set by Cloud Run) or 8080.
`token`	Bearer token for the HTTP transport; `NULL` or `""` disables authentication.
`srv`	Server state; built with `mcp_server()` when `NULL`.

Value

`mcp_serve()` runs until the client closes the connection (stdio) or the process is interrupted (HTTP) and then returns `invisible(NULL)`.

stdio

`transport = "stdio"` reads newline-delimited JSON-RPC messages from standard input and writes responses to standard output, which is what desktop clients (Claude Desktop, Claude Code, Cursor, ...) expect from a locally launched server. Log lines go to standard error. A client configuration is:

```
{"mcpServers": {"tidybrreg": {"command": "Rscript",
  "args": ["-e", "tidybrregmcp::mcp_serve()"]}}}
```

Streamable HTTP

`transport = "http"` serves the streamable HTTP transport with httpuv: `POST /mcp` carries one JSON-RPC message (or a batch) and answers with `application/json`; `GET /mcp` answers 405 because this server never pushes messages; `DELETE /mcp` ends a session; `GET /health` reports the server and the tidybrreg version. When the `MCP_BEARER_TOKEN` environment variable (or `token`) is set, every `/mcp` request must carry it as `Authorization: Bearer`.

Examples

```
## Not run:
tidybrregmcp::mcp_serve()
tidybrregmcp::mcp_serve("http", port = 8080)

## End(Not run)
```

`mcp_server` *Create the MCP server state*

Description

Builds the tool catalogue and resource index from the installed tidybrreg once; transports call `mcp_handle()` with it for every message.

Usage

```
mcp_server(package = "tidybrreg", exclude = mcp_excluded())
```

Arguments

`package`	Package whose exports become tools. Only `"tidybrreg"` is supported; the argument exists so the derivation is visibly generic.
`exclude`	Character vector of export names not to expose. Defaults to the comma-separated `TIDYBRREG_MCP_EXCLUDE` environment variable.

Value

An environment holding `tools`, `resources`, `info`, `instructions`, and `initialized`.

Examples

```
srv <- mcp_server()
length(srv$tools)
```

`mcp_tools` *MCP tool definitions derived from the tidybrreg namespace*

Description

Every exported function of tidybrreg becomes one tool. The tool's title, description and argument documentation are rendered from the function's installed help page, the input schema is inferred from the function's formals together with the documented argument types, and nothing is hand-written per tool. A change to tidybrreg's exports or documentation changes the catalogue the next time the server starts.

Usage

```
mcp_tools(package = "tidybrreg", exclude = mcp_excluded())
```

Arguments

package	Package whose exports become tools. Only "tidybrreg" is supported; the argument exists so the derivation is visibly generic.
exclude	Character vector of export names not to expose. Defaults to the comma-separated TIDYBRREG_MCP_EXCLUDE environment variable.

Value

A named list of tool definitions, each a list with name, title, description, inputSchema, outputSchema, annotations, and (not sent to clients) fn and arg_info.

Output shaping

Four optional arguments that tidybrreg does not have are added to every tool by the server, all prefixed with a dot so they can never collide with a tidybrreg argument:

- .limit: maximum rows returned (default 200, at most getOption("tidybrregmcp.max_rows", 5000)).
- .offset: rows to skip before returning (default 0).
- .columns: column names to keep, in tidybrreg's English names.
- .format: "records" (default, JSON objects), "csv", or "markdown" for the text rendering of the rows. The structured result always reports the total row count so a caller knows when it saw a slice.

Examples

```
tools <- mcp_tools()
names(tools)
tools$brreg_entity$inputSchema$properties$registry
```

Index

mcp_handle, 2
mcp_handle(), *5*
mcp_manifest, 3
mcp_serve, 4
mcp_server, 5
mcp_server(), *2*, *4*
mcp_tools, 5
mcp_tools(), *3*, *4*