Package 'mcplite'

August 3, 2026

Title Lightweight Stdio MCP Server for R

Version 0.1.0

Description Provides a lightweight Model Context Protocol (MCP) server for exposing 'R' functions as tools over standard input and output ('stdio'). Designed for local, client-launched integrations, with protocol-aware tool definitions and results, JSON Schema helpers, and optional interoperability with 'ellmer'.

License MIT + file LICENSE

URL https://tosidata.github.io/mcplite/, https://github.com/tosidata/mcplite

BugReports https://github.com/tosidata/mcplite/issues

Depends R (>= 4.1.0)

Imports jsonlite, nanonext (>= 1.6.0), otel

Suggests ellmer (>= 0.3.0), otelsdk, processx, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Encoding UTF-8

Config/roxygen2/version 8.0.0

Config/Needs/website pkgdown, rmarkdown

NeedsCompilation no

Author Juha Itkonen [aut, cre, cph]

Maintainer Juha Itkonen <juha@tosidata.com>

Repository CRAN

Date/Publication 2026-08-03 18:10:13 UTC

Contents

content-blocks 2
mcp_server 3

tool 5
tool-types 6
tool_result 8

Index **10**

`content-blocks` *Construct protocol-native MCP content blocks*

Description

These constructors create the only content-block objects accepted by `tool_result()`. All R-facing names use `snake_case`; MCP names such as `mimeType` and `_meta` are created only when a result is serialized.

Usage

```
content_text(text, annotations = list(), meta = list())

content_image(data, mime_type, annotations = list(), meta = list())

content_audio(data, mime_type, annotations = list(), meta = list())

content_resource_link(
  uri,
  name,
  title = NULL,
  description = NULL,
  mime_type = NULL,
  size = NULL,
  annotations = list(),
  meta = list()
)

content_resource(
  uri,
  text = NULL,
  blob = NULL,
  mime_type = NULL,
  annotations = list(),
  meta = list()
)
```

Arguments

`text`	A single text string. For `content_resource()`, the resource text payload; exactly one of `text` or `blob` must be supplied.

annotations	Optional named MCP annotations. Known R-facing fields are audience, priority, and last_modified; named extension fields are preserved when serializable.
meta	Optional named outer content-block metadata. For content_resource(), metadata is not duplicated into the inner resource contents object.
data	A scalar base64 string containing inline image or audio data.
mime_type	A single MIME type string. Optional for resources and resource links.
uri	A single non-empty author-supplied resource URI.
name	A single non-empty resource-link name.
title	Optional resource-link display title.
description	Optional resource-link description.
size	Optional non-negative whole-number resource size in bytes.
blob	A scalar base64 string containing an embedded resource payload; exactly one of text or blob must be supplied.

Details

Image, audio, and resource blob data must already be scalar base64 strings. The constructors do not read files, download URLs, infer MIME types, encode, decode, or validate base64 data, or invent resource URIs. Tool authors own payload integrity, accurate MIME types, meaningful URIs, authorization, sanitization, and payload size decisions.

Resource links only transport an author-supplied URI. mcplite remains a tools-only server and does not make that URI readable or implement resources/read. Embedded resources are self-contained content blocks.

Value

A validated content block for use in tool_result().

---

mcp_server *Run an MCP server for R tools over stdio*

Description

mcp_server() runs a Model Context Protocol server over standard input and output. It exposes tools created with mcplite::tool() so MCP clients can discover and call R functions. Compatible ellmer::tool() objects can also be supplied when users already use ellmer, but ellmer is not required for ordinary mcplite tools.

Usage

```
mcp_server(tools, instructions = NULL)
```

Arguments

`tools`	Tool definitions to expose. Supply one `mcplite::tool()` object or a list of tool definitions. Compatible `ellmer::tool()` objects are accepted when supplied directly.
`instructions`	Optional server instructions to advertise to clients that negotiate a protocol version that supports them.

Details

`mcplite` supports the lifecycle, ping, `tools/list`, and `tools/call` subset for protocol versions `2024-11-05`, `2025-06-18`, and `2025-11-25`. Tools may return ordinary R values for legacy text conversion or opt into native content and structured output with `tool_result()`. Structured output, output schemas, audio, resource links, and per-content metadata require MCP `2025-06-18` or later. Text, images, embedded resources, annotations, and result metadata also work with `2024-11-05`.

The server does not implement JSON-RPC batching, HTTP transports, sessions, prompts, resource listing or reading, sampling, elicitation, roots, tasks, progress notifications, or server-initiated requests. Embedded resource blocks are self-contained, and resource links do not make their URIs readable through `mcplite`.

Supply tool definitions directly. For client-launched workflows, put the complete server setup in a script that defines or sources tools and ends with `mcplite::mcp_server(actual_tool_or_list)`, then launch that script with `Rscript --vanilla /absolute/path/to/server.R`.

Value

`mcp_server()` is called for its side effect of serving MCP requests. It blocks the current R process until standard input closes.

OpenTelemetry tracing

`mcplite` automatically creates one OpenTelemetry server span for every parsed MCP request or notification that passes JSON-RPC envelope validation. This includes initialization and notifications, ping, tool discovery and calls, and valid unknown methods. Blank input, JSON parse failures, and malformed JSON-RPC envelopes do not create MCP operation spans.

Tool authors do not need to call `otel::start_local_active_span()` for the MCP operation or tool invocation. The server span remains active while tool code runs, so optional tool-authored spans can become children without being required. `mcplite` does not create a redundant automatic tool-execution child span.

Provider and exporter configuration belongs to the standard `otel` and `otelsdk` environment variables and APIs. `mcplite` does not add telemetry arguments, choose an exporter, or configure a provider. With no exporter configured, tracing is an effective no-op and MCP behavior is unchanged. A safe stderr configuration is `OTEL_R_TRACES_EXPORTER=stderr Rscript --vanilla /absolute/path/to/ser` a remote exporter such as OTLP is also suitable.

Do not use a stdout or console exporter with a stdio MCP server. Standard output is reserved exclusively for MCP protocol messages, so telemetry written there will corrupt the protocol stream.

Remote W3C parent context may be supplied in `params._meta.traceparent`, with optional `params._meta.tracestate`. Malformed propagation data is ignored, and `_meta` is not passed to tool functions. By default, spans

contain selected low-cardinality operation metadata; raw requests and responses, tool arguments and results, `_meta`, trace headers, and condition messages are not recorded as span attributes.

Examples

```
if (identical(Sys.getenv("MCPLITE_CAN_BLOCK_PROCESS"), "true")) {
  add_numbers <- tool(
    function(x, y) {
      x + y
    },
    name = "add_numbers",
    description = "Add two numbers and return the result.",
    arguments = list(
      x = type_number("First number."),
      y = type_number("Second number.")
    )
  )

  mcp_server(list(add_numbers))
}
```

`tool` *Define an MCP tool*

Description

`tool()` wraps an R function with MCP metadata. The resulting object can be supplied directly, or in a list, to `mcp_server()`. Ordinary function return values use the default single-text-block conversion; return `tool_result()` to opt into protocol-native content, structured output, or result metadata.

Usage

```
tool(
  fun,
  description,
  ...,
  arguments = list(),
  name = NULL,
  annotations = list(),
  output_schema = NULL
)
```

Arguments

`fun`	Function to expose as a tool.
`description`	A single string describing when and how to use the tool.
`...`	Not used. Supply argument schemas with `arguments`.

arguments	Named list of argument schemas created by the type_*() helpers.
name	Optional tool name. If omitted, tool() uses the symbol name supplied to fun; anonymous functions must supply name. Tool names must use 1 to 128 characters from letters, digits, underscore, dot, and hyphen.
annotations	Optional named list of MCP tool annotations to advertise.
output_schema	Optional object-shaped output schema created by a supported type_*() helper. Raw schema lists must be wrapped with type_from_schema(). The schema is advertised to MCP 2025-06-18 and later clients; tool authors remain responsible for result conformance.

Details

Character values are returned as literal text, character vectors are joined with newlines, and other JSON-serializable R values are encoded as JSON. Pre-serialized JSON strings remain text, and bare lists are ordinary values, not MCP result objects.

An output_schema advertises the expected object-shaped structured_content to MCP 2025-06-18 and later clients. mcplite checks that the normalized schema has JSON-object wire shape and is serializable, but does not perform runtime schema validation.

Value

A lightweight mcplite tool definition.

Examples

```
add_numbers <- tool(
  function(x, y) {
    x + y
  },
  name = "add_numbers",
  description = "Add two numbers and return the result.",
  arguments = list(
    x = type_number("First number."),
    y = type_number("Second number.")
  )
)
```

---

tool-types *Define MCP tool argument schemas*

Description

These helpers create the JSON Schema subset that mcplite advertises for tool arguments. They describe inputs for clients; tool functions still own domain validation, coercion, authorization or access checks, side-effect safety, output sanitization, and rate limiting where needed.

Usage

```
type_boolean(description = NULL, required = TRUE)

type_integer(description = NULL, required = TRUE)

type_number(description = NULL, required = TRUE)

type_string(description = NULL, required = TRUE)

type_enum(values, description = NULL, required = TRUE)

type_array(items, description = NULL, required = TRUE)

type_object(
  .description = NULL,
  ...,
  .required = TRUE,
  .additional_properties = FALSE
)

type_from_schema(text = NULL, path = NULL)

type_ignore()
```

Arguments

`description`	Optional argument description.
`required`	Whether the argument is listed as required in the parent schema.
`values`	Allowed enum values.
`items`	Type helper describing each array item.
`.description`	Optional object description.
`...`	Named properties for object schemas.
`.required`	Whether the object itself is listed as required in its parent schema.
`.additional_properties`	Whether to allow additional properties.
`text`	A JSON Schema as a list or JSON string.
`path`	Path to a JSON Schema file. Exactly one of `text` or `path` must be supplied.

Details

Generated schemas are MCP-compatible JSON Schema objects. When `$schema` is absent, MCP treats schemas as JSON Schema 2020-12. `type_from_schema()` callers are responsible for supplying valid MCP-compatible schemas. List input preserves the supplied R list shape: use a named empty list for `{}` and an unnamed empty list for `[]`.

Value

A lightweight mcplite tool type object.

Examples

```
type_string("A label.")

type_array(type_integer(), description = "Integer values.")

type_object(
  .description = "A labeled score.",
  label = type_string(),
  score = type_number(required = FALSE)
)

type_from_schema(list(
  type = "string",
  minLength = 1
))
```

`tool_result` *Construct protocol-native MCP tool results*

Description

`tool_result()` is the explicit opt-in boundary for returning protocol-native MCP results. Its `content` must be created with one of the `content_*()` constructors; ordinary R values, including bare lists and JSON strings, retain the legacy single-text-block conversion used by `tool()`.

Usage

```
tool_result(
  content = list(),
  structured_content = NULL,
  is_error = FALSE,
  meta = list()
)
```

Arguments

`content`	One content block created by a `content_*()` constructor, or an ordered list of such blocks. Omitting `content` permits automatic text fallback generation from `structured_content`; explicit `content = list()` suppresses fallback and is invalid.
`structured_content`	Optional named list representing a JSON object. An empty object is allowed. `mcplite` does not validate this value against a tool's advertised output schema.
`is_error`	Whether the result reports a tool error.

`meta` Optional named result metadata. MCP wire names such as `_meta` are created only during serialization.

Details

When `structured_content` is supplied and `content` is omitted, `tool_result()` generates exactly one JSON text fallback block. Explicitly supplied content is preserved as-is and no fallback is appended. Structured content and tool output schemas are sent only to clients using MCP `2025-06-18` or later; older clients receive the generated text fallback.

Value

A protocol-native tool result for return from a tool function.

See Also

`content_text()`, `content_image()`, `content_audio()`, `content_resource_link()`, and `content_resource()`.

Examples

```
tool_result(content_text("Done."))

tool_result(
  content = list(
    content_text("Generated the plot."),
    content_image("base64-data", "image/png")
  )
)

tool_result(structured_content = list(ok = TRUE, count = 2L))
```

Index

content-blocks, 2
content_audio *(*content-blocks*)*, 2
content_audio(), *9*
content_image *(*content-blocks*)*, 2
content_image(), *9*
content_resource *(*content-blocks*)*, 2
content_resource(), *9*
content_resource_link *(*content-blocks*)*, 2
content_resource_link(), *9*
content_text *(*content-blocks*)*, 2
content_text(), *9*

mcp_server, 3
mcp_server(), *5*

tool, 5
tool(), *8*
tool-types, 6
tool_result, 8
tool_result(), *2–5*
type_array *(*tool-types*)*, 6
type_boolean *(*tool-types*)*, 6
type_enum *(*tool-types*)*, 6
type_from_schema *(*tool-types*)*, 6
type_from_schema(), *6*
type_ignore *(*tool-types*)*, 6
type_integer *(*tool-types*)*, 6
type_number *(*tool-types*)*, 6
type_object *(*tool-types*)*, 6
type_string *(*tool-types*)*, 6