Package: mcpr (via r-universe)

May 31, 2026

Title Model Context Protocol for R

Version 0.0.2.9000

Description Create, run, and connect to Model Context Protocol (MCP) servers, and clients in R. The package provides a framework for implementing MCP-compatible servers with tools, resources, and prompts capabilities, as well as client functionality to interact with MCP servers. It includes a JSON-RPC 2.0 implementation for communication between clients and servers.

License GPL (>= 2)

Depends R (>= 4.1.0)

Encoding UTF-8

Roxygen list(markdown = TRUE, roclets = c(``collate'', ``rd'', ``namespace'', ``mcpr::mcp_roclet''))

RoxygenNote 7.3.2

Imports yyjsonr

Suggests ambiorix, processx, knitr, rmarkdown, httr2, ellmer, roxygen2

URL https://mcpr.opifex.org/, https://github.com/devOpifex/mcpr

VignetteBuilder knitr

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

Date/Publication 2025-11-02 14:50:44 UTC

RemoteUrl https://github.com/devOpifex/mcpr

RemoteRef HEAD

RemoteSha b734565e6df0a6fbdd588d5b8892d0779f9a28ca

Contents

`add_capability` *Add a capability to an MCP object*

Description

Add a capability to an MCP object

Usage

```
add_capability(mcp, capability)
```

Arguments

mcp	An MCP server object
capability	A tool, resource, or prompt capability object

Value

The MCP object with the capability added

---

client *Create a new mcp IO*

Description

Create a new mcp IO

Usage

```
new_client_io(command, args = character(), name, version = "1.0.0")

new_client_http(endpoint, name, version = "1.0.0")
```

Arguments

command	The command to run
args	Arguments to pass to the command
name	The name of the client
version	The version of the client
endpoint	The endpoint to connect to

Value

A new mcp client

`ellmer_to_mcpr_tool` *Convert ellmer tools to mcpr tools*

Description

This function converts tools from the ellmer package to a format compatible with the mcpr package. It takes an ellmer ToolDef object and creates an mcpr tool object with the appropriate input schema and handler function.

Usage

```
ellmer_to_mcpr_tool(ellmer_tool)
```

Arguments

`ellmer_tool`	An ellmer ToolDef object created with `ellmer::tool()`

Value

An mcpr tool object compatible with `new_tool()`

See Also

`new_tool()`, `ellmer::tool()`, `add_capability()`

Examples

```
## Not run:
# Create an ellmer tool
ellmer_rnorm <- ellmer::tool(
  rnorm,
  "Generate random normal numbers",
  n = ellmer::type_integer("Number of observations"),
  mean = ellmer::type_number("Mean value", required = FALSE),
  sd = ellmer::type_number("Standard deviation", required = FALSE)
)

# Convert to mcpr format
mcpr_tool <- ellmer_to_mcpr_tool(ellmer_rnorm)

# Add to an mcpr server
server <- new_server("MyServer", "Test server", "1.0.0")
add_capability(server, mcpr_tool)

## End(Not run)
```

get_name *Get the name of a client*

Description

Get the name of a client

Usage

```
get_name(x)
```

Arguments

x	A client object

Value

The name of the client

initialize *Initialize the server with protocol information*

Description

Initialize the server with protocol information

Usage

```
initialize(mcp)
```

Arguments

mcp	A server object

Value

A list containing protocol version, server info, and capabilities

mcp_roclet *MCP Roclet for Generating MCP Servers*

Description

This roclet automatically generates MCP (Model Context Protocol) servers from R functions annotated with @mcp tags.

Usage

```
mcp_roclet()
```

Examples

```
## Not run:
# Use the roclet in roxygenise
roxygen2::roxygenise(roclets = c("rd", "mcpr::mcp_roclet"))

## End(Not run)
```

new_prompt *Create a new prompt*

Description

Create a new prompt

Usage

```
new_prompt(name, description, arguments = list(), handler)
```

Arguments

name	Name of the prompt
description	Description of the prompt
arguments	List of arguments for the prompt
handler	Function to handle the prompt execution

Value

A new prompt capability

Examples

```
prompt <- new_prompt(
  name = "My Prompt",
  description = "This is a description",
  arguments = list(
    input1 = list(
      type = "string",
      description = "Input 1"
    ),
    input2 = list(
      type = "number",
      description = "Input 2"
    )
  ),
  handler = function(params) {
    # Process the prompt request
    return(list(text = "Generated text from prompt"))
  }
)
```

`new_resource` *Create a new resource*

Description

Create a new resource

Usage

```
new_resource(name, description, uri, mime_type = NULL, handler)
```

Arguments

`name`	Name of the resource
`description`	Description of the resource
`uri`	URI of the resource
`mime_type`	MIME type of the resource (optional)
`handler`	Function to handle the resource request

Value

A new resource capability

Examples

```
resource <- new_resource(
  name = "My Resource",
  description = "This is a description",
  uri = "https://example.com/resource",
  mime_type = "text/plain",
  handler = function(params) {
    # Process the resource request
    return(list(content = "Resource content"))
  }
)
```

---

new_server *Create a new MCP object*

Description

Create a new MCP object

Usage

```
new_server(
  name,
  description,
  version,
  tools = list(),
  resources = list(),
  prompts = list()
)

new_mcp(...)
```

Arguments

name	Name of the MCP server
description	Description of the MCP server
version	Version of the MCP server
tools	List of tools (optional)
resources	List of resources (optional)
prompts	List of prompts (optional)
...	Forwarded to new_server()

Value

A new MCP object

Examples

```
mcp <- new_server(
  name = "My MCP",
  description = "This is a description",
  version = "1.0.0"
)
```

`new_tool` *Create a new tool*

Description

Create a new tool

Usage

```
new_tool(name, description, input_schema, handler)
```

Arguments

`name`	Name of the tool
`description`	Description of the tool
`input_schema`	Input schema for the tool (must be a schema object)
`handler`	Function to handle the tool execution

Value

A new tool capability

Examples

```
tool <- new_tool(
  name = "My Tool",
  description = "This is a description",
  input_schema = schema(
    properties = list(
      input1 = property_string("Input 1", "Description of input 1"),
      input2 = property_number("Input 2", "Description of input 2")
    )
  ),
  handler = function(input) {
    # Process the input here
    return(input)
  }
)
```

prompts_get *Get a prompt with the given parameters*

Description

Get a prompt with the given parameters

Usage

```
prompts_get(mcp, params, id = NULL)
```

Arguments

mcp	A server object
params	Parameters for the prompt request
id	Optional request ID for response tracking

Value

A response object with the prompt results or an error

prompts_list *List all available prompts*

Description

List all available prompts

Usage

```
prompts_list(mcp)
```

Arguments

mcp	A server object

Value

A list containing all available prompts

properties *Create a new properties list*

Description

Create a new properties list

Usage

```
properties(...)
```

Arguments

...	Property objects

Value

A list of property objects

Examples

```
properties <- properties(
  property_string("Name", "The name of the user", required = TRUE),
  property_number("Age", "The age of the user in years", minimum = 0)
)
```

property_array *Create an array property definition*

Description

Create an array property definition

Usage

```
property_array(
  title,
  description,
  items,
  required = FALSE,
  min_items = NULL,
  max_items = NULL,
  unique_items = FALSE
)
```

Arguments

title	Short title for the property
description	Longer description of the property
items	Schema for the items in the array
required	Whether the property is required
min_items	Optional minimum number of items
max_items	Optional maximum number of items
unique_items	Logical indicating if items must be unique

Value

An array property object

Examples

```
tags_prop <- property_array(
  "Tags",
  "List of tags for the user",
  items = property_string("Tag", "A user tag"),
  unique_items = TRUE
)
```

---

property_boolean *Create a boolean property definition*

Description

Create a boolean property definition

Usage

```
property_boolean(title, description, required = FALSE)
```

Arguments

title	Short title for the property
description	Longer description of the property
required	Whether the property is required

Value

A boolean property object

Examples

```
active_prop <- property_boolean(
  "Active status",
  "Whether the user account is active",
  required = TRUE
)
```

`property_enum` *Create an enum property with predefined values*

Description

Create an enum property with predefined values

Usage

```
property_enum(title, description, values, required = FALSE)
```

Arguments

`title`	Short title for the property
`description`	Longer description of the property
`values`	Character vector of allowed values
`required`	Whether the property is required

Value

An enum property object

Examples

```
status_prop <- property_enum(
  "Status",
  "User account status",
  values = c("active", "pending", "suspended"),
  required = TRUE
)
```

`property_number` *Create a number property definition*

Description

Create a number property definition

Usage

```
property_number(
  title,
  description,
  required = FALSE,
  minimum = NULL,
  maximum = NULL,
  exclusive_minimum = NULL,
  exclusive_maximum = NULL,
  multiple_of = NULL,
  integer = FALSE
)
```

Arguments

`title`	Short title for the property
`description`	Longer description of the property
`required`	Whether the property is required
`minimum`	Optional minimum value
`maximum`	Optional maximum value
`exclusive_minimum`	Whether minimum is exclusive
`exclusive_maximum`	Whether maximum is exclusive
`multiple_of`	Optional value the number must be a multiple of
`integer`	Whether the number should be an integer

Value

A number property object

Examples

```
age_prop <- property_number(
  "User age",
  "The age of the user in years",
  required = TRUE,
  minimum = 0,
```

```
  integer = TRUE
)
```

`property_object` *Create an object property definition*

Description

Create an object property definition

Usage

```
property_object(
  title,
  description,
  properties,
  required = FALSE,
  additional_properties = FALSE
)
```

Arguments

`title`	Short title for the property
`description`	Longer description of the property
`properties`	List of property definitions for this object
`required`	Whether the property is required
`additional_properties`	Logical indicating if additional properties are allowed

Value

An object property object

Examples

```
address_prop <- property_object(
  "Address",
  "User's address information",
  properties = list(
    street = property_string("Street", "Street address", required = TRUE),
    city = property_string("City", "City name", required = TRUE),
    country = property_string("Country", "Country name")
  )
)
```

`property_string` *Create a string property definition*

Description

Create a string property definition

Usage

```
property_string(
  title,
  description,
  required = FALSE,
  enum = NULL,
  pattern = NULL,
  min_length = NULL,
  max_length = NULL,
  format = NULL
)
```

Arguments

`title`	Short title for the property
`description`	Longer description of the property
`required`	Whether the property is required
`enum`	Optional character vector of allowed values
`pattern`	Optional regex pattern the string must match
`min_length`	Optional minimum length
`max_length`	Optional maximum length
`format`	Optional format constraint

Value

A string property object

Examples

```
name_prop <- property_string(
  "User name",
  "The name of the user",
  required = TRUE,
  min_length = 2,
  max_length = 50
)
```

read *Read a JSON-RPC response from a client provider*

Description

Read a JSON-RPC response from a client provider

Usage

```
read(x, timeout = 60 * 1000)
```

Arguments

x	A client provider
timeout	Timeout in milliseconds for reading the response

Value

The response

register_mcpr_tools *Register MCPR tools with an ellmer chat*

Description

This function registers tools from an MCPR client with an ellmer chat instance.

Usage

```
register_mcpr_tools(chat, client)
```

Arguments

chat	An ellmer chat object
client	An mcpr client object

Value

The chat object (invisibly)

resources_list *List all available resources*

Description

List all available resources

Usage

```
resources_list(mcp)
```

Arguments

mcp	A server object

Value

A list containing all available resources

resources_read *Read a resource with the given parameters*

Description

Read a resource with the given parameters

Usage

```
resources_read(mcp, params, id = NULL)
```

Arguments

mcp	A server object
params	Parameters for the resource read
id	Optional request ID for response tracking

Value

A response object with the resource read results or an error

`response` *Create a response object*

Description

Create a response object

Usage

```
response_text(text)

response_image(image, mime_type = "image/png")

response_audio(audio, mime_type = "audio/mpeg")

response_video(video, mime_type = "video/mp4")

response_file(file, mime_type = "application/octet-stream")

response_resource(resource)

response_error(text)

response_item(
  ...,
  type = c("text", "image", "audio", "video", "file", "resource")
)

response(..., is_error = FALSE)
```

Arguments

`text`	Text content for the response
`image`	Image content
`mime_type`	Mime type of the content
`audio`	Audio content
`video`	Video content
`file`	File content
`resource`	Resource content
`...`	Mutliple `response` objects or passed to `ggplot2::ggsave`
`type`	Type of the content
`is_error`	Whether the response is an error

Details

Use `response_item` to create a custom response item.

Value

A response object

Examples

```
response(
  response_text("Hello, world!"),
  response_image(system.file("extdata/logo.png", package = "mcpr")),
  response_audio(system.file("extdata/sound.mp3", package = "mcpr")),
  response_video(system.file("extdata/video.mp4", package = "mcpr")),
  response_file(system.file("extdata/file.txt", package = "mcpr")),
  response_resource(system.file("extdata/resource.json", package = "mcpr"))
)
```

---

roclet_output.roclet_mcp

Generate MCP server output

Description

Generate MCP server output

Usage

```
## S3 method for class 'roclet_mcp'
roclet_output(x, results, base_path, ...)
```

Arguments

`x`	MCP roclet object
`results`	List of processed tools
`base_path`	Base path
`...`	Additional arguments

Value

NULL (invisible)

roclet_process.roclet_mcp *Process blocks for MCP roclet*

Description

Process blocks for MCP roclet

Usage

```
## S3 method for class 'roclet_mcp'
roclet_process(x, blocks, env, base_path)
```

Arguments

x	MCP roclet object
blocks	List of roxy_block objects
env	Environment
base_path	Base path

Value

List of processed MCP tools

roxy_tag_parse.roxy_tag_mcp *Parse @mcp tag*

Description

Parses @mcp tags to extract tool name and description.

Usage

```
## S3 method for class 'roxy_tag_mcp'
roxy_tag_parse(x)
```

Arguments

x	A roxy_tag object

`roxy_tag_parse.roxy_tag_type` *Parse @type tag*

Description

Parses @type tags to extract parameter type information. Format: @type param_name type enum_values

Usage

```
## S3 method for class 'roxy_tag_type'
roxy_tag_parse(x)
```

Arguments

`x`	A roxy_tag object

`roxy_tag_rd.roxy_tag_mcp` *Roxygen2 tag for @mcp This function is called by Roxygen2 to generate documentation for the @mcp tag*

Description

Roxygen2 tag for @mcp This function is called by Roxygen2 to generate documentation for the @mcp tag

Usage

```
## S3 method for class 'roxy_tag_mcp'
roxy_tag_rd(x, base_path, env)
```

Arguments

`x`	Roxygen2 tag object
`base_path`	Base path for the package
`env`	Environment

Value

NULL (invisible)

roxy_tag_rd.roxy_tag_type *Roxygen2 tag handler for @type This function is called by Roxygen2 to generate documentation for the @type*

Description

Roxygen2 tag handler for @type This function is called by Roxygen2 to generate documentation for the @type

Usage

```
## S3 method for class 'roxy_tag_type'
roxy_tag_rd(x, base_path, env)
```

Arguments

x	Roxygen2 tag object
base_path	Base path for the package
env	Environment

Value

NULL (invisible)

schema *Create a new input schema*

Description

Create a new input schema

Usage

```
schema(properties, type = "object", additional_properties = FALSE)
```

Arguments

properties	List of property definitions created with properties()
type	Type of the schema (default: "object")
additional_properties	Whether additional properties are allowed

Value

A list representing a JSON Schema object

Examples

```
schema <- schema(
  properties = properties(
    name = property_string("User name", "The name of the user", required = TRUE),
    age = property_number("User age", "The age of the user in years", minimum = 0)
  )
)
```

---

serve_http *Serve an MCP server over HTTP using ambiorix*

Description

Serve an MCP server over HTTP using ambiorix

Usage

```
serve_http(mcp, port = Sys.getenv("SHINY_PORT", 3000), path = "/mcp")
```

Arguments

mcp	An MCP server object
port	Port to listen on, defaults to 3000
path	Path to serve the MCP endpoint, defaults to "/mcp"

Value

Invisible, runs indefinitely

---

serve_io *Serve an MCP server using stdin/stdout*

Description

Serve an MCP server using stdin/stdout

Usage

```
serve_io(mcp)
```

Arguments

mcp	An MCP server object created with new_server()

Value

Nothing, runs indefinitely in normal mode, or the response in test mode

---

tools_call *Call a tool with the given parameters*

Description

Call a tool with the given parameters

Usage

```
tools_call(mcp, params, id = NULL)
```

Arguments

mcp	A server object
params	Parameters for the tool call
id	Optional request ID for response tracking

Value

A response object with the tool call results or an error

---

tools_list *List all available tools*

Description

List all available tools

Usage

```
tools_list(mcp)
```

Arguments

mcp	A server object

Value

A list containing all available tools

`write` *Write a JSON-RPC request to a client provider*

Description

Write a JSON-RPC request to a client provider

Usage

```
write(x, method, params = NULL, id = generate_id(), timeout = 5000)
```

Arguments

`x`	A client provider
`method`	The method to call
`params`	The parameters to pass to the method
`id`	The id of the request
`timeout`	Timeout in milliseconds for reading the response

Value

The client provider

Index