Package ‘mx.api’

May 13, 2026

Type Package

Title Minimal Matrix Client-Server API

Version 0.2.0

Date 2026-05-13

Description A minimal-dependency client for the 'Matrix' Client-Server HTTP API <https://spec.matrix.org/>, suitable for talking to a 'Synapse' <https://element-hq.github.io/synapse/> or 'Conduit' <https://conduit.rs/> homeserver. Covers login, room management, message send and history, media upload or download, and the transport endpoints needed to coordinate end-to-end encryption (device-key and one-time-key publication, key query and claim, to-device events). Encryption itself is out of scope; pair with a separate crypto package.

License MIT + file LICENSE

URL https://github.com/cornball-ai/mx.api

BugReports https://github.com/cornball-ai/mx.api/issues

Imports curl, jsonlite

Suggests tinytest

Encoding UTF-8

NeedsCompilation no

Author Troy Hernandez [aut, cre] (ORCID: <https://orcid.org/0009-0005-4248-604X>), cornball.ai [cph]

Maintainer Troy Hernandez <troy@cornball.ai>

Repository CRAN

Date/Publication 2026-05-13 19:50:02 UTC

Contents

`mx.api-package` *mx.api: Minimal Matrix Client-Server API*

Description

Base-R bindings for the Matrix Client-Server HTTP API, suitable for talking to a Synapse homeserver. Two dependencies: curl and jsonlite. End-to-end encryption is out of scope; use unencrypted rooms or a separate crypto package.

Author(s)

Maintainer: Troy Hernandez <troy@cornball.ai>

mx_canonical_json *Encode a value as Matrix canonical JSON*

Description

Produces the canonical JSON byte sequence the Matrix specification requires for signed objects: object keys sorted by UTF-8 byte sequence, no insignificant whitespace, raw UTF-8 for non-ASCII strings, integers only (no floats, no exponents, no decimal places, within `[-(2^53)+1, (2^53)-1]`), and rejection of NaN, Inf, NA values, and NA or duplicate object keys. The output is the exact byte sequence to feed into an ed25519 signer. R named lists become JSON objects; unnamed lists and length > 1 atomic vectors become arrays. Length-1 atomics become scalars. To force a length-1 value to encode as an array, wrap it in a single-element `list(...)` or in `I()` (AsIs values are always encoded as arrays, names dropped, mirroring jsonlite).

Usage

```
mx_canonical_json(x)
```

Arguments

x	An R value: NULL, atomic vector, or list.

Value

A length-1 UTF-8 character string. The result is the exact byte sequence to write to disk or feed to an ed25519 signer; non-ASCII content is preserved as raw UTF-8 bytes (jsonlite-style \uXXXX escaping is not used).

Examples

```
mx_canonical_json(list(b = 2, a = 1))
# "{\"a\":1,\"b\":2}"

mx_canonical_json(list(key = "abc"))
# "{\"key\":\"abc\"}"
```

mx_download *Download a media file by mxc URI*

Description

Download a media file by mxc URI

Usage

```
mx_download(session, mxc_url, dest)
```

Arguments

session	An "mx_session" object.
mxc_url	Character. An "mxc://server/id" URI.
dest	Character. Destination file path.

Value

The destination path, invisibly.

Examples

```
## Not run:
mx_download(s, "mxc://matrix.example/abc123", tempfile())

## End(Not run)
```

---

mx_keys_claim *Claim one-time keys for an Olm handshake*

Description

POST `/_matrix/client/v3/keys/claim`. The `one_time_keys` argument selects which algorithm to claim for each `(user_id, device_id)` pair.

Usage

```
mx_keys_claim(session, one_time_keys, timeout = 10000L)
```

Arguments

session	An `mx_session`.
one_time_keys	Named list. Names are user ids; values are named lists mapping device ids to the desired algorithm (typically `"signed_curve25519"`).
timeout	Integer milliseconds. Server-side timeout when talking to remote homeservers.

Value

Parsed response with the claimed keys keyed by `user_id -> device_id -> "<algorithm>:<key_id>" -> key_object`.

Examples

```
## Not run:
mx_keys_claim(s, list(
  "@alice:example.org" = list(ABCD1234 = "signed_curve25519")
))

## End(Not run)
```

mx_keys_query *Query device keys for one or more users*

Description

POST `/_matrix/client/v3/keys/query`. Each entry in `device_keys` maps a Matrix user id to a character vector of device ids to request. An empty character vector requests all of the user's devices.

Usage

```
mx_keys_query(session, device_keys, timeout = 10000L)
```

Arguments

`session`	An `mx_session`.
`device_keys`	Named list. Names are Matrix user ids (e.g. `"@@alice:example.org"`); values are character vectors of device ids, or `character(0)` for "all devices".
`timeout`	Integer milliseconds. Time the server should wait for remote homeservers before returning a partial result.

Value

Parsed response with a `device_keys` map of `user_id -> device_id -> device_keys_object`.

Examples

```
## Not run:
mx_keys_query(s, list("@alice:example.org" = character()))

## End(Not run)
```

mx_keys_upload *Upload device identity and one-time keys*

Description

POST `/_matrix/client/v3/keys/upload`. The `device_keys` and `one_time_keys` arguments must be fully formed and signed per the Matrix specification; mx.api will not canonicalise or sign them. Use `mx_canonical_json` to produce the byte sequence to sign.

Usage

```
mx_keys_upload(session, device_keys = NULL, one_time_keys = NULL,
               fallback_keys = NULL)
```

Arguments

session	An mx_session.
device_keys	Named list. The device_keys object as defined in the Matrix spec, including user_id, device_id, algorithms, keys, and the signatures block produced by the caller's signer. Pass NULL to upload only one-time keys.
one_time_keys	Named list or NULL. Map from "<algorithm>:<key_id>" (e.g. "signed_curve25519:AAAA") to the signed key object. Pass NULL or an empty list to skip.
fallback_keys	Named list or NULL. Same shape as one_time_keys; used to advertise a fallback key when the OTK pool is exhausted.

Value

The parsed homeserver response, including one_time_key_counts.

Examples

```
## Not run:
mx_keys_upload(s, device_keys = signed_dk, one_time_keys = signed_otks)

## End(Not run)
```

---

mx_login *Log in to a Matrix homeserver*

Description

Authenticates with a Matrix homeserver using password login and returns a session object carrying the access token and device id.

Usage

```
mx_login(server, user, password, device_id = NULL)
```

Arguments

server	Character. Homeserver base URL (e.g. "https://matrix.example").
user	Character. User localpart or full Matrix ID.
password	Character. Account password.
device_id	Character or NULL. Reuse an existing device id.

Value

An object of class "mx_session".

Examples

```
## Not run:
s <- mx_login("https://matrix.example", "alice", "hunter2")

## End(Not run)
```

---

mx_logout *Log out of a Matrix session*

Description

Invalidates the access token on the homeserver.

Usage

```
mx_logout(session)
```

Arguments

session	An "mx_session" object.

Value

Invisible NULL.

Examples

```
## Not run:
mx_logout(s)

## End(Not run)
```

---

mx_messages *Fetch historical messages from a room*

Description

Thin wrapper over the /rooms/{id}/messages endpoint.

Usage

```
mx_messages(session, room_id, from = NULL, dir = "b", limit = 50L)
```

Arguments

session	An "mx_session" object.
room_id	Character. The room ID.
from	Character or NULL. Pagination token; NULL starts at the most recent message.
dir	Character. "b" (backwards, default) or "f" (forwards).
limit	Integer. Maximum events to return.

Value

A list with fields chunk, start, end.

Examples

```
## Not run:
mx_messages(s, "!abc:matrix.example", limit = 20L)

## End(Not run)
```

---

mx_react *Send a reaction (annotation) to a room event*

Description

Posts an m.reaction event tying `key` (usually a thumbs-up or other emoji) to `event_id`. Matrix reactions are plain events under the hood; they relate to the target via `m.annotation`.

Usage

```
mx_react(session, room_id, event_id, key)
```

Arguments

session	An "mx_session" object.
room_id	Character. The room ID.
event_id	Character. The event being reacted to.
key	Character. The reaction key (usually an emoji).

Value

The event ID of the sent reaction.

Examples

```
## Not run:
mx_react(s, "!abc:matrix.example", "$eventid", "thumbs-up")

## End(Not run)
```

mx_read_receipt *Send a read receipt for a room event*

Description

Public receipt (`m.read`) advances the "seen" marker in other clients; private receipt (`m.read.private`) only advances the bot's own view. Defaults to public so user clients show "seen by @bot".

Usage

```
mx_read_receipt(session, room_id, event_id,
                receipt_type = c("m.read", "m.read.private"))
```

Arguments

`session`	An "mx_session" object.
`room_id`	Character. The room ID.
`event_id`	Character. The event to mark as read.
`receipt_type`	Character. "m.read" (default) or "m.read.private".

Value

Invisible NULL.

Examples

```
## Not run:
mx_read_receipt(s, "!abc:matrix.example", "$eventid")

## End(Not run)
```

mx_register *Register a new account on a Matrix homeserver*

Description

Creates a new user via POST `/_matrix/client/v3/register` using the `m.login.dummy` auth flow. Most homeservers only accept this when open registration is enabled (or a registration token is supplied). On success returns a ready-to-use `mx_session` — registration also logs the new user in.

Usage

```
mx_register(server, username, password, device_id = NULL,
            initial_device_display_name = NULL, inhibit_login = FALSE)
```

Arguments

server	Character. Homeserver base URL.
username	Character. Desired localpart (e.g. "alice").
password	Character. Account password.
device_id	Character or NULL. Device id to assign; a server- generated one is used if NULL.
initial_device_display_name	Character or NULL. Human-readable label for the device.
inhibit_login	Logical. When TRUE, the server creates the account but does not return a session; the call returns a list with the new user_id instead of an mx_session.

Value

An mx_session object on login, or a list with user_id when inhibit_login = TRUE.

Examples

```
## Not run:
s <- mx_register("https://matrix.example", "alice", "hunter2")

## End(Not run)
```

---

mx_rooms *List rooms the user has joined*

Description

List rooms the user has joined

Usage

```
mx_rooms(session)
```

Arguments

session	An "mx_session" object.

Value

Character vector of room IDs.

Examples

```
## Not run:
mx_rooms(s)

## End(Not run)
```

mx_room_create *Create a room*

Description

Create a room

Usage

```
mx_room_create(session, name = NULL, topic = NULL, visibility = "private",
               preset = NULL, invite = character())
```

Arguments

session	An "mx_session" object.
name	Character or NULL. Human-readable room name.
topic	Character or NULL. Room topic.
visibility	Character. "private" (default) or "public".
preset	Character or NULL. A Matrix room preset ("private_chat", "trusted_private_chat", "public_chat").
invite	Character vector. Matrix IDs to invite.

Value

The new room ID as a character string.

Examples

```
## Not run:
room_id <- mx_room_create(s, name = "test", topic = "hello")

## End(Not run)
```

mx_room_join *Join a room by ID or alias*

Description

Join a room by ID or alias

Usage

```
mx_room_join(session, room)
```

Arguments

session	An "mx_session" object.
room	Character. Room ID (!abc:server) or alias (#name:server).

Value

The joined room ID.

Examples

```
## Not run:
mx_room_join(s, "#general:matrix.example")

## End(Not run)
```

---

mx_room_leave *Leave a room*

Description

Leave a room

Usage

```
mx_room_leave(session, room_id)
```

Arguments

session	An "mx_session" object.
room_id	Character. The room ID.

Value

Invisible NULL.

Examples

```
## Not run:
mx_room_leave(s, "!abc:matrix.example")

## End(Not run)
```

mx_room_members *List the members of a room*

Description

List the members of a room

Usage

```
mx_room_members(session, room_id)
```

Arguments

session	An "mx_session" object.
room_id	Character. The room ID.

Value

Character vector of Matrix user IDs currently joined.

Examples

```
## Not run:
mx_room_members(s, "!abc:matrix.example")

## End(Not run)
```

mx_room_name *Get a room's human-readable name*

Description

Reads the `m.room.name` state event. Returns NULL if the room has no name set or the state event is inaccessible.

Usage

```
mx_room_name(session, room_id)
```

Arguments

session	An "mx_session" object.
room_id	Character. The room ID.

Value

Character scalar or NULL.

Examples

```
## Not run:
mx_room_name(s, "!abc:matrix.example")

## End(Not run)
```

`mx_room_topic` *Get a room's topic*

Description

Reads the `m.room.topic` state event. Returns NULL if the room has no topic set or the state event is inaccessible.

Usage

```
mx_room_topic(session, room_id)
```

Arguments

`session`	An "mx_session" object.
`room_id`	Character. The room ID.

Value

Character scalar or NULL.

Examples

```
## Not run:
mx_room_topic(s, "!abc:matrix.example")

## End(Not run)
```

`mx_send` *Send a message to a room*

Description

Send a message to a room

Usage

```
mx_send(session, room_id, body, msgtype = "m.text", extra = NULL)
```

Arguments

session	An "mx_session" object.
room_id	Character. The room ID.
body	Character. The message body.
msgtype	Character. Matrix msgtype, default "m.text".
extra	List or NULL. Extra fields merged into the event content (e.g. formatted body, reply relation).

Value

The event ID of the sent message.

Examples

```
## Not run:
mx_send(s, "!abc:matrix.example", "hello world")

## End(Not run)
```

---

mx_send_to_device *Send a to-device event*

Description

PUT /_matrix/client/v3/sendToDevice/{eventType}/{txnId}. Used to ship encrypted Olm payloads (e.g. m.room_key carriers wrapped as m.room.encrypted) to specific (user_id, device_id) targets.

Usage

```
mx_send_to_device(session, event_type, messages, txn_id = NULL)
```

Arguments

session	An mx_session.
event_type	Character. The to-device event type, e.g. "m.room.encrypted".
messages	Named list. Outer names are user ids; values are named lists mapping device id (or the wildcard "*") to the event content.
txn_id	Character or NULL. Idempotency key; auto-generated when NULL.

Value

Invisible NULL (the server returns an empty body on success).

Examples

```
## Not run:
mx_send_to_device(s, "m.room.encrypted", list(
  "@bob:example.org" = list(BBBB = encrypted_content)
))

## End(Not run)
```

`mx_session` *Reconstruct a session from saved credentials*

Description

Reconstruct a session from saved credentials

Usage

```
mx_session(server, token, user_id, device_id)
```

Arguments

`server`	Character. Homeserver base URL.
`token`	Character. Access token from a prior login.
`user_id`	Character. Full Matrix ID (e.g. "@troy:example.org").
`device_id`	Character. Device id from the prior login.

Value

An object of class "mx_session".

Examples

```
s <- mx_session(
    server = "https://matrix.example",
    token = "syt_...",
    user_id = "@alice:matrix.example",
    device_id = "ABC123"
)
```

mx_sync *One-shot sync against the homeserver*

Description

Calls /sync once and returns immediately. For streaming behaviour, the caller writes its own loop, passing the previous batch's next_batch token as `since`.

Usage

```
mx_sync(session, since = NULL, timeout = 0L, filter = NULL)
```

Arguments

`session`	An "mx_session" object.
`since`	Character or NULL. Sync token from a prior sync.
`timeout`	Integer. Long-poll timeout in milliseconds (0 returns immediately).
`filter`	Character or NULL. Filter ID or inline JSON filter.

Value

The parsed sync response, including next_batch.

Examples

```
## Not run:
batch <- mx_sync(s)
next_batch <- batch$next_batch

## End(Not run)
```

mx_upload *Upload a file to the homeserver media repository*

Description

Upload a file to the homeserver media repository

Usage

```
mx_upload(session, path, content_type = NULL, filename = NULL)
```

Arguments

session	An "mx_session" object.
path	Character. Local file path.
content_type	Character or NULL. MIME type; guessed from the file extension if NULL.
filename	Character or NULL. Filename advertised to the server.

Value

An "mxc://" URI as a character string.

Examples

```
## Not run:
uri <- mx_upload(s, "photo.png")

## End(Not run)
```

---

mx_whoami *Return the identity of the current session*

Description

Return the identity of the current session

Usage

```
mx_whoami(session)
```

Arguments

session	An "mx_session" object.

Value

A list with user_id and device_id.

Examples

```
## Not run:
mx_whoami(s)

## End(Not run)
```

Index