Package: mx.api (via r-universe)

June 10, 2026

Type Package

Title Minimal Matrix Client-Server API

Version 0.3.0

Date 2026-06-10

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 https://cran.r-universe.dev

Date/Publication 2026-06-10 18:00:02 UTC

RemoteUrl https://github.com/cran/mx.api

RemoteRef HEAD

RemoteSha b396737d2d2cf02797db161d9539ef5a8e91a683

Contents

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

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.

Package Content

Index of help topics:

mx_send_to_device	Send a to-device event
mx_session	Reconstruct a session from saved credentials
mx_set_account_data	Set account data
mx_set_avatar_url	Set this user's avatar
mx_set_displayname	Set this user's display name
mx_set_state	Set a room state event
mx_sync	One-shot sync against the homeserver
mx_typing	Send a typing notification
mx_upload	Upload a file to the homeserver media repository
mx_whoami	Return the identity of the current session

Maintainer

Troy Hernandez <troy@cornball.ai>

Author(s)

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

`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.

Usage

```
mx_canonical_json(x)
```

Arguments

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

Details

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).

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_delete_device *Delete a device*

Description

Removes a device and invalidates its access token. Most homeservers protect this with user-interactive authentication: the first call fails with M_FORBIDDEN or a 401 carrying a `flows` object, and the caller retries with a completed `auth` payload, e.g. `list(type = "m.login.password", identifier = list(type = "m.id.user", user = "bot"), password = "...", session = "<from the 401>")`. mx.api deliberately does not automate that exchange.

Usage

```
mx_delete_device(session, device_id, auth = NULL)
```

Arguments

`session`	An "mx_session" object.
`device_id`	Character. The device to delete.
`auth`	List or NULL. Completed user-interactive auth payload.

Value

Invisibly TRUE on success.

Examples

```
## Not run:
mx_delete_device(s, "OLDDEVICE", auth = list(
    type = "m.login.password",
    identifier = list(type = "m.id.user", user = "bot"),
    password = "secret"
))

## End(Not run)
```

mx_devices *List this account's devices*

Description

List this account's devices

Usage

```
mx_devices(session)
```

Arguments

session	An "mx_session" object.

Value

A list of devices, each with `device_id`, `display_name`, `last_seen_ip`, and `last_seen_ts`.

Examples

```
## Not run:
vapply(mx_devices(s), function(d) d$device_id, character(1))

## End(Not run)
```

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_get_account_data` *Get account data*

Description

Reads a global account-data event for a user, e.g. `"m.direct"` (the DM room map) or any custom namespaced type.

Usage

```
mx_get_account_data(session, type, user_id = session$user_id)
```

Arguments

`session`	An "mx_session" object.
`type`	Character. Event type, e.g. `"m.direct"`.
`user_id`	Character. Defaults to the session's own user.

Value

The account-data content as a list, or NULL when the type has never been set.

Examples

```
## Not run:
direct <- mx_get_account_data(s, "m.direct")

## End(Not run)
```

mx_get_state *Get a room state event*

Description

Read-side counterpart of mx_set_state, e.g. to check whether a room is encrypted before joining the send path.

Usage

```
mx_get_state(session, room_id, event_type, state_key = "")
```

Arguments

session	An "mx_session" object.
room_id	Character. The room ID.
event_type	Character. State event type, e.g. "m.room.encryption".
state_key	Character. State key (default empty string).

Value

The state event content as a list, or NULL when the state event is not set.

Examples

```
## Not run:
enc <- mx_get_state(s, "!abc:example", "m.room.encryption")
is.null(enc)   # FALSE in an encrypted room

## End(Not run)
```

mx_guess_mime *Guess a MIME type from a file extension*

Description

The extension table mx.api uses for uploads, exported so callers do not maintain their own. Unknown extensions fall back to "application/octet-stream".

Usage

```
mx_guess_mime(path)
```

Arguments

path	Character. File path or name.

Value

Character MIME type.

Examples

```
mx_guess_mime("clip.mp4")
mx_guess_mime("notes.txt")
```

`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_media_config` *Query the homeserver's media configuration*

Description

Asks the server for its media limits, chiefly `m.upload.size` (maximum upload bytes), so callers can check a file fits before uploading. Tries the v1 endpoint and falls back to the legacy location for older homeservers.

Usage

```
mx_media_config(session)
```

Arguments

session	An "mx_session" object.

Value

A list; $`m.upload.size` is the upload cap in bytes (may be absent if the server does not advertise one).

Examples

```
## Not run:
cap <- mx_media_config(s)$`m.upload.size`
file.size("clip.mp4") <= cap

## 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_profile *Get a user's profile*

Description

Get a user's profile

Usage

```
mx_profile(session, user_id = session$user_id)
```

Arguments

session	An "mx_session" object.
user_id	Character. Defaults to the session's own user.

Value

A list with displayname and avatar_url (either may be absent when unset).

Examples

```
## Not run:
mx_profile(s)$displayname

## 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_redact *Redact an event*

Description

Removes the content of a message, reaction, or other event. This is how Matrix deletes things.

Usage

```
mx_redact(session, room_id, event_id, reason = NULL, txn_id = NULL)
```

Arguments

session	An "mx_session" object.
room_id	Character. The room ID.
event_id	Character. The event to redact.
reason	Character or NULL. Optional human-readable reason.
txn_id	Character or NULL. Transaction id (generated if NULL).

Value

The event ID of the redaction event.

Examples

```
## Not run:
mx_redact(s, "!abc:example", "$someevent", reason = "typo")

## 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_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_invite` *Invite a user to a room*

Description

Invitation at creation time is covered by `mx_room_create()`; this covers the other common lifecycle case, inviting into an existing room.

Usage

```
mx_room_invite(session, room_id, user_id)
```

Arguments

`session`	An "mx_session" object.
`room_id`	Character. The room ID.
`user_id`	Character. The Matrix ID to invite.

Value

Invisibly TRUE on success.

Examples

```
## Not run:
mx_room_invite(s, "!abc:example", "@friend:example.org")

## 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_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_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_event` *Send an arbitrary room event*

Description

Generic counterpart to `mx_send` for event types other than `m.room.message`, such as `m.room.encrypted`. The content is sent verbatim.

Usage

```
mx_send_event(session, room_id, event_type, content, txn_id = NULL)
```

Arguments

`session`	An "mx_session" object.
`room_id`	Character. The room ID.
`event_type`	Character. The event type, e.g. `"m.room.encrypted"`.
`content`	List. The event content, sent as-is.
`txn_id`	Character or NULL. Transaction id (generated if NULL).

Value

The event ID of the sent event.

Examples

```
## Not run:
mx_send_event(s, "!abc:example", "m.room.encrypted", encrypted_content)

## End(Not run)
```

`mx_send_media` *Send a media file to a room*

Description

Uploads `path` to the media repository and posts an `m.room.message` referencing it. The default `info` carries `mimetype` and `size`; pass richer metadata (width, height, duration) yourself – mx.api deliberately does not inspect media files.

Usage

```
mx_send_media(session, room_id, path, body = basename(path), msgtype = NULL,
              content_type = NULL, info = list())

mx_send_file(session, room_id, path, body = basename(path),
             content_type = NULL, info = list())

mx_send_image(session, room_id, path, body = basename(path),
              content_type = NULL, info = list())

mx_send_audio(session, room_id, path, body = basename(path),
              content_type = NULL, info = list())

mx_send_video(session, room_id, path, body = basename(path),
              content_type = NULL, info = list())
```

Arguments

`session`	An "mx_session" object.
`room_id`	Character. The room ID.
`path`	Character. Path to the file to upload.
`body`	Character. Message body / filename shown by clients.
`msgtype`	Character or NULL. One of `"m.file"`, `"m.image"`, `"m.audio"`, `"m.video"`. NULL (the default) derives it from the MIME type, so a .mp4 posts as m.video without being told.
`content_type`	Character or NULL. MIME type (guessed from the extension when NULL).
`info`	List. Extra fields merged into the `info` object.

Value

The event ID of the sent message.

Examples

```
## Not run:
mx_send_media(s, "!abc:example", "clip.mp4", msgtype = "m.video")

## 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_set_account_data *Set account data*

Description

Writes a global account-data event for a user. The content replaces whatever was stored under type.

Usage

```
mx_set_account_data(session, type, content, user_id = session$user_id)
```

Arguments

session	An "mx_session" object.
type	Character. Event type, e.g. "m.direct".
content	List. The content, sent as-is.
user_id	Character. Defaults to the session's own user.

Value

Invisibly TRUE on success.

Examples

```
## Not run:
mx_set_account_data(s, "ai.cornball.notes", list(theme = "dark"))

## End(Not run)
```

---

mx_set_avatar_url *Set this user's avatar*

Description

Set this user's avatar

Usage

```
mx_set_avatar_url(session, avatar_url)
```

Arguments

session	An "mx_session" object.
avatar_url	Character. An mxc:// URI, typically from mx_upload.

Value

Invisibly TRUE on success.

Examples

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

## End(Not run)
```

mx_set_displayname *Set this user's display name*

Description

Set this user's display name

Usage

```
mx_set_displayname(session, displayname)
```

Arguments

session	An "mx_session" object.
displayname	Character. The new display name.

Value

Invisibly TRUE on success.

Examples

```
## Not run:
mx_set_displayname(s, "cornelius")

## End(Not run)
```

mx_set_state *Set a room state event*

Description

Generic state setter, e.g. to mark a room encrypted by putting an `m.room.encryption` event.

Usage

```
mx_set_state(session, room_id, event_type, content, state_key = "")
```

Arguments

session	An "mx_session" object.
room_id	Character. The room ID.
event_type	Character. State event type, e.g. `"m.room.encryption"`.
content	List. The state content, sent as-is.
state_key	Character. State key (default empty string).

Value

The event ID of the state event.

Examples

```
## Not run:
mx_set_state(s, "!abc:example", "m.room.encryption",
             list(algorithm = "m.megolm.v1.aes-sha2"))

## End(Not run)
```

`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_typing *Send a typing notification*

Description

Shows (or clears) the session user's typing indicator in a room. Useful bot polish while a slow reply is being generated.

Usage

```
mx_typing(session, room_id, typing = TRUE, timeout = 30000L)
```

Arguments

session	An "mx_session" object.
room_id	Character. The room ID.
typing	Logical. TRUE to show typing, FALSE to clear it.
timeout	Integer. How long the indicator lasts, in milliseconds (ignored when typing = FALSE).

Value

Invisibly TRUE on success.

Examples

```
## Not run:
mx_typing(s, "!abc:example", TRUE)
# ... generate the reply ...
mx_typing(s, "!abc:example", FALSE)

## 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