Package ‘mx.client’

August 5, 2026

Type Package

Title Stateful Matrix Client Helpers

Version 0.2.0

Date 2026-08-04

Description Stateful helpers for building 'Matrix' (<https://matrix.org>) chat clients in R. Builds on the low-level 'mx.api' Client-Server API bindings, adding local configuration persistence, room resolution, sync cursor handling, sync-event extraction, invite acceptance, a conservative Markdown-to-HTML converter for formatted messages, and 'Olm'/'Megolm' end-to-end encryption orchestration over the optional 'mx.crypto' package.

License Apache License (>= 2)

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

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

Depends R (>= 4.0)

Imports jsonlite, mx.api (>= 0.3.0), stats, tools, utils

Suggests mx.crypto (>= 0.2.0), simplermarkdown, tinytest

VignetteBuilder simplermarkdown

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 2026-08-04 22:10:08 UTC

Contents

mx.client-package . . . 3
mx_accept_invites . . . 5
mx_client_configure . . . 5
mx_client_config_path . . . 6
mx_client_from_config . . . 7
mx_client_legacy_config_path . . . 7
mx_client_load . . . 8
mx_client_relogin . . . 9
mx_client_save . . . 9
mx_client_session . . . 10
mx_crypto_account . . . 11
mx_crypto_account_save . . . 11
mx_crypto_claim_otks . . . 12
mx_crypto_decrypt_event . . . 13
mx_crypto_device_keys . . . 13
mx_crypto_encrypt_event . . . 14
mx_crypto_encrypt_for_devices . . . 15
mx_crypto_handle_to_device . . . 16
mx_crypto_inbound_session . . . 17
mx_crypto_known_devices . . . 18
mx_crypto_process_sync . . . 18
mx_crypto_publish_keys . . . 19
mx_crypto_room_key_payload . . . 20
mx_crypto_sessions_load . . . 21
mx_crypto_sessions_new . . . 22
mx_crypto_sessions_save . . . 22
mx_crypto_store_dir . . . 23
mx_extract_invites . . . 23
mx_extract_reaction_verdict . . . 24
mx_extract_text_events . . . 25
mx_markdown_to_html . . . 26
mx_pill_mentions . . . 26
mx_resolve_room . . . 27
mx_room_encrypted . . . 27
mx_room_lookup_by_name . . . 28
mx_send_encrypted . . . 29
mx_send_media . . . 30
mx_send_table . . . 31
mx_send_text . . . 31
mx_set_displayname . . . 32
mx_sync_update . . . 33
mx_table_html . . . 34
mx_with_relogin . . . 34
print.mx_client_config . . . 35

Index **37**

mx.client-package *Stateful Matrix Client Helpers*

Description

Stateful helpers for building 'Matrix' (<https://matrix.org>) chat clients in R. Builds on the low-level 'mx.api' Client-Server API bindings, adding local configuration persistence, room resolution, sync cursor handling, sync-event extraction, invite acceptance, a conservative Markdown-to-HTML converter for formatted messages, and 'Olm'/'Megolm' end-to-end encryption orchestration over the optional 'mx.crypto' package.

Package Content

Index of help topics:


```
mx.client-package       Stateful Matrix Client Helpers
mx_accept_invites       Accept pending Matrix room invites
mx_client_config_path   Path to a Matrix client config file
mx_client_configure     Configure and save a Matrix client
mx_client_from_config   Wrap a list as an mx.client config
mx_client_legacy_config_path
                        Legacy Matrix config path for an application
mx_client_load          Load a Matrix client config
mx_client_relogin       Re-login with stored credentials and refresh
                        the saved token
mx_client_save          Save a Matrix client config
mx_client_session       Build an mx.api session from a client config
mx_crypto_account       Load or create this client's Olm account
mx_crypto_account_save
                        Persist an Olm account to the store
mx_crypto_claim_otks    Claim a one-time key for each device
mx_crypto_decrypt_event
                        Decrypt an m.room.encrypted event (Megolm)
mx_crypto_device_keys   Build a signed device_keys object for upload
mx_crypto_encrypt_event
                        Encrypt event content for a room (Megolm)
mx_crypto_encrypt_for_devices
                        Encrypt an event for an encrypted room's
                        devices
mx_crypto_handle_to_device
                        Decrypt an inbound Olm to-device payload
mx_crypto_inbound_session
                        Build an inbound Megolm session from a shared
                        room key
mx_crypto_known_devices
                        List the devices (and identity keys) of some
                        users
```

```
mx_crypto_process_sync
                        Process a sync response: store room keys,
                        decrypt room events
mx_crypto_publish_keys
                        Publish this device's identity and one-time
                        keys
mx_crypto_room_key_payload
                        Encrypt a Megolm room key to one device as a
                        to-device payload
mx_crypto_sessions_load
                        Load a session set from the crypto store
mx_crypto_sessions_new
                        Create an empty E2EE session set
mx_crypto_sessions_save
                        Persist a session set to the crypto store
mx_crypto_store_dir     Directory holding this client's encryption
                        state
mx_extract_invites      Extract pending invite room ids from a sync
                        response
mx_extract_reaction_verdict
                        Extract a reaction approval verdict from sync
                        events
mx_extract_text_events
                        Extract text message events from a sync
                        response
mx_markdown_to_html     Convert a conservative markdown subset to
                        Matrix custom HTML
mx_pill_mentions        Turn textual @mentions into Matrix pills in
                        formatted HTML
mx_resolve_room         Resolve a room id, name, or default room
mx_room_encrypted       Is a room end-to-end encrypted?
mx_room_lookup_by_name
                        Look up a joined room by display name
mx_send_encrypted       Send an end-to-end encrypted message to a room
mx_send_media           Send a media file to a Matrix room
mx_send_table           Send tabular data to a Matrix room
mx_send_text            Send plain text to a Matrix room
mx_set_displayname      Set the account's profile display name
mx_sync_update          Sync once and update the stored cursor
mx_table_html           Render tabular data as Matrix custom HTML
mx_with_relogin         Run a client operation, re-logging in once on
                        an expired token
print.mx_client_config
                        Print a Matrix client config
```

Maintainer

Troy Hernandez <troy@cornball.ai>

Author(s)

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

---

mx_accept_invites *Accept pending Matrix room invites*

Description

Accept pending Matrix room invites

Usage

```
mx_accept_invites(client, invites)
```

Arguments

client	Matrix client config.
invites	Character vector of room ids.

Value

Character vector of joined room ids.

Examples

```
## Not run:
# Needs a live homeserver session.
client <- mx_client_load("myapp")
res <- mx_sync_update(client)
mx_accept_invites(res$client, mx_extract_invites(res$sync))

## End(Not run)
```

---

mx_client_configure *Configure and save a Matrix client*

Description

Logs in with **mx.api**, joins or records the target room, and writes a reusable local config. Extra fields are merged into the saved config so applications can persist their own defaults without reimplementing login.

Usage

```
mx_client_configure(server, user, password, room, app = "mx.client",
                    path = NULL, device_id = NULL, extra = list())
```

Arguments

server	Character. Homeserver base URL.
user	Character. Matrix user localpart or full MXID.
password	Character. Account password.
room	Character. Room ID or alias to join.
app	Character. Application namespace.
path	Character or NULL. Explicit destination path.
device_id	Character or NULL. Existing device id to reuse.
extra	Named list. Additional fields to save.

Value

Saved "mx_client_config", invisibly.

Examples

```
## Not run:
# Needs a live homeserver and account credentials.
mx_client_configure("https://matrix.example.org", "bot", "secret",
                    room = "#general:example.org", app = "myapp")

## End(Not run)
```

---

mx_client_config_path *Path to a Matrix client config file*

Description

Resolves the config path for an application that uses mx.client. The default environment variable is derived from app; for example, app = "corteza" honors CORTEZA_MATRIX_CONFIG.

Usage

```
mx_client_config_path(app = "mx.client", env_var = NULL)
```

Arguments

app	Character. Application namespace for tools::R_user_dir().
env_var	Character or NULL. Override environment variable name.

Value

Character path.

Examples

```
mx_client_config_path("myapp")
```

mx_client_from_config *Wrap a list as an mx.client config*

Description

Wrap a list as an mx.client config

Usage

```
mx_client_from_config(cfg, path = NULL, app = NULL)
```

Arguments

cfg	Named list.
path	Character or NULL. Source/sink path for saves.
app	Character or NULL. Application namespace.

Value

An object of class "mx_client_config".

Examples

```
cfg <- mx_client_from_config(list(server = "https://matrix.example.org",
                                  token = "syt_example",
                                  user_id = "@bot:example.org",
                                  device_id = "DEVICEID"))
class(cfg)
```

mx_client_legacy_config_path *Legacy Matrix config path for an application*

Description

Currently only app = "corteza" has a historical path: ~/.corteza/matrix.json.

Usage

```
mx_client_legacy_config_path(app = "mx.client")
```

Arguments

app	Character. Application namespace.

Value

Character path or NULL.

Examples

```
mx_client_legacy_config_path("corteza")
```

`mx_client_load` *Load a Matrix client config*

Description

Reads a JSON config. If `path` or the derived environment variable is explicit, that path is authoritative. Otherwise `legacy_path` is used as a compatibility fallback when present.

Usage

```
mx_client_load(app = "mx.client", path = NULL,
               legacy_path = mx_client_legacy_config_path(app), env_var = NULL)
```

Arguments

`app`	Character. Application namespace.
`path`	Character or NULL. Explicit config path.
`legacy_path`	Character or NULL. Backward-compatible fallback path.
`env_var`	Character or NULL. Override environment variable name.

Value

An `"mx_client_config"` object.

Examples

```
path <- file.path(tempdir(), "matrix.json")
cfg <- mx_client_from_config(list(server = "https://matrix.example.org",
                                  token = "syt_example",
                                  user_id = "@bot:example.org",
                                  device_id = "DEVICEID"))
mx_client_save(cfg, path = path)
mx_client_load(path = path)$user_id
unlink(path)
```

mx_client_relogin *Re-login with stored credentials and refresh the saved token*

Description

Uses the password persisted in the client config to obtain a fresh access token for the *same* device (reusing `client$device_id`, so an E2EE device identity survives the refresh), then saves the updated config. Typical use is recovering from an invalidated token; see `mx_with_relogin` for the catch-and-retry wrapper.

Usage

```
mx_client_relogin(client, save = TRUE)
```

Arguments

`client`	Matrix client config with `password`.
`save`	Logical. Persist the refreshed config (default TRUE).

Value

The refreshed `"mx_client_config"`.

Examples

```
## Not run:
# Needs a live homeserver and a stored password.
client <- mx_client_relogin(mx_client_load("myapp"))

## End(Not run)
```

mx_client_save *Save a Matrix client config*

Description

Writes JSON with mode 0600.

Usage

```
mx_client_save(client, app = NULL, path = NULL)
```

Arguments

`client`	Named list or `"mx_client_config"`.
`app`	Character or NULL. Application namespace.
`path`	Character or NULL. Destination path.

Value

The saved config, invisibly.

Examples

```
path <- file.path(tempdir(), "matrix.json")
mx_client_save(list(server = "https://matrix.example.org",
                    token = "syt_example",
                    user_id = "@bot:example.org",
                    device_id = "DEVICEID"),
               path = path)
unlink(path)
```

---

mx_client_session *Build an mx.api session from a client config*

Description

Build an mx.api session from a client config

Usage

```
mx_client_session(client)
```

Arguments

client	Named list or "mx_client_config" with server, token, user_id, and device_id.

Value

An "mx_session" from **mx.api**.

Examples

```
s <- mx_client_session(list(server = "https://matrix.example.org",
                            token = "syt_example",
                            user_id = "@bot:example.org",
                            device_id = "DEVICEID"))
class(s)
```

mx_crypto_account *Load or create this client's Olm account*

Description

Unpickles `account.pickle` from the store, or mints a fresh account and persists it. The account holds the device's long-lived Curve25519/Ed25519 identity keys.

Usage

```
mx_crypto_account(store_dir)
```

Arguments

`store_dir`	Character. Crypto store directory.

Value

An mx.crypto account handle.

Examples

```
if (requireNamespace("mx.crypto", quietly = TRUE)) {
  store <- mx_crypto_store_dir("myapp", path = tempfile())
  acct <- mx_crypto_account(store)
  unlink(store, recursive = TRUE)
}
```

mx_crypto_account_save *Persist an Olm account to the store*

Description

Persist an Olm account to the store

Usage

```
mx_crypto_account_save(account, store_dir)
```

Arguments

`account`	An mx.crypto account handle.
`store_dir`	Character. Crypto store directory.

Value

The pickle path, invisibly.

Examples

```
if (requireNamespace("mx.crypto", quietly = TRUE)) {
  store <- mx_crypto_store_dir("myapp", path = tempfile())
  acct <- mx_crypto_account(store)
  mx_crypto_account_save(acct, store)
  unlink(store, recursive = TRUE)
}
```

---

mx_crypto_claim_otks *Claim a one-time key for each device*

Description

Calls `/keys/claim` and attaches the claimed key to each device as `$otk`, ready for `mx_crypto_encrypt_for_devices()`.

Usage

```
mx_crypto_claim_otks(client, devices)
```

Arguments

`client`	Matrix client config.
`devices`	List of devices from `mx_crypto_known_devices()`.

Details

Each claimed key's signature is checked against the device's Ed25519 key, which must itself have come from a verified `device_keys` (that is, from `mx_crypto_known_devices()`). A key that fails is dropped with a warning and its device comes back with no `otk`.

Value

The devices with an `otk` field added where one was claimed and verified.

Examples

```
## Not run:
devs <- mx_crypto_claim_otks(client, mx_crypto_known_devices(client, uid))

## End(Not run)
```

mx_crypto_decrypt_event

Decrypt an m.room.encrypted event (Megolm)

Description

Decrypt an m.room.encrypted event (Megolm)

Usage

```
mx_crypto_decrypt_event(inbound_session, encrypted)
```

Arguments

inbound_session	An inbound Megolm session for the event's session_id.
encrypted	The m.room.encrypted event content.

Value

The decrypted event payload (a parsed list).

Examples

```
## Not run:
ev <- mx_crypto_decrypt_event(inb, encrypted_content)
ev$content$body

## End(Not run)
```

mx_crypto_device_keys *Build a signed device_keys object for upload*

Description

Produces the device_keys structure /keys/upload expects: the device's public identity keys plus an Ed25519 signature over their canonical JSON. Hand the result to mx.api::mx_keys_upload().

Usage

```
mx_crypto_device_keys(account, user_id, device_id)
```

Arguments

account	An mx.crypto account handle.
user_id	Character. Full Matrix user id.
device_id	Character. This device's id.

Value

A named list ready to upload.

Examples

```
if (requireNamespace("mx.crypto", quietly = TRUE)) {
  store <- mx_crypto_store_dir("myapp", path = tempfile())
  acct <- mx_crypto_account(store)
  dk <- mx_crypto_device_keys(acct, "@bot:example.org", "DEVICEID")
  unlink(store, recursive = TRUE)
}

## Not run:
# Uploading needs a live homeserver session:
mx.api::mx_keys_upload(session, device_keys = dk)

## End(Not run)
```

---

mx_crypto_encrypt_event

Encrypt event content for a room (Megolm)

Description

Returns the `m.room.encrypted` content to send as the event body.

Usage

```
mx_crypto_encrypt_event(megolm_out, content, room_id, sender_curve25519,
                        device_id)
```

Arguments

`megolm_out`	An outbound Megolm session.
`content`	Named list. The plaintext event content (e.g. an `m.room.message`).
`room_id`	Character. Room id.
`sender_curve25519`	Character. This device's Curve25519 key.
`device_id`	Character. This device's id.

Value

A named list: `m.room.encrypted` content.

Examples

```
## Not run:
enc <- mx_crypto_encrypt_event(megolm_out,
  list(msgtype = "m.text", body = "hi"), "!room:ex", my_curve, "DEV")

## End(Not run)
```

`mx_crypto_encrypt_for_devices` *Encrypt an event for an encrypted room's devices*

Description

Ensures an outbound Megolm session for the room, shares its key with any recipient device that has not received it yet (establishing an Olm session, claiming a one-time key when needed), and encrypts the event. Returns the to-device payloads and the `m.room.encrypted` event; the caller sends them with `mx.api::mx_send_to_device()` and `mx.api::mx_send()`.

Usage

```
mx_crypto_encrypt_for_devices(account, sessions, room_id, content,
                              sender_curve25519, device_id,
                              recipients = list(), sender_user_id = NULL)
```

Arguments

`account`	An mx.crypto account handle.
`sessions`	A session set.
`room_id`	Character room id.
`content`	Named list. Plaintext event content.
`sender_curve25519`	Character. This device's Curve25519 key.
`device_id`	Character. This device's id.
`recipients`	List of recipient devices, each a list with `user_id`, `device_id`, `curve25519`, `ed25519`, and (only needed to open a new Olm session) `otk`, a claimed one-time key. Use `mx_crypto_known_devices()`, which returns exactly this shape for devices whose keys verified.
`sender_user_id`	Character. This user's Matrix id. Required to build a spec-conformant Olm payload that the recipient can attribute.

Value

List with `to_device` (per-device payloads), `event` (the `m.room.encrypted` content), and the updated `sessions`.

Examples

```
if (requireNamespace("mx.crypto", quietly = TRUE)) {
  acct <- mx.crypto::mxc_account_new()
  out <- mx_crypto_encrypt_for_devices(
    acct, mx_crypto_sessions_new(), "!r:ex",
    list(msgtype = "m.text", body = "hi"),
    mx.crypto::mxc_account_identity_keys(acct)$curve25519, "DEV",
    recipients = list(), sender_user_id = "@me:ex")
  names(out)
}
```

`mx_crypto_handle_to_device` *Decrypt an inbound Olm to-device payload*

Description

Accepts an `m.room.encrypted` to-device content addressed to this device and returns the decrypted event. When it is an `m.room_key`, the caller builds an inbound Megolm session from `content$session_key` with `mx_crypto_inbound_session()`.

Usage

```
mx_crypto_handle_to_device(account, my_curve25519, content, self_id = NULL,
                           self_ed25519 = NULL, devices = NULL)
```

Arguments

`account`	An mx.crypto account handle.
`my_curve25519`	Character. This device's Curve25519 key.
`content`	The to-device `m.room.encrypted` content.
`self_id`	Character or NULL. This user's Matrix id. When NULL the recipient user-id check is skipped; pass it whenever it is known.
`self_ed25519`	Character or NULL. This device's Ed25519 key. Defaults to the account's own key.
`devices`	List of verified devices from `mx_crypto_known_devices()`, or NULL. Used to tie the payload's claimed sender to a device whose keys were verified. The result carries `sender_bound`, which is FALSE when no list is supplied or nothing matches; an unbound sender identity is a claim, not a fact, because anyone can open an Olm session to this device.

Details

The decrypted payload is checked against this device's identity before it is returned: a message that does not name us as recipient is dropped, because decrypting successfully only proves it was encrypted to our key, not that it was meant for us here.

Value

The decrypted event (a parsed list) with a `sender_bound` flag, or NULL if it was not for us or failed the recipient checks.

Examples

```
## Not run:
ev <- mx_crypto_handle_to_device(acct, my_curve, td_content,
                                 self_id = "@me:example.org",
                                 devices = mx_crypto_known_devices(cl, uid))
if (identical(ev$type, "m.room_key") && isTRUE(ev$sender_bound)) {
  inb <- mx_crypto_inbound_session(ev$content$session_key)
}

## End(Not run)
```

`mx_crypto_inbound_session` *Build an inbound Megolm session from a shared room key*

Description

Build an inbound Megolm session from a shared room key

Usage

```
mx_crypto_inbound_session(session_key)
```

Arguments

`session_key`	Character. The `session_key` from an `m.room_key` event.

Value

An inbound Megolm session.

Examples

```
## Not run:
inb <- mx_crypto_inbound_session(ev$content$session_key)

## End(Not run)
```

mx_crypto_known_devices

List the devices (and identity keys) of some users

Description

Queries `/keys/query` and flattens the result to a list of devices.

Usage

```
mx_crypto_known_devices(client, user_ids)
```

Arguments

`client`	Matrix client config.
`user_ids`	Character vector of Matrix user ids.

Details

Devices are returned only if their Ed25519 self-signature verifies against the identity the homeserver claims for them. A device that fails is dropped with a warning naming it, and the remaining devices are still returned: one bad device must not make a room unusable.

Value

List of verified devices, each `list(user_id, device_id, curve25519, ed25519)`.

Examples

```
## Not run:
mx_crypto_known_devices(client, "@bob:example.org")

## End(Not run)
```

mx_crypto_process_sync

Process a sync response: store room keys, decrypt room events

Description

Handles inbound to-device `m.room.encrypted` (Olm) messages, storing any `m.room_key` as an inbound Megolm session, then decrypts `m.room.encrypted` timeline events whose session is known. Returns normalized text events in the same shape as `mx_extract_text_events()`, plus the updated session set.

Usage

```
mx_crypto_process_sync(account, sessions, sync_resp, self_curve25519,
                       self_id = NULL, devices = NULL)
```

Arguments

account	An mx.crypto account handle.
sessions	A session set.
sync_resp	Parsed /sync response.
self_curve25519	Character. This device's Curve25519 key.
self_id	Character or NULL. This user's Matrix id, for is_self tagging.
devices	List of verified devices from mx_crypto_known_devices(), or NULL. Room keys arrive over Olm carrying a claimed sender, and anyone who can reach this device can send one, so the claim is only worth something once it is matched against a device whose device_keys verified. Without this list decrypted events always report sender_verified = FALSE: they still decrypt, but nothing attests to who sent them.

Value

List with events (decrypted, normalized) and the updated sessions.

Examples

```
if (requireNamespace("mx.crypto", quietly = TRUE)) {
  acct <- mx.crypto::mxc_account_new()
  res <- mx_crypto_process_sync(acct, mx_crypto_sessions_new(),
    list(to_device = list(events = list()), rooms = list(join = list())),
    mx.crypto::mxc_account_identity_keys(acct)$curve25519)
  length(res$events)
}
```

---

mx_crypto_publish_keys

Publish this device's identity and one-time keys

Description

Builds and signs the device keys and a batch of one-time keys, uploads them with mx.api::mx_keys_upload(), marks them published, and persists the account. Call once after login and again to replenish one-time keys.

Usage

```
mx_crypto_publish_keys(client, account, store_dir, n_otks = 50L)
```

Arguments

client	Matrix client config (needs user_id, device_id).
account	An mx.crypto account handle.
store_dir	Character. Crypto store directory.
n_otks	Integer. Number of one-time keys to publish.

Value

The /keys/upload response, invisibly.

Examples

```
## Not run:
acct <- mx_crypto_account(mx_crypto_store_dir("corteza"))
mx_crypto_publish_keys(mx_client_load(app = "corteza"), acct,
                       mx_crypto_store_dir("corteza"))

## End(Not run)
```

---

mx_crypto_room_key_payload

Encrypt a Megolm room key to one device as a to-device payload

---

Description

Wraps the outbound Megolm session's key in an m.room_key event, Olm-encrypts it to the recipient device, and returns the m.room.encrypted to-device content to hand to mx.api::mx_send_to_device().

Usage

```
mx_crypto_room_key_payload(olm_session, sender_curve25519,
                           recipient_curve25519, room_id, megolm_out,
                           sender_user_id, sender_ed25519, recipient_user_id,
                           recipient_ed25519)
```

Arguments

olm_session	An outbound Olm session (mx.crypto::mxc_olm_create_outbound()).
sender_curve25519	Character. This device's Curve25519 key.
recipient_curve25519	Character. Target device's Curve25519 key.
room_id	Character. Room the key is for.
megolm_out	An outbound Megolm session.
sender_user_id	Character. This user's Matrix id.

`sender_ed25519` Character. This device's Ed25519 key.

`recipient_user_id`
Character. Target user's Matrix id.

`recipient_ed25519`
Character. Target device's Ed25519 key.

Value

A named list: the to-device `m.room.encrypted` content.

Examples

```
## Not run:
content <- mx_crypto_room_key_payload(olm, my_curve, their_curve,
                                      "!room:ex", megolm_out,
                                      "@me:ex", my_ed, "@them:ex", their_ed)

## End(Not run)
```

---

mx_crypto_sessions_load

Load a session set from the crypto store

Description

Load a session set from the crypto store

Usage

```
mx_crypto_sessions_load(store_dir)
```

Arguments

`store_dir` Character. Crypto store directory.

Value

A session set (empty if nothing is stored yet).

Examples

```
if (requireNamespace("mx.crypto", quietly = TRUE)) {
  s <- mx_crypto_sessions_load(file.path(tempfile(), "crypto"))
}
```

mx_crypto_sessions_new

Create an empty E2EE session set

Description

Create an empty E2EE session set

Usage

```
mx_crypto_sessions_new()
```

Value

A session set: named lists `olm`, `olm_in`, `megolm_out`, `megolm_in`.

Examples

```
s <- mx_crypto_sessions_new()
names(s)
```

mx_crypto_sessions_save

Persist a session set to the crypto store

Description

Pickles every live session (encrypted at rest with the store key) into `sessions.json`. Reload with `mx_crypto_sessions_load()`.

Usage

```
mx_crypto_sessions_save(sessions, store_dir)
```

Arguments

`sessions`	A session set.
`store_dir`	Character. Crypto store directory.

Value

The path written, invisibly.

Examples

```
if (requireNamespace("mx.crypto", quietly = TRUE)) {
  dir <- file.path(tempfile(), "crypto")
  mx_crypto_sessions_save(mx_crypto_sessions_new(), dir)
}
```

`mx_crypto_store_dir` *Directory holding this client's encryption state*

Description

The crypto store keeps the pickled Olm account, the 32-byte key that encrypts those pickles at rest, and (later) per-peer Olm and per-room Megolm sessions. It lives beside the JSON config under `tools::R_user_dir()`.

Usage

```
mx_crypto_store_dir(app = "mx.client", path = NULL)
```

Arguments

`app`	Character. Application namespace.
`path`	Character or NULL. Explicit store directory.

Value

Character directory path.

Examples

```
mx_crypto_store_dir("myapp", path = tempfile())
```

`mx_extract_invites` *Extract pending invite room ids from a sync response*

Description

Extract pending invite room ids from a sync response

Usage

```
mx_extract_invites(sync_resp)
```

Arguments

sync_resp	Parsed /sync response.

Value

Character vector of invited room ids.

Examples

```
sync_resp <- list(rooms = list(invite = list("!inv:example.org" = list())))
mx_extract_invites(sync_resp)
```

---

mx_extract_reaction_verdict

Extract a reaction approval verdict from sync events

Description

Scans a room timeline for a reaction on target_event_id from someone other than self_id. Returns TRUE for approval keys, FALSE for denial keys, or NULL when no verdict is present.

Usage

```
mx_extract_reaction_verdict(sync_resp, room_id, self_id, target_event_id,
                            approve_keys = NULL, deny_keys = NULL)
```

Arguments

sync_resp	Parsed /sync response.
room_id	Character room id.
self_id	Current user's Matrix id.
target_event_id	Event id being reacted to.
approve_keys	Character vector of reaction keys read as approval. NULL (default) uses thumbs-up (U+1F44D), check-mark (U+2705), and "y"/"yes"/"ok".
deny_keys	Character vector of reaction keys read as denial. NULL (default) uses thumbs-down (U+1F44E), cross-mark (U+274C), and "n"/"no"/"nope".

Value

TRUE, FALSE, or NULL.

Examples

```
sync_resp <- list(rooms = list(join = list("!room:example.org" = list(
    timeline = list(events = list(list(type = "m.reaction",
        sender = "@alice:example.org",
        content = list("m.relates_to" = list(rel_type = "m.annotation",
            event_id = "$msg", key = "yes")))))))))
mx_extract_reaction_verdict(sync_resp, "!room:example.org",
                            self_id = "@bot:example.org",
                            target_event_id = "$msg")
```

`mx_extract_text_events` *Extract text message events from a sync response*

Description

Walks joined-room timeline events and returns normalized text-message records. Self events are retained and tagged with `is_self`.

Usage

```
mx_extract_text_events(sync_resp, self_id, msgtypes = "m.text")
```

Arguments

`sync_resp`	Parsed `/sync` response.
`self_id`	Current user's Matrix id.
`msgtypes`	Character vector of message types to include.

Value

List of normalized event records, each carrying `room_id`, `event_id`, `sender`, `is_self`, `body`, `msgtype`, `ts` (the event's `origin_server_ts`, in milliseconds since the epoch, or NULL when the server omits it), and `mentions`.

Examples

```
sync_resp <- list(rooms = list(join = list("!room:example.org" = list(
    timeline = list(events = list(list(type = "m.room.message",
        event_id = "$1", sender = "@alice:example.org",
        content = list(msgtype = "m.text", body = "hello")))))))))
mx_extract_text_events(sync_resp, self_id = "@bot:example.org")
```

mx_markdown_to_html *Convert a conservative markdown subset to Matrix custom HTML*

Description

Supports headings, bullets, numbered lists, fenced code blocks, inline code, bold, simple underscore emphasis, and GitHub-style pipe tables.

Usage

```
mx_markdown_to_html(text)
```

Arguments

text	Character markdown body.

Value

Character HTML suitable for m.room.message formatted_body.

Examples

```
mx_markdown_to_html("# Status\n- built\n- checked\n\nShip `0.1.0` **soon**")
```

mx_pill_mentions *Turn textual @mentions into Matrix pills in formatted HTML*

Description

Turn textual @mentions into Matrix pills in formatted HTML

Usage

```
mx_pill_mentions(html, user_ids)
```

Arguments

html	Character HTML (e.g. from mx_markdown_to_html).
user_ids	Character Matrix user ids, such as "@jorge:example.org".

Value

HTML with textual @localpart occurrences replaced by matrix.to links. Unmatched user ids leave the HTML unchanged; they can still be placed in m.mentions by mx_send_text().

Examples

```
mx_pill_mentions("<p>ping @jorge</p>", "@jorge:example.org")
```

mx_resolve_room *Resolve a room id, name, or default room*

Description

Resolution order: literal room IDs beginning with `!`, a supplied `room_cache` name-to-id map, joined-room display-name lookup, then the config's `room_id` fallback.

Usage

```
mx_resolve_room(client, room = NULL, room_cache = NULL, fallback = TRUE,
                details = FALSE, quiet = FALSE)
```

Arguments

`client`	Matrix client config.
`room`	Character or NULL.
`room_cache`	Named list or character vector mapping names to ids.
`fallback`	Logical. Use `client$room_id` when lookup misses.
`details`	Logical. Return source metadata instead of just the id.
`quiet`	Logical. Suppress fallback message.

Value

Character room id, or a list when `details = TRUE`.

Examples

```
client <- list(room_id = "!default:example.org")
# Literal ids and cache hits resolve without a server round-trip:
mx_resolve_room(client, "!abc:example.org")
mx_resolve_room(client, "general",
                room_cache = list(general = "!gen:example.org"))
mx_resolve_room(client) # NULL room falls back to the config default
```

mx_room_encrypted *Is a room end-to-end encrypted?*

Description

Resolves the room (by name, id, or the config default) and reads its `m.room.encryption` state. Needs mx.api >= 0.3.0.

Usage

```
mx_room_encrypted(client, room = NULL, room_cache = NULL)
```

Arguments

client	Matrix client config.
room	Character room id/name or NULL for the default room.
room_cache	Optional room name-to-id cache.

Value

TRUE when the room advertises an encryption algorithm, FALSE otherwise.

Examples

```
## Not run:
if (mx_room_encrypted(client, "secret plans")) {
    # use mx_send_encrypted() instead of mx_send_text()
}

## End(Not run)
```

---

mx_room_lookup_by_name

Look up a joined room by display name

Description

Look up a joined room by display name

Usage

```
mx_room_lookup_by_name(client, name)
```

Arguments

client	Matrix client config.
name	Character room name.

Value

Room id, or NULL when no joined room has that name.

Examples

```
## Not run:
# Needs a live homeserver session.
mx_room_lookup_by_name(mx_client_load("myapp"), "general")

## End(Not run)
```

`mx_send_encrypted` *Send an end-to-end encrypted message to a room*

Description

Discovers the room members' devices (unless `recipients` is given), claims one-time keys for any without an Olm session, shares the room key over to-device, encrypts the content with Megolm, sends the `m.room.encrypted` event, and persists the updated sessions.

Usage

```
mx_send_encrypted(client, account, sessions, room_id, content, store_dir,
                  recipients = NULL, member_ids = NULL)
```

Arguments

`client`	Matrix client config.
`account`	An mx.crypto account handle.
`sessions`	A session set (see `mx_crypto_sessions_new()`).
`room_id`	Character room id.
`content`	Named list. Plaintext event content.
`store_dir`	Character. Crypto store directory.
`recipients`	List of recipient devices, or NULL to discover them from `member_ids`.
`member_ids`	Character vector of room member user ids (used when `recipients` is NULL).

Value

List with `event_id` and the updated `sessions`.

Examples

```
## Not run:
res <- mx_send_encrypted(client, acct, sessions, "!r:ex",
  list(msgtype = "m.text", body = "secret"), store,
  member_ids = "@bob:example.org")

## End(Not run)
```

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

Description

Client-layer wrapper over `mx.api::mx_send_media()`: resolves the room by name (or falls back to the config's default room), builds the session from the client config, and uploads + posts in one call. The msgtype is derived from the file's MIME type unless given.

Usage

```
mx_send_media(client, path, room = NULL, body = basename(path), msgtype = NULL,
              content_type = NULL, info = list(), room_cache = NULL,
              dry_run = FALSE)
```

Arguments

`client`	Matrix client config.
`path`	Character. Path to the file to upload.
`room`	Character room id/name or NULL for the default room.
`body`	Character. Message body / filename shown by clients.
`msgtype`	Character or NULL. NULL derives it from the MIME type.
`content_type`	Character or NULL. MIME type override for files whose extension guesses wrong (tempfiles, odd extensions); NULL guesses from the extension.
`info`	List. Extra fields merged into the media `info`.
`room_cache`	Optional room name-to-id cache.
`dry_run`	Logical. Print instead of uploading/sending.

Details

If you attach mx.api and mx.client together, namespace-qualify – the two packages export an `mx_send_media` each (session-first there, client-first here).

Value

Event id, or NULL on dry-run.

Examples

```
client <- list(room_id = "!default:example.org")
png <- file.path(tempdir(), "plot.png")
file.create(png)
mx_send_media(client, png, dry_run = TRUE)
unlink(png)
```

mx_send_table *Send tabular data to a Matrix room*

Description

Sends a plain-text fallback body plus Matrix custom HTML table in `formatted_body`. This bypasses Markdown entirely.

Usage

```
mx_send_table(client, x, room = NULL, header = TRUE, title = NULL,
              room_cache = NULL, dry_run = FALSE)
```

Arguments

`client`	Matrix client config.
`x`	A data frame, matrix, or list coercible to a data frame.
`room`	Character room id/name or NULL for the default room.
`header`	Logical. Include a header row using column names.
`title`	Optional text prepended to the plain fallback body.
`room_cache`	Optional room name-to-id cache.
`dry_run`	Logical. Print instead of sending.

Value

Event id, or NULL on dry-run.

Examples

```
client <- list(room_id = "!default:example.org")
mx_send_table(client, data.frame(A = 1, B = 2), dry_run = TRUE)
```

mx_send_text *Send plain text to a Matrix room*

Description

Send plain text to a Matrix room

Usage

```
mx_send_text(client, text, room = NULL, msgtype = "m.text", room_cache = NULL,
             dry_run = FALSE, markdown = FALSE, mentions = NULL)
```

Arguments

client	Matrix client config.
text	Character message body.
room	Character room id/name or NULL for the default room.
msgtype	Character Matrix message type.
room_cache	Optional room name-to-id cache.
dry_run	Logical. Print instead of sending.
markdown	Logical. If TRUE, include Matrix custom HTML derived from a conservative markdown subset.
mentions	Character vector of Matrix user ids to mention (e.g. "@jorge:cornball.ai"). Each id is added to the event's m.mentions (so the user is notified) and any textual @localpart in the body becomes a matrix.to pill in the HTML. Implies an HTML formatted body even when markdown is FALSE – pills only render from HTML.

Value

Event id, or NULL on dry-run.

Examples

```
client <- list(room_id = "!default:example.org")
mx_send_text(client, "release is out", dry_run = TRUE)
## Not run:
# A real send needs a live homeserver session:
client <- mx_client_load("myapp")
mx_send_text(client, "release is out", markdown = TRUE,
             mentions = "@jorge:example.org")

## End(Not run)
```

---

mx_set_displayname *Set the account's profile display name*

Description

Client-level wrapper over mx.api::mx_set_displayname(): builds the session from the client config and retries once through mx_with_relogin when the homeserver rejects a rotated token. The display name is account-global and shows in the sender line of every message the account posts.

Usage

```
mx_set_displayname(client, name, save = TRUE)
```

Arguments

client	Matrix client config.
name	Character. New display name.
save	Logical. On a relogin, persist the refreshed token to the client's config path (see mx_client_relogin).

Value

TRUE, invisibly.

Examples

```
## Not run:
# A real rename needs a live homeserver session:
client <- mx_client_load("myapp")
mx_set_displayname(client, "mybot (maintenance)")

## End(Not run)
```

---

mx_sync_update *Sync once and update the stored cursor*

Description

Calls mx.api::mx_sync() using client$sync_token, stores the returned next_batch in a returned client object, and optionally saves it back to disk.

Usage

```
mx_sync_update(client, timeout = 0L, filter = NULL, save = TRUE, path = NULL,
               app = NULL)
```

Arguments

client	Matrix client config.
timeout	Integer long-poll timeout in milliseconds.
filter	Character or NULL. Matrix sync filter.
save	Logical. Persist the updated client config.
path	Character or NULL. Save destination.
app	Character or NULL. Application namespace for default saves.

Value

List with sync, client, and first_run.

Examples

```
## Not run:
# Needs a live homeserver session.
client <- mx_client_load("myapp")
res <- mx_sync_update(client, timeout = 30000L)
events <- mx_extract_text_events(res$sync, client$user_id)

## End(Not run)
```

`mx_table_html` *Render tabular data as Matrix custom HTML*

Description

Produces the conservative table shape rendered by Matrix clients such as FluffyChat 2.6.0+: a bare `<table>` containing `<tr>`, `<th>`, and `<td>` nodes. No CSS, colspan, rowspan, or custom attributes are emitted.

Usage

```
mx_table_html(x, header = TRUE)
```

Arguments

`x`	A data frame, matrix, or list coercible to a data frame.
`header`	Logical. Include a header row using column names.

Value

Character HTML suitable for Matrix `formatted_body`.

Examples

```
mx_table_html(data.frame(A = 1:2, B = c("x", "y")))
```

`mx_with_relogin` *Run a client operation, re-logging in once on an expired token*

Description

Calls `fn(client)`; if it fails with the server's invalid-token error (`M_UNKNOWN_TOKEN`, signalled as a classed condition by mx.api >= 0.3.0), re-logs in via `mx_client_relogin` and retries once with the refreshed client. Any other error propagates.

Usage

```
mx_with_relogin(client, fn, save = TRUE)
```

Arguments

client	Matrix client config with password.
fn	Function taking a client config.
save	Logical. Persist the refreshed config on relogin.

Value

fn's return value.

Examples

```
## Not run:
mx_with_relogin(client, function(cl) {
    mx_send_text(cl, "still here after a token rotation")
})

## End(Not run)
```

---

print.mx_client_config

Print a Matrix client config

Description

Prints the config field by field with credentials masked, so that an interactive session, a screenshot, or a pasted bug report does not leak the access token or password. Secret fields show whether they are set (<hidden>) or empty (<unset>) without showing the value. Use unclass(x) or str(unclass(x)) when the raw credentials are genuinely needed.

Usage

```
## S3 method for class 'mx_client_config'
print(x, ...)
```

Arguments

x	An mx_client_config, as returned by mx_client_load() or mx_client_from_config().
...	Ignored.

Value

x, invisibly.

Examples

```
cfg <- mx_client_from_config(list(server = "https://matrix.example.org",
                                  token = "syt_secret_value",
                                  user_id = "@bot:example.org",
                                  device_id = "DEVICEID"))
cfg
```

Index

mx.client *(mx.client-package)*, 3
mx.client-package, 3
mx_accept_invites, 5
mx_client_config_path, 6
mx_client_configure, 5
mx_client_from_config, 7
mx_client_legacy_config_path, 7
mx_client_load, 8
mx_client_relogin, 9, *33, 34*
mx_client_save, 9
mx_client_session, 10
mx_crypto_account, 11
mx_crypto_account_save, 11
mx_crypto_claim_otks, 12
mx_crypto_decrypt_event, 13
mx_crypto_device_keys, 13
mx_crypto_encrypt_event, 14
mx_crypto_encrypt_for_devices, 15
mx_crypto_handle_to_device, 16
mx_crypto_inbound_session, 17
mx_crypto_known_devices, 18
mx_crypto_process_sync, 18
mx_crypto_publish_keys, 19
mx_crypto_room_key_payload, 20
mx_crypto_sessions_load, 21
mx_crypto_sessions_new, 22
mx_crypto_sessions_save, 22
mx_crypto_store_dir, 23
mx_extract_invites, 23
mx_extract_reaction_verdict, 24
mx_extract_text_events, 25
mx_markdown_to_html, *26*, 26
mx_pill_mentions, 26
mx_resolve_room, 27
mx_room_encrypted, 27
mx_room_lookup_by_name, 28
mx_send_encrypted, 29
mx_send_media, 30
mx_send_table, 31
mx_send_text, 31
mx_set_displayname, 32
mx_sync_update, 33
mx_table_html, 34
mx_with_relogin, *9*, *32*, 34

print.mx_client_config, 35