

Package: mx.client (via r-universe)

June 19, 2026

Type Package

Title Stateful Matrix Client Helpers

Version 0.1.1

Date 2026-06-13

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, simplermardown, tinytest

VignetteBuilder simplermardown

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 2026-06-19 15:50:07 UTC

RemoteUrl <https://github.com/cran/mx.client>

RemoteRef HEAD

RemoteSha f65dd1f9988e54ed78770f49b93f0334f54ab273

Contents

mx.client-package	3
mx_accept_invites	5
mx_client_config_path	5
mx_client_configure	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	17
mx_crypto_process_sync	18
mx_crypto_publish_keys	19
mx_crypto_room_key_payload	19
mx_crypto_sessions_load	20
mx_crypto_sessions_new	21
mx_crypto_sessions_save	21
mx_crypto_store_dir	22
mx_extract_invites	23
mx_extract_reaction_verdict	23
mx_extract_text_events	24
mx_markdown_to_html	25
mx_pill_mentions	25
mx_resolve_room	26
mx_room_encrypted	27
mx_room_lookup_by_name	27
mx_send_encrypted	28
mx_send_media	29
mx_send_text	30
mx_sync_update	31
mx_with_relogin	32

Index

33

 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

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

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_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_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_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	<i>Load a Matrix client config</i>
----------------	------------------------------------

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	<i>Re-login with stored credentials and refresh the saved token</i>
-------------------	---

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

<code>client</code>	Matrix client config with password.
<code>save</code>	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	<i>Save a Matrix client config</i>
----------------	------------------------------------

Description

Writes JSON with mode 0600.

Usage

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

Arguments

<code>client</code>	Named list or "mx_client_config".
<code>app</code>	Character or NULL. Application namespace.
<code>path</code>	Character or NULL. Destination path.

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().

Value

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

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())
```

Arguments

<code>account</code>	An <code>mx.crypto</code> account handle.
<code>sessions</code>	A session set.
<code>room_id</code>	Character room id.
<code>content</code>	Named list. Plaintext event content.
<code>sender_curve25519</code>	Character. This device's Curve25519 key.
<code>device_id</code>	Character. This device's id.
<code>recipients</code>	List of recipient devices, each a list with <code>user_id</code> , <code>device_id</code> , <code>curve25519</code> , and (only needed to open a new Olm session) <code>otk</code> , a claimed one-time key.

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())
  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)
```

Arguments

account	An mx.crypto account handle.
my_curve25519	Character. This device's Curve25519 key.
content	The to-device m.room.encrypted content.

Value

The decrypted event (a parsed list), or NULL if not for us.

Examples

```

## Not run:
ev <- mx_crypto_handle_to_device(acct, my_curve, td_content)
if (identical(ev$type, "m.room_key")) {
  inb <- mx_crypto_inbound_session(ev$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.

Value

List of 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)
```

Arguments

<code>account</code>	An <code>mx.crypto</code> account handle.
<code>sessions</code>	A session set.
<code>sync_resp</code>	Parsed /sync response.
<code>self_curve25519</code>	Character. This device's Curve25519 key.
<code>self_id</code>	Character or NULL. This user's Matrix id, for <code>is_self</code> tagging.

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

<code>client</code>	Matrix client config (needs <code>user_id</code> , <code>device_id</code>).
<code>account</code>	An <code>mx.crypto</code> account handle.
<code>store_dir</code>	Character. Crypto store directory.
<code>n_otks</code>	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)
```

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.

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)

## 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, megoim_out, megoim_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.

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, and simple underscore emphasis.

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.to pills*

Description

Replaces each occurrence of @localpart (or the full @localpart:server id) in already-rendered HTML with a matrix.to anchor, which Matrix clients render as a mention pill. A user id with no textual occurrence is left to m.mentions alone, which still notifies.

Usage

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

Arguments

html Character HTML (e.g. from mx_markdown_to_html).

user_ids Character vector of full Matrix user ids.

Value

Character HTML with mention pills.

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.encrypted 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	<i>Send an end-to-end encrypted message to a room</i>
-------------------	---

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	<i>Send a media file to a Matrix room</i>
---------------	---

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_text	<i>Send plain text to a Matrix room</i>
--------------	---

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_sync_update	<i>Sync once and update the stored cursor</i>
----------------	---

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

<code>client</code>	Matrix client config.
<code>timeout</code>	Integer long-poll timeout in milliseconds.
<code>filter</code>	Character or NULL. Matrix sync filter.
<code>save</code>	Logical. Persist the updated client config.
<code>path</code>	Character or NULL. Save destination.
<code>app</code>	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_with_relogin	<i>Run a client operation, re-logging in once on an expired token</i>
-----------------	---

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

<code>client</code>	Matrix client config with password.
<code>fn</code>	Function taking a client config.
<code>save</code>	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)
```

Index

`mx.client` (`mx.client-package`), 3
`mx.client-package`, 3
`mx_accept_invites`, 5
`mx_client_config_path`, 5
`mx_client_configure`, 6
`mx_client_from_config`, 7
`mx_client_legacy_config_path`, 7
`mx_client_load`, 8
`mx_client_relogin`, 9, 32
`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`, 17
`mx_crypto_process_sync`, 18
`mx_crypto_publish_keys`, 19
`mx_crypto_room_key_payload`, 19
`mx_crypto_sessions_load`, 20
`mx_crypto_sessions_new`, 21
`mx_crypto_sessions_save`, 21
`mx_crypto_store_dir`, 22
`mx_extract_invites`, 23
`mx_extract_reaction_verdict`, 23
`mx_extract_text_events`, 24
`mx_markdown_to_html`, 25, 25
`mx_pill_mentions`, 25
`mx_resolve_room`, 26
`mx_room_encrypted`, 27
`mx_room_lookup_by_name`, 27
`mx_send_encrypted`, 28
`mx_send_media`, 29
`mx_send_text`, 30
`mx_sync_update`, 31
`mx_with_relogin`, 9, 32