Workgroup:
More Instant Messaging Interoperability
Internet-Draft:
draft-ralston-mimi-matrix-message-format-01
Published: 9 May 2023
Intended Status: Informational
Expires: 10 November 2023
Authors: T. Ralston
The Matrix.org Foundation C.I.C.
M. Hodgson
The Matrix.org Foundation C.I.C.

Matrix Message Format

Abstract

This document describes the Matrix event format for use over a delivery protocol like Linearized Matrix.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://turt2live.github.io/ietf-mimi-matrix-message-format/draft-ralston-mimi-matrix-message-format.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ralston-mimi-matrix-message-format/.

Discussion of this document takes place on the More Instant Messaging Interoperability Working Group mailing list (mailto:mimi@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/mimi/. Subscribe at https://www.ietf.org/mailman/listinfo/mimi/.

Source for this draft and an issue tracker can be found at https://github.com/turt2live/ietf-mimi-matrix-message-format.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Interoperable instant messaging requires a common format for all participants to contribute to the conversation or state of the room. Matrix is an open protocol for interoperable, decentralized, secure communication, and includes a "linearized" API surface to abstract away the complexities of DAGs, described by [I-D.ralston-mimi-linearized-matrix]. Matrix defines a rich taxonomy of arbitrarily extensible payloads of information called "events" to carry information between machines and users, which may in turn be layered over end-to-end encryption. Matrix events have been designed for interoperability from the outset between heterogenous messaging platforms, and define a real-world highest-common denominator set of message types, including:

*Instant messages (plain & rich text)

*End-to-end encrypted payloads

*File transfer

*Reactions (emoji, textual, image-based)

*Edits

*Replies

*Deletions

*Stickers

*Custom emoji

*Voice messages

*Polls

*Static location share

*Live location share (ephemeral)

*Live location share (persistent)

*Spoiler text

*Threaded messages

*Typing notifications

*Read receipts

*Read-up-to markers

*Presence

*1:1 VoIP signalling

*Multiparty VoIP signalling

Matrix events are extensible, and proposals exist for additional event formats ranging from attaching 3D world geometry to conversations (for openly standardized metaverse communication) through to transferring healthcare data (FHIR).

2. Matrix Events

TODO: Bring spec references into the I-D, like we did with Linearized Matrix.

Events are JSON objects which by default follow the formal schemas defined in the Matrix Client Server API [MxEvents], also available as JSON Schema definitions [MxEventsSchema]. Events are extensible and may contain additional off-schema prefixed fields, or use prefixed event types not yet defined in the spec. Events then get augumented and signed by the server before being forwarded to other servers/users in the room.

These JSON objects have a few key fields:

*type: A string the client can use to determine how to render the event. This is reverse-DNS namespaced, with m. as a privileged prefix for event types formally adopted and defined within the Matrix specification.

*sender: The user ID (@alice:example.org) which sent the event.

*room_id: The room ID (!room:example.org) for where the event was sent.

*content: Type-specific JSON object.

*Other fields (TODO: define these in detail when more relevant to the doc).

Under MSC1767 [MSC1767] (a spec change proposal in the existing Matrix open standard ecosystem), callers can combine together multiple event types to provide fallback representations of an event, to provide backwards compatibility when rendering unknown event types.

An example of a simple text message would be:

```
{
    "type": "m.message",
    "content": {
        "m.text": "i am a fish",
        "m.html": "i am a <strong>fish</strong>"
    }
}
```

This can be made more complex if the sender chooses to mix in other mimetypes:

```
{
  "type": "m.message",
  "content": {
    "m.message": [
        { "mimetype": "text/html", "body": "i am a <strong>fish</strong>
        { "mimetype": "text/plain", "body": "i am a fish" },
        {
          "mimetype": "application/vnd.exampleorg.message+html",
          "body": "<content>i am a <strong>fish</strong></content>"
        }
    ]
  }
}
```

To demonstrate extensibility, a file upload [MSC3551] might look like:

```
{
  "type": "m.file",
  "content": {
    "m.text": "Upload: foo.pdf https://<download url> (12 KB)",
    "m.file": {
      "url": "mxc://example.org/abcd1234",
      "name": "foo.pdf",
      "mimetype": "application/pdf",
      "size": 12345
    }
  }
}
```

In this example, clients which do not understand m.file but do understand m.text (or m.message) would show just the plain text instead of a download button. The alternative to falling back would be to hide the unrenderable event, causing the conversation history to be fragmented: this has clear negative consequences on user experience. Instead, by defining a fallback mechanism the user is still able to participate in the conversation, though might need to ask for more information. It is expected that the "base types" (text messages, images, videos, and generic files) would be supported by all clients to ensure there are sufficient building blocks for future extensibility.

A more complete use-case for extensible events is described by "MSC3381: Polls" [MSC3381] - clients which do not yet have support for polls can present their users with text fallback for the question and the question asker can manually tally up "improper" responses (if those users simply sent text messages in response to the question). Clients which do support polls would simply show the poll and its question/options for the user to click on - their response would be sent to the room as a (deliberately) unrenderable event for other clients to tally up automatically.

3. Encryption

Matrix has specified an encryption algorithm for events called Megolm, however for the purposes of MIMI it would be desirable to adopt MLS [I-D.ietf-mls-protocol] instead. Some bookkeeping changes are required to support MLS in a decentralized environment like Matrix: those are currently defined by [DMLS].

4. Security Considerations

TODO Security. Future drafts should consider the encryption aspects in particular.

5. References

5.1. Normative References

[I-D.ietf-mls-protocol]
Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., and K. Cohn-Gordon, "The Messaging Layer Security (MLS) Protocol", Work in Progress, Internet-Draft, draft-ietf-mls-protocol-20, 27 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-mls-protocol-20>.

[MSC1767] Hodgson, M. and T. Ralston, "Extensible event types & fallback in Matrix (v2)", 2022, <https://github.com/matrix-org/matrix-spec-proposals/blob/d6046d8402e7a3c7a4fcbc9da16ea9bad5968992/proposals/1767-extensible-events.md>.

[MSC3551] Ralston, T., "Extensible Events - Files", 2021, <https://github.com/matrix-org/matrix-spec-proposals/blob/5bf2118e8ac873e7845b1eedde8dd7bc187ed673/proposals/3551-extensible-events-files.md>.

[MxEvents] The Matrix.org Foundation C.I.C., "Events | Client-Server API", 2022, <https://spec.matrix.org/v1.4/client-server-api/#events>.

[MxEventsSchema] The Matrix.org Foundation C.I.C., "Matrix Event JSON Schemas", 2022, <https://github.com/matrix-org/matrix-spec/tree/main/data/event-schemas/schema>.

5.2. Informative References

[DMLS]

Chathi, H., "Decentralised MLS", Web https://gitlab.matrix.org/matrix-org/mls-ts/-/blob/dd57bc25f6145ddedfb6d193f6baebf5133db7ed/decentralised.org, 2021, <https://gitlab.matrix.org/matrix-org/mls-ts/-/blob/dd57bc25f6145ddedfb6d193f6baebf5133db7ed/decentralised.org>.

[I-D.ralston-mimi-linearized-matrix] Ralston, T. and M. Hodgson, "Linearized Matrix API", Work in Progress, Internet-Draft, draft-ralston-mimi-linearized-matrix-00, 25 March 2023, <https://datatracker.ietf.org/doc/html/draft-ralston-mimi-linearized-matrix-00>.

[MSC3381] Ralston, T., "Polls (mk II)", 2022, <https://github.com/matrix-org/matrix-spec-proposals/blob/95fdc44b904d2b4d2f227db99050e539e43f3509/proposals/3381-polls.md>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Travis Ralston
The Matrix.org Foundation C.I.C.

Email: travisr@matrix.org

Matthew Hodgson
The Matrix.org Foundation C.I.C.

Email: matthew@matrix.org