MIMI
Internet-Draft
Intended status: Informational
Expires: 12 December 2024

R. Mahy
Rohan Mahy Consulting Services

10 June 2024

More Instant Messaging Interoperability (MIMI) message content
draft-ietf-mimi-content-04

Abstract

This document describes content semantics common in Instant Messaging (IM) systems and describes a profile suitable for instant messaging interoperability of messages end-to-end encrypted inside the MLS (Message Layer Security) Protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 12 December 2024.

Copyright Notice

Table of Contents

1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terms MLS client, MLS group, and KeyPackage have the same meanings as in the MLS protocol [RFC9420]. Other relevant terminology may be found in [I-D.ietf-mimi-arch] and [I-D.ralston-mimi-terminology].

2. Introduction

RFC EDITOR: PLEASE REMOVE THE FOLLOWING PARAGRAPH. The source for this draft is maintained in GitHub. Suggested changes should be submitted as pull requests at https://github.com/ietf-wg-mimi/draft-ietf-mimi-content (https://github.com/ietf-wg-mimi/draft-ietf-mimi-content). Editorial changes can be managed in GitHub, but any substantive change should be discussed on the MIMI mailing list (mimi@ietf.org).

MLS [RFC9420] is a group key establishment protocol motivated by the desire for group chat with efficient end-to-end encryption. While one of the motivations of MLS is interoperable standards-based secure

messaging, the MLS protocol does not define or prescribe any format for the encrypted "application messages" encoded by MLS. The development of MLS was strongly motivated by the needs of a number of Instant Messaging (IM) systems, which encrypt messages end-to-end using variations of the Double Ratchet protocol [DoubleRatchet].

End-to-end encrypted instant messaging was also a motivator for the Common Protocol for Instant Messaging (CPIM) [RFC3862], however the model used at the time assumed standalone encryption of each message using a protocol such as S/MIME [RFC8551] or PGP [RFC3156] to interoperate between IM protocols such as SIP [RFC3261] and XMPP [RFC6120]. For a variety of practical reasons, interoperable end-to-end encryption between IM systems was never deployed commercially.

There are now several instant messaging vendors implementing MLS, and the MIMI (More Instant Messaging Interoperability) Working Group is chartered to standardize an extensible interoperable messaging format for common features to be conveyed "inside" MLS application messages.

This document assumes that MLS clients advertise media types they support and can determine what media types are required to join a specific MLS group using the content advertisement extensions in Section 2.3 of [I-D.ietf-mls-extensions]. It allows implementations to define MLS groups with different media type requirements and allows MLS clients to send extended or proprietary messages that would be interpreted by some members of the group while assuring that an interoperable end-to-end encrypted baseline is available to all members, even when the group spans multiple systems or vendors.

Below is a list of some features commonly found in IM group chat systems:

* plain text and rich text messaging
* mentions
* replies
* reactions
* edit or delete previously sent messages
* expiring messages
* delivery notifications
* read receipts
* shared files/audio/videos
* calling / conferencing
* message threading

3. Overview

3.1. Binary encoding

The MIMI Content format is encoded in Concise Binary Object Representation (CBOR) [RFC8949]. The Working Group chose a binary format in part because:

* we do not want to scan body parts to check for boundary marker collisions. This rules out using multipart MIME types.
* we do not want to base64 encode body parts with binary media types (ex: images). This rules out using JSON to carry the binary data.

All examples start with an instance document annotated in the CBOR Extended Diagnostic Notation (described in [Appendix G of @!RFC8610] and more rigorously specified in [I-D.ietf-cbor-edn-literals]), and then include a hex dump of the CBOR data in the pretty printed format popularized by the CBOR playground website (https://cbor.me (https://cbor.me)) with some minor whitespace and comment reformatting. Finally a message ID for the message is included for most messages.

All the instance documents validate using the CDDL schemas in Appendix B and are included in the examples directory in the github repo for this document.

3.2. Naming schemes

IM systems have a number of types of identifiers. These are described in detail in [I-D.mahy-mimi-identity]. A few of these used in this document are:

* handle identifier (external, friendly representation). This is the type of identifier described later as the senderUserUrl in the examples, which is analogous to the From header in email.
* client/device identifier (internal representation). This is the type of identifier described as the senderClientUrl in the examples.
* group or room or conversation or channel name (either internal or external representation). This is the type of identifier described as the MLS group URL in the examples.

This proposal relies on URIs for naming and identifiers. All the example use the im: URI scheme (defined in [RFC3862]), but any instant messaging scheme could be used.

3.3. Message ID and Accepted Timestamp

Every MIMI content message has a message ID which is calculated from the hash of the ciphertext of the message. When the content is end-to-encrypted with MLS for a specific MLS group, the cipher suite for the group specifies a hash algorithm. The message ID is the first 32 octets of the hash of the MLSMessage struct using that hash algorithm.

As described in the the MIMI architecture [I-D.ietf-mimi-arch], one provider, called the hub, is responsible for ordering messages. The hub is also responsible for recording the time that any application message is accepted, and conveying it to any "follower" providers which receive messages from the group. It is represented as the whole number of milliseconds since the start of the UNIX epoch (01-Jan-1970 00:00:00 UTC). To the extent that the accepted timestamp is available to a MIMI client, the client can use it for fine grain sorting of messages into a consistent order.

3.4. Message Container

Most common instant messaging features are expressed as individual messages. A plain or rich text message is obviously a message, but a reaction (ex: like), a reply, editing a previous message, deleting an earlier message, and read receipts are all typically modeled as another message with different properties.

This document describes the semantics of a message container, which can represent most of these previously mentioned message types. The container typically carries one or more body parts with the actual message content (for example, an emoji used in a reaction, a plain text or rich text message or reply, a link, or an inline image).

3.5. Message Status Report

This document also describes the semantics of a status report of other messages. Because some messaging systems deliver messages in batches and allow a user to mark several messages read at a time, the report format allows a single report to convey the read/delivered status of multiple messages (by message ID) within the same MLS group at a time.

4. MIMI Content Container Message Semantics

Each MIMI Content message is a container format with two categories of information:

* the message behavior fields (which can have default or empty values), and
* the body part(s) and associated parameters

The object fields in the structure defined below are numbered in curly braces for reference in the text.

The subsections that follow contain snippets of Concise Data Definition Language (CDDL) [RFC8610] schemas for the MIMI Content Container. The complete collected CDDL schema for MIMI Content Container is available in Appendix A.1.

4.1. Message Behavior Fields

```
mimiContent = [
  replaces: null / MessageId,       ; {1}
  topicId: bstr,                    ; {2}
  expires: uint .size 4,            ; {3}
  inReplyTo: null / InReplyTo,      ; {4}
  lastSeen: [* MessageId],          ; {5}
  extensions: {* name => value },   ; {6}
  nestedPart: NestedPart            ; {7}
]

MessageId = bstr .size 32          ; MessageId is derived from SHA256 hash
```

The replaces {1} data field indicates that the current message is a replacement or update to a previous message whose message ID is in the replaces data field. It is used to edit previously-sent messages, delete previously-sent messages, and adjust reactions to messages to which the client previously reacted. If the replaces field is absent, the receiver assumes that the current message has not identified any special relationship with another previous message.

The topicId {2} data field indicates that the current message is part of a logical grouping of messages which all share the same value in the topicId data field. If the topicId is zero length, there is no such grouping.

The expires {3} data field is a hint from the sender to the receiver that the message should be locally deleted and disregarded at a specific timestamp in the future. Indicate a message with no specific expiration time with the value zero. The data field is an unsigned integer number of seconds after the start of the UNIX epoch. Using an 32-bit unsigned integer allows expiration dates until the year 2106. Note that specifying an expiration time provides no assurance that the client actually honors or can honor the expiration time, nor that the end user didn't otherwise save the expiring message (ex: via a screenshot).

The inReplyTo {4} data field indicates that the current message is a related continuation of another message sent in the same MLS group. If present, it contains the message ID of the referenced message and the SHA-256 hash [RFC6234] of its MimiContent structure. Otherwise, the receiver assumes that the current message has not identified any special relationship with another previous message.

The inReplyTo hash is a message digest from the IANA Named Information Hash Algorithm Registry (https://www.iana.org/assignments/named-information/named-information.xhtml#hash-alg), used to make sure that a MIMI message cannot refer to a sequence of referred messages which refers back to itself. When replying, a client MUST NOT knowingly create a sequence of replies which create a loop.

When receiving a message, the client verifies that the hash is correct. Next it checks if the referenced message is itself a Reply. If so, it continues following the referenced messages, checking that neither the messageId nor the hash of any of referenced messages indicates a Reply which "loops" back to a message later in the inReplyTo chain.

```
InReplyTo = [
  message: MessageId,
  hashAlg: uint .size 8,   ; a value from the IANA Named Information Hash
                           ; Algorithm Registry. Default: SHA-256 = 1
  hash: bstr
]
```

Note that a inReplyTo always references a specific message ID. Even if the original message was edited several times, a reply always refers to a specific version of that message, and SHOULD refer to the most current version at the time the reply is sent.

4.2. Message Ordering

The lastSeen {5} data field indicates the latest message the sender was aware of in the group. It is a list of message ids.

If the sender recently joined the group and has not yet seen any messages, the list is empty.

If the sender identifies a single message as unambiguously the latest message in the group, the lastSeen list contains a single message id from that message.

Imagine however that two users (Bob and Cathy) see a message from Alice offering free Hawaiian pizza, and reply at the same time. Bob and Cathy both send messages with their lastSeen including a single message id (Alice's) message about pizza. Their messages don't need to be replies or reactions. Bob might just send a message saying he doesn't like pineapple on pizza. Now Doug receives all these messages and replies as well. Doug's message contains a lastSeen including the message id list of both Bob's and Cathy's replies, effectively "merging" the order of messages.

The next message after Doug's message contains a lastSeen containing only the message id of Doug's message.

4.3. Extension Fields

In order to add additional functionality to MIMI, senders can include extension fields in the message format {6}. Each extension has a name, which contains between 1 and 255 octets of UTF-8, and an opaque value. The value of each extension can be between 0 and 65535 octets. The message content extensions field MUST NOT include more than one extension field with the same name.

```
name = tstr .size (1..255)
value = bstr .size (0..4095)
```

4.4. Message Bodies

Every MIMI content message has a body {7} which can have multiple, possibly nested parts. A body with zero parts is permitted when deleting or unliking. External body parts Section 4.5 are also supported. When there is a single (inline) part or a (single) externally reference part, its IANA media type, subtype, and parameters are included in the contentType field {8}.

```
NestedPart = [
  disposition: baseDispos / $extDispos / unknownDispos,  ; {10}
  language: tstr,                                        ; {11}
  partIndex: uint .size 2,                               ; {12}
  ( NullPart // SinglePart // ExternalPart // MultiPart)
]

NullPart = ( cardinality: nullpart )

SinglePart = (
    cardinality: single,
    contentType: tstr,        ; {8}
    content: bstr
)

ExternalPart = (
    cardinality: external,
    contentType: tstr,
    url: uri,
    expires: uint .size 4,
    size: uint .size 8,
    encAlg: uint .size 2,
    key: bstr,
    nonce: bstr,
    aad: bstr,
    hashAlg: uint .size 1,
    contentHash: bstr,
    description: tstr
)

MultiPart = (
    cardinality: multi,
    partSemantics: chooseOne / singleUnit / processAll,
    parts: [2* NestedPart]
)

; cardinality
nullpart = 0
single   = 1
external = 2
multi    = 3

; part semantics {9}
chooseOne  = 0  ; receiver picks exactly one part to process
singleUnit = 1  ; receiver processes all parts as single unit
processAll = 2  ; receiver processes all parts individually
```

With some types of message content, there are multiple media types associated with the same message which need to be rendered together, for example a rich-text message with an inline image. With other messages, there are multiple choices available for the same content, for example a choice among multiple languages, or between two different image formats. The relationship semantics among the parts is specified as an enumeration {9}.

The chooseOne part semantic is roughly analogous to the semantics of the multipart/alternative media type, except that the ordering of the nested body parts is merely a preference of the sender. The receiver can choose the body part among those provided according to its own policy.

The singleUnit part semantic is roughly analogous to the semantics of the multipart/related media type, in that all the nested body parts at this level are part of a single entity (for example, a rich text message with an inline image). If the receiver does not understand even one of the nested parts at this level, the receiver should not process any of them.

The processAll part semantic is roughly analogous to the semantics of the multipart/mixed media type. The receiver should process as many of the nested parts at this level as possible. For example, a rich text document with a link, and a preview image of the link target could be expressed using this semantic. Processing the preview image is not strictly necessary for the correct rendering of the rich text part.

The disposition {10} and language {11} of each part can be specified for any part, including for nested parts. The disposition represents the intended semantics of the body part or a set of nested parts. It is inspired by the values in the Content-Disposition MIME header [RFC2183].

```
baseDispos = &(
    unspecified: 0,
    render: 1,
    reaction: 2,
    profile: 3,
    inline: 4,
    icon: 5,
    attachment: 6,
    session: 7,
    preview: 8
)
unknownDispos = &( unknown: 9..255 ) ; Note: any ext_dispos take precedence
```

The render disposition means that the content should be rendered according to local policy. The inline dispositions means that the content should be rendered "inline" directly in the chat interface. The attachment disposition means that the content is intended to be downloaded by the receiver instead of being rendered immediately. The reaction disposition means that the content is a single reaction to another message, typically an emoji, but which could be an image, sound, or video. The reaction disposition was originally published in [RFC9078], but was incorrectly placed in the Content Disposition Parameters IANA registry instead of in the Content Disposition Values registry. The session disposition means that the content is a description of a multimedia session, or a URI used to join one. The preview disposition means that the content is a sender-generated preview of something, such as the contents of a link.

The value of the language data field is an empty string or a comma-separated list of one or more Language-tags as defined in [RFC5646].

Each part also has an part index {12}, which is a zero-indexed, depth-first integer. It is used to efficiently refer to a specific body part (for example, an inline image) within another part. See {Nested body examples} for an example of how the part index is calculated.

4.5. External content

It is common in Instant Messaging systems to reference external content via URI that will be processed automatically, either to store bulky content (ex: videos, images, recorded sounds) outside the messaging infrastructure, or to access a specific service URI, for example, a media forwarding service for conferencing.

An ExternalPart is a convenient way to reference this content. It provides a similar function to the message/external-body media type. It optionally includes the size of the data in octets (or zero if the length is not provided). It also includes an optional timestamp after which the external content is invalid, expressed as seconds since the start of the UNIX epoch (01-Jan-1970), or zero if the content does not expire.

```
ExternalPart = (
    cardinality: external,
    contentType: tstr,     ; An IANA media type {8}
    url: uri,              ; A URL where the content can be fetched
    expires: uint .size 4, ; expiration in seconds since UNIX epoch
    size: uint .size 8,    ; size of content in octets
    encAlg: uint .size 2,  ; An IANA AEAD Algorithm number, or zero
    key: bstr,             ; AEAD key
    nonce: bstr,           ; AEAD nonce
    aad: bstr,             ; AEAD additional authentiation data
    hashAlg: uint .size 1, ; An IANA Named Information Hash Algorithm
    contentHash: bstr,     ; hash of the content at the target url
    description: tstr      ; an optional text description
)
```

Typically, external content is encrypted with an ephemeral symmetric key before it is uploaded, and whatever is necessary for decryption is shared over the message channel.

It is a matter of local policy to where the content is uploaded. Often in federated messaging systems, the sender of the content stores the external content in their own domain, but in some systems the content is stored in the "owning" or "hub" domain of the MLS group.

Before being uploaded, private external content is encrypted with an IANA-registered Authenticated Encryption with Additional Data (AEAD) algorithm as described in [RFC5116]. The key, nonce, and additional authenticated data (aad) values are set to the values used during the encryption. Unless modified by an extension, the default value of the aad is empty.

If the external URL is a service, or the external content is not considered private, the encAlg is set to zero, and the key, nonce, and aad fields are zero length.

Implementations of this specification MUST implement the AES-128-GCM algorithm.

4.6. Derived Data Values

In addition to fields which are contained in a MIMI content message, there are also two fields which the implementation can definitely derive (the MLS group ID {13}, and the leaf index of the sender {14}). Many implementations could also determine one or more of: the sender’s client identifier URL {15}, the user identifier URL of the credential associated with the sender {16}, and the identifier URL for the MIMI room {17}.

```
MessageDerivedValues = [
    messageId: MessageId,              ; sha256 hash of message ciphertext
    hubAcceptedTimestamp: Timestamp,
    mlsGroupId: bstr,                  ; value always available {13}
    senderLeafIndex: uint .size 4,     ; value always available {14}
    senderClientUrl: uri               ; {15},
    senderUserUrl: uri,                ; "From" {16}
    roomUrl: uri                       ; "To"   {17}
]

MessageId = bstr .size 32
Timestamp = #6.62(uint .size 8)    ; milliseconds since start of UNIX epoch
```

5. Examples

In the following examples, we assume that an MLS group is already established and that either out-of-band or using the MLS protocol or MLS extensions, or their client to provider protocol that the following is known to every member of the group:

* The membership of the group (via MLS).
* The identity of any MLS client which sends an application message (via MLS).
* The MLS group ID (via MLS)
* The human readable name(s) of the MIMI room, if any (out-of-band or extension).
* Which media types are mandatory to implement (MLS content advertisement extensions).
* For each member, the media types each supports (MLS content advertisement extensions).

Messages sent to an MLS group are delivered to every member of the group active during the epoch in which the message was sent.

All the examples start with a CBOR instance document annotated in the Extended Diagnostic Format (described in [Appendix G of @!RFC8610] and more rigorously specified in [I-D.ietf-cbor-edn-literals]), and then include a hex dump of the CBOR data in the pretty printed format popularized by the CBOR playground website (https://cbor.me (https://cbor.me)) with some minor whitespace and comment reformatting. Finally a message ID for the message is included for most messages.

All the instance documents validate using the CDDL schemas in Appendix B and are included in the examples directory in the github repo for this document.

5.1. Original Message

In this example, Alice Smith sends a rich-text (Markdown) [RFC7763] message to the Engineering Team MLS group. The following values are derived from the client, except for the hub received timestamp, which might be available for the client from its provider:

* Sender leaf index: 4
* Sender client ID URL: im:3b52249d-68f9-45ce-8bf5-c799f3cad7ec/0003@example.com
* Sender user handle URL: im:%40alice-smith@example.com
* MLS group ID: 7u4NEqe1tbeBFa0aHdsTgRyD/XOHxD5meZpZS+7aJr8=
* The MIMI room URL: im:#engineering_team@example.com
* The MIMI room name: "Engineering Team"
* Message ID: 0xd3c14744d1791d02548232c23d35efa9 7668174ba385af066011e43bd7e51501
* Timestamp: 1644387225019 = 2022-02-08T22:13:45.019-00:00

Below is the message in annotated Extended Diagnostic Notation, and pretty printed CBOR.

```
[
  null,                              / replaces          /
  h'',                               / topicId           /
  0,                                 / expires           /
  null,                              / inReplyTo         /
  [],                                / lastSeen          /
  {},                                / extensions        /
  [                                  / body (NestedPart) /
    1,                                 / dispostion = render       /
    "",                                / language                  /
    0,                                 / partIndex = 1st part      /
    1,                                 / cardinality = single part /
    "text/markdown;variant=GFM",       / contentType               /
                                       / content                   /
    'Hi everyone, we just shipped release 2.0. __Good  work__!'
  ]
]
```

```
87                                      # array(7)
   f6                                   # primitive(22)
   40                                   # bytes(0)
                                        # ""
   00                                   # unsigned(0)
   f6                                   # primitive(22)
   80                                   # array(0)
   a0                                   # map(0)
   86                                   # array(6)
      01                                # unsigned(1)
      60                                # text(0)
                                        # ""
      00                                # unsigned(0)
      01                                # unsigned(1)
      78 1b                             # text(27)
         746578742f6d61726b646f776e3b636861727365743d7574662d38
         # "text/markdown;charset=utf-8"
      58 38                             # bytes(56)
         486920657665727796f6e652c207765206a757374207368697070656420
         72656c6561736520322e302e205f5f476f6f6420776f726b5f5f21
         # "Hi everyone, we just shipped release 2.0. __Good work__!"
```

Below are the rest of the implied values for this message:

```
[
  / messageId  (Original message) /
  h'd3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501',

  / hubAcceptedTimestamp = 2022-02-08T22:13:45.019-00:00 /
  62(1644387225019),

  / mlsGroupId /
  h'eeee0d12a7b5b5b78115ad1a1ddb13811c83fd7387c43e66799a594beeda26bf',

  / senderLeadIndex /
  4,

  / senderClientUrl /
  32("mimi://example.com/d/3b52249d-68f9-45ce-8bf5-c799f3cad7ec/0003"),

  / senderUserUrl /
  32("mimi://example.com/u/alice-smith"),

  / roomUrl /
  32("mimi://example.com/r/engineering_team")
]
```

5.2. Reply

A reply message looks similar, but contains the message ID of the original message in the inReplyTo data field. The derived MLS group ID, URL, and name do not change in this example. The derived senderClientId and senderLeafIndex are not especially relevant so all but the user handle URL, message ID, and hub received timestamp will be omitted.

* Sender user handle URL: im:%40bob-jones@example.com
* MessageId: 0xe701beee59f9376282f39092e1041b2a c2e3aad1776570c1a28de244979c71ed
* Timestamp = 1644387237492 = 2022-02-08T22:13:57.492-00:00

Below is the annotated message in EDN and pretty printed CBOR:

```
[
  null,                                  / replaces             /
  h'',                                   / topicId              /
  0,                                     / expires = never      /
  [                                      / InReplyTo            /
     /  message = Original message                               /
     h'd3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501',
     1,                                    /  hashAlg = sha256  /
     /  hash  /
     h'6b44053cb68e3f0cdd219da8d7104afc2ae5ffff782154524cef093de39345a5'
  ],
  [                                        / lastSeen (1 item)   /
     /  Original message                                         /
     h'd3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501'
  ],
  {},                                    / extensions             /
  [                                      / body (NestedPart)      /
    1,                                     / dispostion = render  /
    "",                                    / language             /
    0,                                     / partIndex = 1st part /
    1,                                     / cardinality = single /
    "text/markdown;variant=GFM",           / contentType          /
                                           / content              /
    'Right on! _Congratulations_ \'all!'
  ]
]
```

```
87                                      # array(7)
   f6                                   # primitive(22)
   40                                   # bytes(0)
                                        # ""
   00                                   # unsigned(0)
   83                                   # array(3)
      58 20                             # bytes(32)
         d3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501
      01                                # unsigned(1)
      58 20                             # bytes(32)
         6b44053cb68e3f0cdd219da8d7104afc2ae5ffff782154524cef093de39345a5
   81                                   # array(1)
      58 20                             # bytes(32)
         d3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501
   a0                                   # map(0)
   86                                   # array(6)
      01                                # unsigned(1)
      60                                # text(0)
                                        # ""
      00                                # unsigned(0)
      01                                # unsigned(1)
      78 1b                             # text(27)
         746578742f6d61726b646f776e3b636861727365743d7574662d38
         # "text/markdown;charset=utf-8"
      58 21                             # bytes(33)
         5269676874206f6e21205f436f6e67726174756c6174696f6e735f2027616c6c21
         # "Right on! _Congratulations_ 'all!"
```

5.3. Reaction

A reaction looks like a reply, but uses the Disposition token of reaction. It is modeled on the reaction Content-Disposition token defined in [RFC9078]. Both indicate that the intended disposition of the contents of the message is a reaction.

The content in the sample message is a single Unicode heart character (U+2665) which is expressed in UTF-8 as 0xe299a5. Discovering the range of characters each implementation could render as a reaction can occur out-of-band and is not within the scope of this proposal. However, an implementation which receives a reaction character string it does not recognize could render the reaction as a reply, possibly prefixing with a localized string such as "Reaction: ". Note that a reaction could theoretically even be another media type (ex: image, audio, or video), although not currently implemented in major instant messaging systems. Note that many systems allow mutiple independent reactions per sender.

* Sender user handle URL: im:cathy-washington@example.com

* Message ID: 0x4dcab7711a77ea1dd025a6a1a7fe01ab
  3b0d690f82417663cb752dfcc37779a1
* Timestamp: 1644387237728 = 2022-02-08T22:13:57.728-00:00

Below is the annotated message in EDN and pretty printed CBOR:

```
[
  null,                                / replaces             /
  h'',                                 / topicId              /
  0,                                   / expires = never      /
  [                                    / InReplyTo            /
     /  message = Original message                            /
     h'd3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501',
     1,                                  /  hashAlg = sha256  /
     /  hash  /
     h'6b44053cb68e3f0cdd219da8d7104afc2ae5ffff782154524cef093de39345a5'
  ],
  [                                      / lastSeen (1 item)   /
     /  Reply message                                         /
     h'e701beee59f9376282f39092e1041b2ac2e3aad1776570c1a28de244979c71ed'
  ],
  {},                                  / extensions             /
  [                                    / body (NestedPart)      /
    2,                                   / dispostion = reaction /
    "",                                  / language             /
    0,                                   / partIndex = 1st part  /
    1,                                   / cardinality = single  /
    "text/plain;charset=utf-8",          / contentType          /
    ''                                  / content = U+2665 (heart) /
  ]
]
```

```
87                                      # array(7)
   f6                                   # primitive(22)
   40                                   # bytes(0)
                                        # ""
   00                                   # unsigned(0)
   83                                   # array(3)
      58 20                             # bytes(32)
         d3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501
      01                                # unsigned(1)
      58 20                             # bytes(32)
         6b44053cb68e3f0cdd219da8d7104afc2ae5ffff782154524cef093de39345a5
   81                                   # array(1)
      58 20                             # bytes(32)
         e701beee59f9376282f39092e1041b2ac2e3aad1776570c1a28de244979c71ed
   a0                                   # map(0)
   86                                   # array(6)
      02                                # unsigned(2)
      60                                # text(0)
                                        # ""
      00                                # unsigned(0)
      01                                # unsigned(1)
      78 18                             # text(24)
         746578742f706c61696e3b636861727365743d7574662d38
         # "text/plain;charset=utf-8"
      43                                # bytes(3)
         e299a5                         # ""
```

5.4. Mentions

In instant messaging systems and social media, a mention allows special formatting and behavior when a name, handle, or tag associated with a known group is encountered, often when prefixed with a commercial-at "@" character for mentions of users or a hash "#" character for groups or tags. A message which contains a mention may trigger distinct notifications on the IM client.

We can convey a mention by linking the user handle URI, or group URI in Markdown or HTML rich content. For example, a mention using Markdown is indicated below.

* Sender user handle URL: im:cathy-washington@example.com
* Message ID: 0x6b50bfdd71edc83554ae21380080f4a3 ba77985da34528a515fac3c38e4998b8
* Timestamp: 1644387243008 = 2022-02-08T22:14:03.008-00:00

Below is the annotated message in EDN and pretty printed CBOR:

```
   [
     null,                                 / replaces             /
     h'',                                  / topicId              /
     0,                                    / expires = never      /
     null,                                 / InReplyTo            /
     [                                       / lastSeen (1 item)   /
        /  Reply message                                           /
        h'e701beee59f9376282f39092e1041b2ac2e3aad1776570c1a28de244979c71ed'
     ],
     {},                                   / extensions             /
     [                                     / body (NestedPart)      /
       1,                                    / dispostion = render  /
       "",                                   / language             /
       0,                                    / partIndex = 1st part /
       1,                                    / cardinality = single /
       "text/markdown;variant=GFM",          / contentType          /
                                             / content              /
       'Kudos to [@Alice Smith](im:alice-smith@example.com)'
       ' for making the release happen!'
     ]
   ]

87                                     # array(7)
   f6                                  # primitive(22)
   40                                  # bytes(0)
                                       # ""
   00                                  # unsigned(0)
   f6                                  # primitive(22)
   81                                  # array(1)
      58 20                            # bytes(32)
         e701beee59f9376282f39092e1041b2ac2e3aad1776570c1a28de244979c71ed
   a0                                  # map(0)
   86                                  # array(6)
      01                               # unsigned(1)
      60                               # text(0)
                                       # ""
      00                               # unsigned(0)
      01                               # unsigned(1)
      78 1b                            # text(27)
         746578742f6d61726b646f776e3b636861727365743d7574662d38
         # "text/markdown;charset=utf-8"
      58 52                            # bytes(82)
         4b75646f7320746f205b40416c69636520536d6974685d28696d3a
         616c6963652d736d697468406578616d706c652e636f6d2920666f
         72206d616b696e6720746865207265c656173652068617070656e21
         # "Kudos to [@Alice Smith](im:alice-smith@example.com)
         # for making the release happen!"
```

The same mention using HTML [W3C.CR-html52-20170808] would instead replace in the EDN the contentType and content indicated below.

```
  / ... /
  "text/html;charset=utf-8",
  '<p>Kudos to <a href="im:alice-smith@example.com">@Alice Smith</a> for makin
g the release happen!</p>'
```

5.5. Edit

Unlike with email messages, it is common in IM systems to allow the sender of a message to edit or delete the message after the fact. Typically the message is replaced in the user interface of the receivers (even after the original message is read) but shows a visual indication that it has been edited.

The replaces data field includes the message ID of the message to edit/replace. The message included in the body is a replacement for the message with the replaced message ID.

Here Bob Jones corrects a typo in his original message:

* Sender user handle URL: im:%40bob-jones@example.com
* Message ID:0x89d3472622a4d9de526742bcd00b09dc 78fa4edceaf2720e17b730c6dfba8be4
* Timestamp: 1644387248621 = 2022-02-08T22:14:08.621-00:00

```
[
  / replaces = Reply message                                    /
  h'e701beee59f9376282f39092e1041b2ac2e3aad1776570c1a28de244979c71ed',
  h'',                                  / topicId               /
  0,                                    / expires = never       /
  [                                       / InReplyTo           /
     /  message = Original message                              /
     h'd3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501',
     1,                                   /  hashAlg = sha256   /
     /  hash                 /
     h'6b44053cb68e3f0cdd219da8d7104afc2ae5ffff782154524cef093de39345a5'
  ],
  [                                       / lastSeen (2 items)  /
    /  Reaction message /
    h'4dcab7711a77ea1dd025a6a1a7fe01ab
      3b0d690f82417663cb752dfcc37779a1',
    /  Mention message  /
    h'6b50bfdd71edc83554ae21380080f4a3
      ba77985da34528a515fac3c38e4998b8'
  ],
  {},                                   / extensions            /
  [                                     / body (NestedPart)     /
    1,                                    / dispostion = render /
    "",                                   / language            /
    0,                                    / partIndex = 1st part /
    1,                                    / cardinality = single /
    "text/markdown;variant=GFM",          / contentType         /
                                          / content             /
    'Right on! _Congratulations_ y\'all'
  ]
]
```

```
87                                      # array(7)
   58 20                                # bytes(32)
      e701beee59f9376282f39092e1041b2ac2e3aad1776570c1a28de244979c71ed
   40                                   # bytes(0)
                                        # ""
   00                                   # unsigned(0)
   83                                   # array(3)
      58 20                             # bytes(32)
         d3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501
      01                                # unsigned(1)
      58 20                             # bytes(32)
         6b44053cb68e3f0cdd219da8d7104afc2ae5ffff782154524cef093de39345a5
   82                                   # array(2)
      58 20                             # bytes(32)
         4dcab7711a77ea1dd025a6a1a7fe01ab3b0d690f82417663cb752dfcc37779a1
      58 20                             # bytes(32)
         6b50bfdd71edc83554ae21380080f4a3ba77985da34528a515fac3c38e4998b8
   a0                                   # map(0)
   86                                   # array(6)
      01                                # unsigned(1)
      60                                # text(0)
                                        # ""
      00                                # unsigned(0)
      01                                # unsigned(1)
      78 1b                             # text(27)
         746578742f6d61726b646f776e3b636861727365743d7574662d38
         # "text/markdown;charset=utf-8"
      58 22                             # bytes(34)
         5269676874206f6e21205f436f6e67726174756c6174696f6e735f
         207927616c6c21
         # "Right on! _Congratulations_ y'all!"
```

Note that replies and reactions always refer to a specific message id, and therefore a specific "version" of a message, which could have been edited before and/or after the message id referenced in the reply or reaction. It is a matter of local policy how to render (if at all) a reaction to a subsequently edited message.

5.6. Delete

In IM systems, a delete means that the author of a specific message has retracted the message, regardless if other users have read the message or not. Typically a placeholder remains in the user interface showing that a message was deleted. Replies which reference a deleted message typically hide the quoted portion and reflect that the original message was deleted.

If Bob deleted his message instead of modifying it, we would represent it using the replaces data field, and using an empty body (NullPart), as shown below.

* Sender user handle URL: im:%40bob-jones@example.com
* Message ID: 0x89d3472622a40d6ceeb27c42490fdc64 c0e9c20c598f9d7c8e81640dae8db0fb
* Timestamp: 1644387248621 = 2022-02-08T22:14:08.621-00:00

```
[
  / replaces = Reply message                                  /
  h'e701beee59f9376282f39092e1041b2ac2e3aad1776570c1a28de244979c71ed',
  h'',                                  / topicId             /
  0,                                    / expires = never     /
  [                                       / InReplyTo         /
     /  message = Original message                            /
     h'd3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501',
     1,                                   /  hashAlg = sha256  /
     /  hash                 /
     h'6b44053cb68e3f0cdd219da8d7104afc2ae5ffff782154524cef093de39345a5'
  ],
  [                                       / lastSeen (1 items)   /
    /  Edit message /
    h'89d3472622a4d9de526742bcd00b09dc78fa4edceaf2720e17b730c6dfba8be4'
  ],
  {},                                   / extensions            /
  [                                     / body (NestedPart)     /
    1,                                    / dispostion = render  /
    "",                                   / language            /
    0,                                    / partIndex = 1st part /
    0                                     / cardinality = zero parts /
  ]
]
```

```
87                                      # array(7)
   58 20                                # bytes(32)
      4dcab7711a77ea1dd025a6a1a7fe01ab3b0d690f82417663cb752dfcc37779a1
   40                                   # bytes(0)
                                        # ""
   00                                   # unsigned(0)
   83                                   # array(3)
      58 20                             # bytes(32)
         d3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501
      01                                # unsigned(1)
      58 20                             # bytes(32)
         6b44053cb68e3f0cdd219da8d7104afc2ae5ffff782154524cef093de39345a5
   81                                   # array(1)
      58 20                             # bytes(32)
         89d3472622a40d6ceeb27c42490fdc64c0e9c20c598f9d7c8e81640dae8db0fb
   a0                                   # map(0)
   84                                   # array(4)
      02                                # unsigned(2)
      60                                # text(0)
                                        # ""
      00                                # unsigned(0)
      00                                # unsigned(0)
```

5.7. Unlike

In most IM systems, not only is it possible to react to a message ("Like"), but it is possible to remove a previous reaction ("Unlike"). This can be accomplished by deleting the message which creates the original reaction

If Cathy removes her reaction, we would represent the removal using a replaces data field with an empty body, referring to the message which created the reaction, as shown below.

* Sender user handle URL: im:cathy-washington@example.com
* Message ID: 0x1a771ca1d84f8fda4184a1e02a549e20 1bf434c6bfcf1237fa45463c6861853b
* Timestamp: 1644387250389 = 2022-02-08T22:14:10.389-00:00

```
   [
     / replaces = Reaction message                                  /
     h'4dcab7711a77ea1dd025a6a1a7fe01ab3b0d690f82417663cb752dfcc37779a1',
     h'',                                  / topicId                /
     0,                                    / expires = never        /
     [                                       / InReplyTo            /
        /  message = Original message                                /
        h'd3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501',
        1,                                    /  hashAlg = sha256  /
        /  hash                 /
        h'6b44053cb68e3f0cdd219da8d7104afc2ae5ffff782154524cef093de39345a5'
     ],
     [                                       / lastSeen (1 items)   /
       /  Delete message /
       h'89d3472622a40d6ceeb27c42490fdc64c0e9c20c598f9d7c8e81640dae8db0fb'
     ],
     {},                                   / extensions             /
     [                                     / body (NestedPart)      /
       2,                                    / dispostion = reaction /
       "",                                   / language             /
       0,                                    / partIndex = 1st part  /
       0                                     / cardinality = zero parts /
     ]
   ]

87                                     # array(7)
   58 20                               # bytes(32)
      e701beee59f9376282f39092e1041b2ac2e3aad1776570c1a28de244979c71ed
   40                                  # bytes(0)
                                       # ""
   00                                  # unsigned(0)
   83                                  # array(3)
      58 20                            # bytes(32)
         d3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501
      01                               # unsigned(1)
      58 20                            # bytes(32)
         6b44053cb68e3f0cdd219da8d7104afc2ae5ffff782154524cef093de39345a5
   81                                  # array(1)
      58 20                            # bytes(32)
         89d3472622a4d9de526742bcd00b09dc78fa4edceaf2720e17b730c6dfba8be4
   a0                                  # map(0)
   84                                  # array(4)
      01                               # unsigned(1)
      60                               # text(0)
                                       # ""
      00                               # unsigned(0)
      00                               # unsigned(0)
```

5.8. Expiring

Expiring messages are designed to be deleted automatically by the receiving client at a certain time whether they have been read or not. As with manually deleted messages, there is no guarantee that an uncooperative client or a determined user will not save the content of the message, however most clients respect the convention.

The expires data field contains the timestamp when the message can be deleted. The semantics of the header are that the message is automatically deleted by the receiving clients at the indicated time without user interaction or network connectivity necessary.

* Sender user handle URL: im:alice-smith@example.com
* Message ID: 0x5c95a4dfddab84348bcc265a479299fb d3a2eecfa3d490985da5113e5480c7f1
* Timestamp: 1644389403227 = 2022-02-08T22:49:06.227-00:00

```
[
  null,                              / replaces                /
  h'',                               / topicId                 /
  1644390004,                        / expires = in 10 minutes /
  null,                              / inReplyTo               /
  [                                  / lastSeen (1 item)       /
     /  Unlike message                                          /
     h'1a771ca1d84f8fda4184a1e02a549e201bf434c6bfcf1237fa45463c6861853b'
  ],
  {},                                / extensions        /
  [                                  / body (NestedPart) /
    1,                                 / dispostion = render       /
    "",                                / language                  /
    0,                                 / partIndex = 1st part      /
    1,                                 / cardinality = single part /
    "text/markdown;variant=GFM",       / contentType               /
                                       / content                   /
    '__*VPN GOING DOWN*__ I\'m rebootinging the VPN in ten minutes'
    ' unless anyone objects.'
  ]
]
```

```
87                                      # array(7)
   f6                                   # primitive(22)
   40                                   # bytes(0)
                                        # ""
   1a 62036674                          # unsigned(1644390004)
   f6                                   # primitive(22)
   81                                   # array(1)
      58 20                             # bytes(32)
         1a771ca1d84f8fda4184a1e02a549e201bf434c6bfcf1237fa45463c6861853b
   a0                                   # map(0)
   86                                   # array(6)
      01                                # unsigned(1)
      60                                # text(0)
                                        # ""
      00                                # unsigned(0)
      01                                # unsigned(1)
      78 1b                             # text(27)
         746578742f6d61726b646f776e3b636861727365743d7574662d38
         # "text/markdown;charset=utf-8"
      58 50                             # bytes(80)
         5f5f2a56504e20474f494e4720444f574e2a5f5f0a49276d207265
         626f6f74696e67207468652056504e20696e2074656e206d696e75
         74657320756e6c65737320616e796f6e65206f626a656374732e
         # "__*VPN GOING DOWN*__\nI'm rebooting the VPN in ten
         #  minutes unless anyone objects."
```

5.9. Attachments

An ExternalPart is a convenient way to present both "attachments" and (possibly inline rendered) content which is too large to be included in an MLS application message. The disposition data field is set to inline if the sender recommends inline rendering, or attachment if the sender intends the content to be downloaded or rendered separately.

```
[
  null,                              / replaces                /
  h'',                               / topicId                 /
  0,                                 / expires = never         /
  null,                              / inReplyTo               /
  [                                  / lastSeen (1 item)       /
     /  Expiring message                                       /
     h'5c95a4dfddab84348bcc265a479299fbd3a2eecfa3d490985da5113e5480c7f1'
  ],
  {},                                / extensions        /
  [                                  / body (NestedPart) /
    6,                                 / dispostion = attachment   /
    "en",                              / language = en             /
    0,                                 / partIndex = 1st part      /
    2,                                 / cardinality = external part /
    "video/mp4",                       / contentType               /
    32("https://example.com/storage/bigfile.mp4"), / url           /
    0,                                 / expires                   /
    708234961,                         / size                      /
    1,                                 / encAlg = AES-128-GCM      /
    h'21399320958a6f4c745dde670d95e0d8',   / key                   /
    h'c86cf2c33f21527d1dd76f5b',       / nonce                     /
    h'',                               / aad                       /
    1,                                 / hashAlg = sha256          /
    /  content hash                                                /
    h'9ab17a8cf0890baaae7ee016c7312fcc080ba46498389458ee44f0276e783163',
    "2 hours of key signing video"     / description               /
  ]
]
```

```
87                                      # array(7)
   f6                                   # primitive(22)
   40                                   # bytes(0)
                                        # ""
   00                                   # unsigned(0)
   f6                                   # primitive(22)
   81                                   # array(1)
      58 20                             # bytes(32)
         5c95a4dfddab84348bcc265a479299fbd3a2eecfa3d490985da5113e5480c7f1
   a0                                   # map(0)
   8f                                   # array(15)
      06                                # unsigned(6)
      62                                # text(2)
         656e                           # "en"
      00                                # unsigned(0)
      02                                # unsigned(2)
      69                                # text(9)
         766964656f2f6d7034             # "video/mp4"
      d8 20                             # tag(32)
         78 1c                          # text(28)
            68747470733a6578616d706c652e636f6d62696766696c652e6d7034
            # "https:example.combigfile.mp4"
      00                                # unsigned(0)
      1a 2a36ced1                       # unsigned(708234961)
      01                                # unsigned(1)
      50                                # bytes(16)
         21399320958a6f4c745dde670d95e0d8
      4c                                # bytes(12)
         c86cf2c33f21527d1dd76f5b
      40                                # bytes(0)
                                        # ""
      01                                # unsigned(1)
      58 20                             # bytes(32)
         9ab17a8cf0890baaae7ee016c7312fcc080ba46498389458ee44f0276e783163
      78 1c                             # text(28)
         3220686f757273206f66206b6579207369676e696e6720766964656f
         # "2 hours of key signing video"
```

```
message ID
  0xb267614d43e7676d28ef5b15e8676f23
    679fe365c78849d83e2ba0ae8196ec4e
```

Other dispositions of external content are also possible, for example an external GIF animation of a rocket ship could be used with a reaction disposition.

5.10. Conferencing

Joining a conference via an external URL is possible. The link could be rendered to the user, requiring a click. Alternatively the URL could be rendered the disposition could be specified as session which could be processed differently by the client (for example, alerting the user or presenting a dialog box). Further discussion of calling and conferencing functionality is out-of-scope of this document.

```
[
  null,                               / replaces                /
  h'466f6f20313138',                  / topicId = Foo 118       /
  0,                                  / expires                 /
  null,                               / inReplyTo               /
  [                                   / lastSeen (1 item)       /
     /  Attachment message                                      /
     h'b267614d43e7676d28ef5b15e8676f23679fe365c78849d83e2ba0ae8196ec4e'
  ],
  {},                                 / extensions              /
  [                                   / body (NestedPart)       /
    7,                                  / dispostion = session     /
    "",                                 / language                 /
    0,                                  / partIndex = 1st part     /
    2,                                  / cardinality = external part /
    "",                                 / contentType              /
    32("https://example.com/join/12345"),  / url                   /
    0,                                  / expires                  /
    0,                                  / size                     /
    0,                                  / encAlg = none            /
    h'',                                / key                      /
    h'',                                / nonce                    /
    h'',                                / aad                      /
    0,                                  / hashAlg = none           /
    h'',                                / content hash             /
    "Join the Foo 118 conference"       / description              /
  ]
]
```

```
87                                      # array(7)
   f6                                   # primitive(22)
   47                                   # bytes(7)
      466f6f20313138                    # "Foo 118"
   00                                   # unsigned(0)
   f6                                   # primitive(22)
   81                                   # array(1)
      58 20                             # bytes(32)
         b267614d43e7676d28ef5b15e8676f23679fe365c78849d83e2ba0ae8196ec4e
   a0                                   # map(0)
   8f                                   # array(15)
      07                                # unsigned(7)
      60                                # text(0)
                                        # ""
      00                                # unsigned(0)
      02                                # unsigned(2)
      60                                # text(0)
                                        # ""
      d8 20                             # tag(32)
         76                             # text(22)
            68747470733a6578616d706c652e636f6d3132333435
            # "https:example.com12345"
      00                                # unsigned(0)
      00                                # unsigned(0)
      00                                # unsigned(0)
      40                                # bytes(0)
                                        # ""
      40                                # bytes(0)
                                        # ""
      40                                # bytes(0)
                                        # ""
      00                                # unsigned(0)
      40                                # bytes(0)
                                        # ""
      78 1b                             # text(27)
         4a6f696e2074686520466f6f2031313820636f6e666572656e6365
         # "Join the Foo 118 conference"

   message ID
     0xb267614d43e7676d28ef5b15e8676f23
       679fe365c78849d83e2ba0ae8196ec4e
```

5.11. Topics / Threading

As popularized by the messaging application Slack, some messaging applications have a notion of a Topic or message Thread (not to be confused with message threading as used in email). Clients beginning a new "topic" populate the topicId with a unique opaque octet string. This could be the message ID of the first message sent related to the topic. Subsequent messages may include the same topicId for those messages to be associated with the same topic. The sort order for messages within a thread uses the timestamp field. If more than one message has the same timestamp, the lexically lowest message ID sorts earlier.

5.12. Delivery Reporting and Read Receipts

In instant messaging systems, read receipts typically generate a distinct indicator for each message. In some systems, the number of users in a group who have read the message is subtly displayed and the list of users who read the message is available on further inspection.

Of course, Internet mail has support for read receipts as well, but the existing message disposition notification mechanism defined for email in [RFC8098] is completely inappropriate in this context:

* notifications can be sent by intermediaries
* only one notification can be sent about a single message per recipient
* a human-readable version of the notification is expected
* each notification can refer to only one message
* it is extremely verbose

Instead we would like to be able to include status changes about multiple messages in each report, the ability to mark a message delivered, then read, then unread, then expired for example.

The format below, application/mimi-message-status is sent by one member of an MLS group to the entire group and can refer to multiple messages in that group. The format contains its own timestamp, and a list of message ID / status pairs. As the status at the recipient changes, the status can be updated in a subsequent notification. Below is the CDDL schema for message status.

```
   MessageStatusReport = [
       timestamp: Timestamp,
       statuses: [ * PerMessageStatus ]
   ]

   PerMessageStatus = [
       messageId: MessageId,
       status: baseStatus / $extStatus / unknownStatus
   ]

   baseStatus = &(
       unread: 0,
       delivered: 1,
       read: 2,
       expired: 3,
       deleted: 4,
       hidden: 5,
       error: 6
   )
   unknownStatus = &( unknown: 7..255 )

   MessageId = bstr .size 32
   Timestamp = #6.62(uint .size 8)    ; milliseconds since start of UNIX epoch
```

   * Sender user handle URL: im:bob-jones@example.com

5.12.1. Delivery Report Example

```
   [
     62(1644284703227),   / Timestamp of the report /
     [
       [
         /  Original message     /
         h'd3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501',
         2    / status = read    /
       ],
       [
         /  Reply message        /
         h'e701beee59f9376282f39092e1041b2ac2e3aad1776570c1a28de244979c71ed',
         2    / status = read    /
       ],
       [
         /  Mention message      /
         h'6b50bfdd71edc83554ae21380080f4a3ba77985da34528a515fac3c38e4998b8',
         0    / status = unread  /
       ],
       [
         /  Expiring message     /
         h'5c95a4dfddab84348bcc265a479299fbd3a2eecfa3d490985da5113e5480c7f1',
         3    / status = expired /
       ]
     ]
   ]

82                                      # array(2)
   d8 3e                                # tag(62)
      1b 0000017ed70171fb               # unsigned(1644284703227)
   84                                   # array(4)
      82                                # array(2)
         58 20                          # bytes(32)
            d3c14744d1791d02548232c23d35efa97668174ba385af066011e43bd7e51501
         02                             # unsigned(2)
      82                                # array(2)
         58 20                          # bytes(32)
            e701beee59f9376282f39092e1041b2ac2e3aad1776570c1a28de244979c71ed
         02                             # unsigned(2)
      82                                # array(2)
         58 20                          # bytes(32)
            6b50bfdd71edc83554ae21380080f4a3ba77985da34528a515fac3c38e4998b8
         00                             # unsigned(0)
      82                                # array(2)
         58 20                          # bytes(32)
            5c95a4dfddab84348bcc265a479299fbd3a2eecfa3d490985da5113e5480c7f1
         03                             # unsigned(3)
```

6. Support for Specific Media Types

6.1. MIMI Required and Recommended media types

As the MIMI Content container is just a container, the plain text or rich text messages sent inside, and any image or other formats needs to be specified. Clients compliant with MIMI MUST be able to receive the following media types:

* application/mimi-content -- the MIMI Content container format (described in this document)
* text/plain;charset=utf-8
* text/markdown;variant=GFM -- Github Flavored Markdown [GFM])

Note that it is acceptable to render the contents of a received markdown document as plain text.

The following MIME types are RECOMMENDED:

* text/markdown;variant=CommonMark -- CommonMark (https://spec.commonmark.org/0.30)
* text/html
* application/mimi-message-status -- (described in this document)
* image/jpeg
* image/png

Clients compliant with this specification must be able to download ExternalParts with http and https URLs, and decrypt downloaded content encrypted with the AES-128-GCM AEAD algorithm.

6.2. Use of proprietary media types

As most messaging systems are proprietary, standalone systems, it is useful to allow clients to send and receive proprietary formats among themselves. Using the functionality in the MIMI Content container, clients can send a message using the basic functionality described in this document AND a proprietary format for same-vendor clients simultaneously over the same group with end-to-end encryption. An example is given in the Appendix.

7. IANA Considerations

7.1. MIME subtype registration of application/mimi-content

This document proposes registration of a media subtype with IANA.

Type name: application

Subtype name: mimi-content

Required parameters: none

Optional parameters: none

Encoding considerations:
This message type should be encoded as binary data

Security considerations:
See Section A of RFC XXXX

Interoperability considerations:
See Section Y.Z of RFC XXXX

Published specification: RFC XXXX

Applications that use this media type:
Instant Messaging Applications

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A
Magic number(s): N/A
File extension(s): N/A
Macintosh file type code(s): N/A

Person & email address to contact for further information:
IETF MIMI Working Group mimi@ietf.org

7.2. MIME subtype registration of application/mimi-message-status

This document proposes registration of a media subtype with IANA.

Type name: application

Subtype name: mimi-message-status

Required parameters: none

Optional parameters: none

Encoding considerations:
This message type should be encoded as binary data

Security considerations:
See Section A of RFC XXXX

Interoperability considerations:
See Section Y.Z of RFC XXXX

Published specification: RFC XXXX

Applications that use this media type:
Instant Messaging Applications

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A
Magic number(s): N/A
File extension(s): N/A
Macintosh file type code(s): N/A

Person & email address to contact for further information:
IETF MIMI Working Group mimi@ietf.org

8. Security Considerations

8.1. General handling

The following cases are examples of nonsensical values that most likely represent malicious messages. These should be logged and discarded.

* sender of the message
  - where the apparent sender is not a member of the target MLS group
* message IDs
  - which duplicate another message ID already encountered
* timestamps

    - received more than a few minutes in the future, or
    - before the first concrete syntax of this document is published
    - before the room containing them was created
  * inReplyTo
    - inReplyTo.hash-alg is none even when the inReplyTo.message is present
    - inReplyTo.hash-alg is an unknown value
    - the length of inReplyTo.replyToHash does not correspond to the algorithm specified in inReplyTo.hash-alg
  * topicId
    - the topicId is very long (greater than 4096 octets)
    - a topic is specified, but an inReplyTo or replaces field refers to a message outside of the topic
  * expires
    - refers to a date more than a year in the past
    - refers to a date more than a year in the future
  * lastSeen
    - is empty, but the sender has previously sent messages in the room
    - results in a loop
    - refers to an excessive number of lastSeen messages simultaneously (contains more than 65535 message IDs). (Note that a popular message sent in a large group can result in thousands of reactions in a few hundred milliseconds.)
  * body
    - has too many body parts (more than 1024)
    - is nested too deeply (more than 4 levels deep)
    - is too large (according to local policy)
    - has an unknown PartSemantics value
    - contains partIndex values which are not continuous from zero

For the avoidance of doubt, the following cases may be examples of legitimate use cases, and should not be considered the result of a malicious sender.

  * message IDs
    - where inReplyTo.message or replaces refer to an unknown message. Such a message could have been sent before the local client joined.
  * lastSeen
    - refers to an unknown message
    - is empty for the sender’s first message sent in the room
  * body
    - where a body part contains an unrecognized Disposition value. The unknown value should be treated as if it where render.
    - where a contentType is unrecognized or unsupported.
    - where a language tag is unrecognized or unsupported.

8.2. Validation of timestamp

The timestamp is the time a message is accepted by the hub provider. As such, the hub provider can manipulate the timestamp, and the sending provider can delay sending messages selectively to cause the timestamp on a hub to be later.

> *TODO*: Discuss how to sanity check lastSeen, timestamp and the MLS epoch and generation, and the limitations of this approach.

8.3. Alternate content rendering

This document includes a mechanism where the sender can offer alternate versions of content in a single message. For example, the sender could send:

* an plain text and an HTML version of a text message
* a thumbnail preview and link to a high-resolution image or video
* versions of the same message in multiple languages
* an PNG image and a scalable vector graphics version of a line drawing

A malicious client could use this mechanism to send content that will appear different to a subset of the members of a group and possibly elicit an incorrect or misleading response.

Message as seen by Alice (manager)
Xavier: Do you want me to reserve a room for the review meeting?

Message as seen by Bob (Alice’s assistant)
Xavier: @Bob I need to pickup Alice’s Ferarri keys. She’ll confirm

Reply sent by Alice
Alice: Yes please.

8.4. Link and Mention handling

Both Markdown and HTML support links. Using the example of an https link, if the rendered text and the link target match exactly or are canonically equivalent, there is no need for confirmation if the end user selects the link.

example.com/foobar
https://example.com/foobar
[https://example.com:443/foobar](https://example.com/foobar)

However, if the link text is different, or the scheme is downgraded from https to http, the user should be presented with an alert warning that the text is not the same.

[https://example.com/foobar](https://spearphishers.example/foobar)
[https://example.com/foobar](http://example.com/foobar)

An IM URI link to a user who has a member client in the MLS group in which the message was sent is considered a mention. Clients may support special rendering of mentions instead of treating them like any other type of link. In Markdown and HTML, the display text portion of a link is considered a rendering hint from the sender to the receiver of the message. The receiver should use local policy to decide if the hint is an acceptable local representation of the user represented by the link itself. If the hint is not an acceptable representation, the client should instead display its canonical representation for the user.

For example, in the first examples, the sender expresses no preference about how to render the mention. In the second example, the sender requests that the mention is rendered as the literal URI. In the third example, the sender requests the canonical handle for Alice. In the fourth example, the sender requests Alice's first name.

<im:alice-smith@example.com>
im:alice-smith@example.com
[@AliceSmith](im:alice-smith@example.com)
[Alice](im:alice-smith@example.com)

Note that in some clients, presence of a mention for the local user may result in a different notification policy.

If the client does not support special rendering of mentions, the application, should render the text like any other link.

8.5. Delivery and Read Receipts

Delivery and Read Receipts can provide useful information inside a group, or they can reveal sensitive private information. In many IM systems there is are per-group policies for and/or delivery read receipts:

* they are required
* they are permitted, but optional
* they are forbidden

In the first case, everyone in the group would have to claim to support read receipts to be in the group and agree to the policy of sending them whenever a message was read. A user who did not wish to send read receipts could review the policy (automatically or manually) and choose not to join the group. Of course, requiring read receipts is a cooperative effort just like using self-deleting messages. A malicious client could obviously read a message and not send a read receipt, or send a read receipt for a message that was never rendered. However, cooperating clients have a way to agree that they will send read receipts when a message is read in a specific group.

In the second case, sending a read receipt would be at the discretion of each receiver of the message (via local preferences).

9. References

9.1. Normative References

[GFM] GitHub, "GitHub Flavored Markdown Spec, Version 0.29-gfm", 6 March 2019, <https://github.github.com/gfm/>.

[I-D.ietf-mls-extensions] Robert, R., "The Messaging Layer Security (MLS) Extensions", Work in Progress, Internet-Draft, draft-ietf-mls-extensions-04, 24 April 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-mls-extensions-04>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3862] Klyne, G. and D. Atkins, "Common Presence and Instant Messaging (CPIM): Message Format", RFC 3862, DOI 10.17487/RFC3862, August 2004, <https://www.rfc-editor.org/info/rfc3862>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/info/rfc5116>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC7763] Leonard, S., "The text/markdown Media Type", RFC 7763, DOI 10.17487/RFC7763, March 2016, <https://www.rfc-editor.org/info/rfc7763>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC9420] Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., and K. Cohn-Gordon, "The Messaging Layer Security (MLS) Protocol", RFC 9420, DOI 10.17487/RFC9420, July 2023, <https://www.rfc-editor.org/info/rfc9420>.

[W3C.CR-html52-20170808] Faulkner, S., Eicholz, A., Leithead, T., Danilo, A., and S. Moon, "HTML 5.2", World Wide Web Consortium CR CR-html52-20170808, 8 August 2017, <https://www.w3.org/TR/2017/CR-html52-20170808>.

9.2. Informative References

[DoubleRatchet] Perrin, T. and M. Marlinspike, "The Double Ratchet Algorithm", 20 November 2016, <https://signal.org/docs/specifications/doubleratchet/>.

[I-D.ietf-cbor-edn-literals] Bormann, C., "CBOR Extended Diagnostic Notation (EDN): Application-Oriented Literals, ABNF, and Media Type", Work in Progress, Internet-Draft, draft-ietf-cbor-edn-literals-09, 18 May 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-edn-literals-09>.

[I-D.ietf-mimi-arch]
Barnes, R., "An Architecture for More Instant Messaging Interoperability (MIMI)", Work in Progress, Internet-Draft, draft-ietf-mimi-arch-00, 2 April 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-mimi-arch-00>.

[I-D.mahy-mimi-identity]
Mahy, R., "More Instant Messaging Interoperability (MIMI) Identity Concepts", Work in Progress, Internet-Draft, draft-mahy-mimi-identity-02, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-mahy-mimi-identity-02>.

[I-D.ralston-mimi-terminology]
Ralston, T., "MIMI Terminology", Work in Progress, Internet-Draft, draft-ralston-mimi-terminology-03, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ralston-mimi-terminology-03>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC2183] Troost, R., Dorner, S., and K. Moore, Ed., "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, DOI 10.17487/RFC2183, August 1997, <https://www.rfc-editor.org/info/rfc2183>.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", RFC 3156, DOI 10.17487/RFC3156, August 2001, <https://www.rfc-editor.org/info/rfc3156>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/rfc6120>.

[RFC8098] Hansen, T., Ed. and A. Melnikov, Ed., "Message Disposition Notification", STD 85, RFC 8098, DOI 10.17487/RFC8098, February 2017, <https://www.rfc-editor.org/info/rfc8098>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[RFC9078] Crocker, D., Signes, R., and N. Freed, "Reaction: Indicating Summary Reaction to a Message", RFC 9078, DOI 10.17487/RFC9078, August 2021, <https://www.rfc-editor.org/info/rfc9078>.

Appendix A. CDDL Schemas

Below are Concise Data Definition Language (CDDL) [RFC8610] schemas for the formats described in the body of the document.

A.1. Complete Message Format Schema

```
mimiContent = [
  replaces: null / MessageId,
  topicId: bstr,
  expires: uint .size 4,
  inReplyTo: null / InReplyTo,
  lastSeen: [* MessageId],
  extensions: {* name => value },
  nestedPart: NestedPart
]

NestedPart = [
  disposition: baseDispos / $extDispos / unknownDispos,
  language: tstr,
  partIndex: uint .size 2,
  ( NullPart // SinglePart // ExternalPart // MultiPart)
]

NullPart = ( cardinality: nullpart )

SinglePart = (
    cardinality: single,
    contentType: tstr,
    content: bstr
)

ExternalPart = (
    cardinality: external,
```

```
      contentType: tstr,
      url: uri,
      expires: uint .size 4,
      size: uint .size 8,
      encAlg: uint .size 2,
      key: bstr,
      nonce: bstr,
      aad: bstr,
      hashAlg: uint .size 1,
      contentHash: bstr,
      description: tstr
  )

  MultiPart = (
      cardinality: multi,
      partSemantics: chooseOne / singleUnit / processAll,
      parts: [2* NestedPart]
  )

  InReplyTo = [
    message: MessageId,
    hashAlg: uint .size 8,
    hash: bstr
  ]

  baseDispos = &(
      unspecified: 0,
      render: 1,
      reaction: 2,
      profile: 3,
      inline: 4,
      icon: 5,
      attachment: 6,
      session: 7,
      preview: 8
  )
  unknownDispos = &( unknown: 9..255 ) ; Note: any ext_dispos take precedence

  MessageId = bstr .size 32          ; MessageId is derived from SHA256 hash
  name = tstr .size (1..255)
  value = bstr .size (0..4095)

  nullpart = 0
  single   = 1
  external = 2
  multi    = 3

  chooseOne  = 0
```

```
singleUnit = 1
processAll = 2
```

A.2. Implied Message Fields

Below is a CDDL schema for the implied message fields.

```
MessageDerivedValues = [
    messageId: MessageId,               ; sha256 hash of message ciphertext
    hubAcceptedTimestamp: Timestamp,
    mlsGroupId: bstr,
    senderLeafIndex: uint .size 4,
    senderClientUrl: uri,
    senderUserUrl: uri,
    roomUrl: uri
]

MessageId = bstr .size 32
Timestamp = #6.62(uint .size 8)    ; milliseconds since start of UNIX epoch
```

A.3. Delivery Report Format

```
MessageStatusReport = [
    timestamp: Timestamp,
    statuses: [ * PerMessageStatus ]
]

PerMessageStatus = [
    messageId: MessageId,
    status: baseStatus / $extStatus / unknownStatus
]

baseStatus = &(
    unread: 0,
    delivered: 1,
    read: 2,
    expired: 3,
    deleted: 4,
    hidden: 5,
    error: 6
)
unknownStatus = &( unknown: 7..255 )

MessageId = bstr .size 32
Timestamp = #6.62(uint .size 8)    ; milliseconds since start of UNIX epoch
```

Appendix B. Multipart examples

In a heterogenous group of IM clients, it is often desirable to send more than one media type as alternatives, such that IM clients have a choice of which media type to render. For example, imagine an IM group containing a set of clients which support a common video format and a subset which only support animated GIFs. The sender could use a MultiPart NestablePart with chooseOne semantics containing both media types. Every client in the group chat could render something resembling the media sent. This is analogous to the multipart/alternative [RFC2046] media type.

Likewise it is often desirable to send more than one media type intended to be rendered together as in (for example a rich text document with embedded images), which can be represented using a MultiPart NestablePart with processAll semantics. This is analogous to the multipart/mixed [RFC2046] media type.

Note that there is a minor semantic difference between multipart/alternative and MultiPart with chooseOne semantics. In multipart/alternative, the parts are presented in preference order by the sender. With MultiPart the receiver chooses its "best" format to render according to its own preferences.

B.1. Proprietary and Common formats sent as alternatives

This shows sending a message containing both this profile and a proprietary messaging format simultaneously.

```
[
  null,                              / replaces          /
  h'',                               / topicId           /
  0,                                 / expires           /
  null,                              / inReplyTo         /
  [],                                / lastSeen          /
  {},                                / extensions        /
  [
    1,                             / disposition = render      /
    "",                            / language                  /
    0,                             / partIndex = 0 (1st part)  /
    3,                             / cardinality = multi       /
    0,                             / partSemantics = chooseOne /
    [
      [
        1,                              / disposition = render /
        "",                             / language             /
        1,                              / partIndex = 1        /
        1,                              / cardinality = single /
        "text/markdown;variant=GFM",    / contentType          /
        '# Welcome!'                    / content              /
      ],
      [
        1,                              / disposition = render /
        "",                             / language             /
        2,                              / partIndex = 1        /
        1,                              / cardinality = single /
                                        / contentType          /
        "application/vnd.examplevendor-fancy-im-message",
        h'dc861ebaa718fd7c3ca159f71a2001'       / content   /
      ]
    ]
  ]
]
```

B.2. Mulitple Reactions Example

This example shows sending a reaction with multiple separate emojis.

```
[
  null,                              / replaces          /
  h'',                               / topicId           /
  0,                                 / expires           /
  null,                              / inReplyTo         /
  [],                                / lastSeen          /
  {},                                / extensions        /
  [
    1,                             / disposition = render       /
    "",                            / language                   /
    0,                             / partIndex = 0 (1st part)   /
    3,                             / cardinality = multi        /
    2,                             / partSemantics = processAll /
    [
      [
        1,                                 / disposition = render /
        "",                                / language             /
        1,                                 / partIndex = 1        /
        1,                                 / cardinality = single /
        "text/plain;charset=utf-8",        / contentType          /
        h'E2 9D A4'                 / content = Unicode "heart" /
      ],
      [
        1,                                 / disposition = render /
        "",                                / language             /
        2,                                 / partIndex = 2        /
        1,                                 / cardinality = single /
        "text/plain;charset=utf-8",        / contentType          /
        h'F0 9F A5 B3'       / content = Unicode "party face" /
      ],
      [
        1,                                 / disposition = render /
        "",                                / language             /
        3,                                 / partIndex = 3        /
        1,                                 / cardinality = single /
        "text/plain;charset=utf-8",        / contentType          /
        h'F0 9F A4 9E'  / content = Unicode "fingers crossed" /
      ]
    ]
  ]
]
```

B.3. Complicated Nested Example

This example shows separate GIF and PNG inline images with English and French versions of an HTML message. A summary of the 11 parts are shown below.

```
Part  Description
 0    choose either GIF or PNG
 1      (with GIF) process all
 2        choose either English or French
 3          English
 4          French
 5        GIF
 6      (with PNG) process all
 7        choose either English or French
 8          English
 9          French
10        PNG

[
  null,                              / replaces          /
  h'',                               / topicId           /
  0,                                 / expires           /
  null,                              / inReplyTo         /
  [],                                / lastSeen          /
  {},                                / extensions        /
  [
    1,     / disposition = render /
    "",    / language /
    0,     / partIndex = 0 (1st part)  /
    3,     / cardinality = multi       /
    0,     / partSemantics = chooseOne /
    [
      [
        1,   / disposition = render /
        "",  / language /
        1,   / partIndex = 1 /
        3,   / cardinality = multi /
        2,   / partSemantics = processAll /
        [
          [
            1,   / disposition = render /
            "",  / language /
            2,   / partIndex = 2 /
            3,   / cardinality = multi /
            0,   / partSemantics = chooseOne /
            [
              [
                1,   / disposition = render /
                "en",  / language /
                3,   / partIndex = 3 /
                1,   / cardinality = single /
                "text/html;charset=utf-8",   / contentType /
                / content /
```

```
                '<html><body><h1>Welcome!</h1>'
                '<img src="cid:5" alt="Welcome image"/>'
                '</body></html>'
              ],          / English HTML /
              [
                1,   / disposition = render /
                "fr",  / language /
                4,   / partIndex = 4 /
                1,   / cardinality = single /
                "text/html;charset=utf-8",   / contentType /
                / content /
                '<html><body><h1>Bienvenue!</h1>'
                '<img src="cid:5" alt="Image bienvenue"/>'
                  '</body></html>'
              ]           / French HTML /
            ]
          ],          / English or French HTML (refers to GIF)/
          [
            4,   / disposition = inline /
            "",  / language /
            5,   / partIndex = 5 /
            1,   / cardinality = single /
            "image/gif",   / contentType /
            h'028f83c894ca744f'       / content /
          ]               / GIF /
        ]
      ],                  / GIF with English or French HTML /
      [
        1,   / disposition = render /
        "",  / language /
        6,   / partIndex = 6 /
        3,   / cardinality = multi /
        2,   / partSemantics = processAll /
        [
          [
            1,   / disposition = render /
            "",  / language /
            7,   / partIndex = 7 /
            3,   / cardinality = multi /
            0,   / partSemantics = chooseOne /
            [
              [
                1,   / disposition = render /
                "en",  / language /
                8,   / partIndex = 8 /
                1,   / cardinality = single /
                "text/html;charset=utf-8",   / contentType /
                / content /
```

```
                  '<html><body><h1>Welcome!</h1>'
                  '<img src="cid:10" alt="Welcome image"/>'
                  '</body></html>'
                ],          / English HTML /
                [
                  1,   / disposition = render /
                  "fr",  / language /
                  9,   / partIndex = 9 /
                  1,   / cardinality = single /
                  "text/html;charset=utf-8",   / contentType /
                  / content /
                  '<html><body><h1>Bienvenue!</h1>'
                  '<img src="cid:10" alt="Image bienvenue"/>'
                  '</body></html>'
                ]           / French HTML /
              ]
            ],          / English or French HTML (refers to PNG)/
            [
              4,   / disposition = inline /
              "",  / language /
              10,   / partIndex = 10 /
              1,   / cardinality = single /
              "image/png",
              h'6963cff36275fdb8'       / content /
            ]               / PNG /
          ]
        ]                 / PNG with English or French HTML /
      ]           / GIF or PNG (with English or French HTML) /
    ]
  ]
```

Appendix C. Changelog

RFC Editor, please remove this entire section.

C.1. Changes between draft-mahy-mimi-content-01 and draft-mahy-mimi-content-02

* made semantics abstract (C++ structs) instead of using CPIM or MIME headers

C.2. Changes between draft-mahy-mimi-content-02 and draft-ietf-mimi-content-00

* replaced threadId with topicId
* inReplyTo now has a hash of the referenced message
* clarified that replies are always to a specific version of a modified message

* changed timestamp to a whole number of milliseconds since the epoch to avoid confusion
* added Security Considerations section
* added IANA Considerations section
* added change log

C.3. Changes between draft-ietf-mimi-content-00 and draft-ietf-mimi-content-01

* created new abstract format for attachment information, instead of using message/external-body
* added discussion of encrypting external content
* clarified the difference between render and inline dispositions
* created a way for the messageId and timestamp to be shared in the MLS additional authenticated data field
* expanded discussion of what can and should be rendered when a mention is encountered; discussed how to prevent confusion attacks with mentions.
* added a lastSeen field used to ensure a more consistent sort order of messages in a room.

C.4. Changes between draft-ietf-mimi-content-01 and draft-ietf-mimi-content-02

* consensus at IETF 118 was to use a hash of the ciphertext in lieu of the message ID
* consensus at IETF 118 was to use the hub accepted timestamp for protocol actions like sorting
* Updated author's address

C.5. Changes between draft-ietf-mimi-content-02 and draft-ietf-mimi-content-03

* added hash of content to external content
* replaced abstract syntax with concrete TLS Presentation Language and CBOR syntaxes

C.6. Changes between draft-ietf-mimi-content-03 and draft-ietf-mimi-content-04

* use CBOR as the binary encoding
* add multipart examples

Author's Address

Rohan Mahy
Rohan Mahy Consulting Services
Email: rohan.ietf@gmail.com