MIMI
Internet-Draft
Intended status: Informational
Expires: 3 September 2026

R. Mahy
Rohan Mahy Consulting Services

2 March 2026

More Instant Messaging Interoperability (MIMI) message content
draft-ietf-mimi-content-08

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 3 September 2026.

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 can be found in [I-D.ietf-mimi-arch].

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

Delivery notifications and read receipts are addressed in {{?I-D.mahy-mimi-message-status}}. Calling and conferencing will also be addressed in another document.

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

The MIMI content format relies heavily on message IDs to refer to other messages, to reply, react, edit, delete, and report on the status of messages. Every MIMI content message contains a 16-octet per-message cryptographically random salt, and has a 32-octet message ID which is calculated from the hash of the message (including the salt), the sender URI, and the room URI.

Calculation of the message ID works as follows. The first octet of the MessageID is the hash function ID from the IANA hash algorithm registry (https://www.iana.org/assignments/named-information/named-information.xhtml#hash-alg). The senderUriLength and roomUriLength are big-endian unsigned 16-bit integers representing the length of the sender URI, and the room URI, respectively. The senderUriLength, sender URI, roomUriLength, room URI, the entire MIMI message content (including the salt), and the salt (again) are all concatenated, and then hashed with the algorithm identified in the first octet. The first 31 octets of the hash_output is appended to the hash function ID.

```
hash_output = hash( senderUriLength || senderUri ||
                    roomUriLength   || roomUri   ||
                    message         || salt )
messageId = hashAlg || hash_output[0..30]
```

The MIMI content format uses the SHA-256 hash algorithm (identifier 0x01) by default, regardless of the hash algorithm of the cipher suite of a room's MLS group. The initial octet allows the MIMI protocol to deprecate SHA-256 and specify a new default algorithm in the future (for example if a practical birthday attack on SHA_256 becomes feasible).

| The salt is duplicated in the input to the hash to avoid a SHA-256 length extension attack.

3.4. Accepted Timestamp

As described in 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). The accepted timestamp MUST be available to each receiving MIMI client. The client can use it for fine grain sorting of messages into a consistent order.

3.5. 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).

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 = [
  salt: bstr .size 16,
  replaces: null / MessageId,       ; {1}
  topicId: bstr,                    ; {2}
  expires: null / Expiration,       ; {3}
  inReplyTo: null / MessageId,      ; {4}
  mimiExtensions: extensions,       ; {6}
  nestedPart: NestedPart            ; {7}
]

MessageId = bstr .size 32
```

The first data field is the per-message unique salt which MUST be cryptographically random. An example algorithm for generating the salt is described in Section 9.2.

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 null, 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 either a specific timestamp in the future, or a relative amount of time after the receiving client reads the message. Indicate a message with no specific expiration time with the value null. If non-null, the data field is an array of two items.

```
Expiration = [
    relative: bool,
    time: uint .size 4
]
```

The first is a boolean indicating if the time is relative (true) or absolute (false). The second is an unsigned integer. If relative, it is the whole number of seconds the message should be visible before it is deleted. If absolute, it is the number of seconds after the start of the UNIX epoch, at which point the message should be deleted. 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. Otherwise, the receiver assumes that the current message has not identified any special reply relationship with another previous message. The message id of the referenced message is also 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 with inReplyTo message, the client checks if the referenced message is itself inReplyTo another message. If so, it continues following the referenced messages, checking that the message ID of none of the referenced messages "loop" back to a message later in the inReplyTo chain.

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

Message ordering is provided by the Hub in the form of the accepted timestamp. A malicious hub may be able to manipulate the order of messages.

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 somehow referencing (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.

If clients want to detect messages sent out of order by the hub, they require notification of message delivery at the MLS level (ex: the AppAck mechanism provided in [I-D.ietf-mls-extensions]) or at the MIMI level, such as the format defined in {{?I-D.mahy-mimi-message-status}}.

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 CBOR map key which is a positive integer, negative integer, or text string containing between 1 and 255 octets of UTF-8. The value can be any CBOR (including combinations of maps and arrays). The message content mimiExtensions field MUST NOT include more than one extension field with the same map key.

```
extensions = {
  ? &(senderUri: 1) ^ => tstr,
  ? &(roomUri: 2) ^ => tstr,
  * $$otherKnownExtensions,
  * unknownExtension
}

unknownExtension = (
   name => value
)
name = int / tstr .size (1..255)
value = any .size (0..4095)
```

An IANA registry Section 8.2.1 is defined for positive integer keys. Negative integer and text string keys are only for private use.

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}
     ( NullPart // SinglePart // ExternalPart // MultiPart)
   ]

   NullPart = ( cardinality: nullpart )

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

   ExternalPart = (
       cardinality: external,
       contentType: tstr,
       url: tstr,
       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,
       filename: 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]. Disposition values are defined in Section 8.2.2.

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

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 implied part index, 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 Appendix B.3 for an example of how the part index is calculated.

The partIndex can be used inside a content ID URI [RFC2392] in a "container" part (for example HTML, Markdown, vCard [RFC6350], or iCal [RFC5545]) to reference another part inside the same MIMI message. In a MIMI message it has the form cid:_partIndex_@local.invalid. This format of the content ID URI in MIMI MUST only reference the partIndex of a SinglePart or ExternalPart.

When processing a MultiPart nested structure, the client can start from the body in the MIMI content (the "top-level" or "root") and evaluate any chooseOne semantics MultiPart elements, effectively discarding non-chosen Parts. The remaining Parts might still reference each other in content ID URI. To prevent processing a Part more than once, the client can handle the remaining Parts in order, skipping any Parts already referenced by a previously handled Part. This process of skipping already processed Parts is respected regardless of the disposition of the Part.

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
    filename: tstr         ; an optional suggested filename
)
```

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 is the hub accepted timestamp, which can be represented either as milliseconds since the start of the UNIX epoch, or for future extensibility as a CBOR extended time tag as defined in {{Section 3 of !RFC9581}}. There are also two fields the implementation can definitely derive: (the MLS group ID {12}, and the leaf index of the sender {13}). Many implementations could also determine one or more of: the sender's client identifier URL {14}, the user identifier URL of the credential associated with the sender {15}, and the identifier URL for the MIMI room {16}.

```
MessageDerivedValues = [
    messageId: MessageId,
    hubAcceptedTimestamp: Timestamp,
    mlsGroupId: bstr,                   ; value always available {12}
    senderLeafIndex: uint .size 4,      ; value always available {13}
    senderClientUrl: uri                ; {14},
    senderUserUrl: uri,                 ; "From" {15}
    roomUrl: uri                        ; "To"   {16}
]

MessageId = bstr .size 32
Timestamp = MsecsSinceEpoch / ExtendedTime
; milliseconds since start of UNIX epoch
MsecsSinceEpoch = uint .size 8
; extended time from RFC9581
ExtendedTime = #6.1001({* name => value })
```

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 room. The following values are derived from the client, except for the hub received timestamp, which needs to be made available to the client by its provider:

* Sender MLS leaf index: 4
* Sender MLS client ID URL: mimi://example.com/d/3b52249d-68f9-45ce-8bf5-c799f3cad7ec/0003
* MLS group ID: 7u4NEqe1tbeBFa0aHdsTgRyD_XOHxD5meZpZS-7aJr8
* The MIMI room name: "Engineering Team"
* The Hub received timestamp: 1644387225019 = 2022-02-09T06:13:45.019Z

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

```
# ORIGINAL
# message ID = h'017ce54837404c3696e0c747b985cb17
#                2716d0ed0a3d249ca63ace7d82a096f4'
# timestamp  = 1644387225019 = 2022-02-09T06:13:45.019Z
[
  h'5eed9406c2545547ab6f09f20a18b003', # salt
  null,                                # replaces
  h'',                                 # topicId
  null,                                # expires = never
  null,                                # inReplyTo
  {                                    # extensions
    1: "mimi://example.com/u/alice-smith",
    2: "mimi://example.com/r/engineering_team"
  },
  [                                    # body (NestedPart)
    1,                                   # disposition = render
    "",                                  # language
    1,                                   # cardinality = single part
    "text/markdown;variant=GFM-MIMI",    # contentType
                                         # content
    'Hi everyone, we just shipped release 2.0. __Good  work__!'
  ]
]
```

```
  87                                      # array(7)
     50                                   # bytes(16)
        5eed9406c2545547ab6f09f20a18b003
     f6                                   # primitive(22)
     40                                   # bytes(0)
     f6                                   # primitive(22)
     f6                                   # primitive(22)
     a2                                   # map(2)
        01                                # unsigned(1)
        78 20                             # text(32)
           6d696d693a2f2f6578616d706c652e63
           6f6d2f752f616c6963652d736d697468
           # "mimi://example.com/u/alice-smith"
        02                                # unsigned(2)
        78 25                             # text(37)
           6d696d693a2f2f6578616d706c652e63
           6f6d2f722f656e67696e656572696e67
           5f7465616d
           # "mimi://example.com/r/engineering_team"
     85                                   # array(5)
        01                                # unsigned(1)
        60                                # text(0)
                                          # ""
        01                                # unsigned(1)
        78 1e                             # text(30)
           746578742f6d61726b646f776e3b7661
           7269616e743d47464d2d4d494d49
           # "text/markdown;variant=GFM-MIMI"
        58 39                             # bytes(57)
           486920657665727 96f6e652c20776520
           6a757374207368697070656420726c
           6561736520322e302e205f5f476f6f64
           2020776f726b5f5f21
           # "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'017ce54837404c3696e0c747b985cb17
    2716d0ed0a3d249ca63ace7d82a096f4',

  / hubAcceptedTimestamp = 2022-02-08T22:13:45.019Z /
  1644387225019,

  / mlsGroupId /
  h'eeee0d12a7b5b5b78115ad1a1ddb1381
    1c83fd7387c43e66799a594beeda26bf',

  / senderLeafIndex /
  4,

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

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

  / roomUrl /
  "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 only the user handle URL, message ID, and hub received timestamp are provided (in comments at the beginning of the annotated EDN). The annotated EDN follows, then the pretty printed CBOR.

```
   # REPLY
   # message ID = h'015354973c2b65ca937bf1e035ae53a5
   #                ab80e947afa43d46920d4202e5cc0b27'
   # timestamp  = 1644387237492 = 2022-02-09T06:13:57.492Z
   [
     h'11a458c73b8dd2cf404db4b378b8fe4d', # salt
     null,                                # replaces
     h'',                                 # topicId
     null,                                # expires
     h'017ce54837404c3696e0c747b985cb17   # inReplyTo =
       2716d0ed0a3d249ca63ace7d82a096f4', #   Original
     {                                    # extensions
       1: "mimi://example.com/u/bob-jones",
       2: "mimi://example.com/r/engineering_team"
     },
     [                                    # body (NestedPart)
       1,                                   # disposition = render
       "",                                  # language
       1,                                   # cardinality = single part
       "text/markdown;variant=GFM-MIMI",    # contentType
                                            # content
       'Right on! _Congratulations_ \'all!'
     ]
   ]
```

```
87                                      # array(7)
   50                                   # bytes(16)
      11a458c73b8dd2cf404db4b378b8fe4d
   f6                                   # primitive(22)
   40                                   # bytes(0)
   f6                                   # primitive(22)
   58 20                                # bytes(32)
      01b0084467273cc43d6f0ebeac13eb84
      229c4fffe8f6c3594c905f47779e5a79
   a2                                   # map(2)
      01                                # unsigned(1)
      78 1e                             # text(30)
         6d696d693a2f2f6578616d706c652e63
         6f6d2f752f626f622d6a6f6e6573
         # "mimi://example.com/u/bob-jones"
      02                                # unsigned(2)
      78 25                             # text(37)
         6d696d693a2f2f6578616d706c652e63
         6f6d2f722f656e67696e656572696e67
         5f7465616d
         # "mimi://example.com/r/engineering_team"
   85                                   # array(5)
      01                                # unsigned(1)
      60                                # text(0)
                                        # ""
      01                                # unsigned(1)
      78 1e                             # text(30)
         746578742f6d61726b646f776e3b7661
         7269616e743d47464d2d4d494d49
         # "text/markdown;variant=GFM-MIMI"
      58 21                             # bytes(33)
         5269676874206f6e21205f436f6e6772
         6174756c6174696f6e735f2027616c6c
         21
         # "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 the three octet sequence 0xe299a5. Often a single text reaction consists of multiple Unicode characters. For example, the five Unicode characters for a medium skin-toned, female health worker (U+1F469 U+1F3FD U+200D U+2695

U+FE0F: woman, medium skin tone, combined with, staff of Aesculapius, present as image) are typically rendered as a single glyph. This sequence is expressed in UTF-8 as the 17-octet sequence 0xf09f91a9f09f8fbde2808de29a95efb88f.

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 be another media type (ex: image, audio, or video), although not as universally implemented in instant messaging systems.

Note that many systems allow multiple independent reactions per sender. When multiple reactions are supported, each reaction could be represented either as a separate part inside a MultiPart with processAll semantics, or as a separate message. The ensemble of multiple text reactions MUST NOT be represented as a single text part. Due to the semantics of Unicode characters, concatenating reactions could change their semantics, resulting in unexpected or misleading rendering, or even avenues for attack.

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

```
# REACTION
# message ID = h'0158c4288911e50a8f6be3f47746b668
#                2f10fd91bc8c05557aa589a3157aff68'
# timestamp  = 1644387237728 = 2022-02-08T22:13:57.728Z
[
  h'd37bc0e6a8b4f04e9e6382375f587bf6', # salt
  null,                                # replaces
  h'',                                 # topicId
  null,                                # expires
  h'017ce54837404c3696e0c747b985cb17   # inReplyTo =
    2716d0ed0a3d249ca63ace7d82a096f4', #   Original
  {                                    # extensions
    1: "mimi://example.com/u/cathy-washington",
    2: "mimi://example.com/r/engineering_team"
  },
  [                                    # body (NestedPart)
    2,                                   # disposition = reaction
    "",                                  # language
    1,                                   # cardinality = single
    "text/plain;charset=utf-8",          # contentType
    '笑、'                                  # content = U+2665 (heart)
  ]
]
```

```
87                                      # array(7)
   50                                   # bytes(16)
      d37bc0e6a8b4f04e9e6382375f587bf6
   f6                                   # primitive(22)
   40                                   # bytes(0)
   f6                                   # primitive(22)
   58 20                                # bytes(32)
      01b0084467273cc43d6f0ebeac13eb84
      229c4fffe8f6c3594c905f47779e5a79
   a2                                   # map(2)
      01                                # unsigned(1)
      78 25                             # text(37)
         6d696d693a2f2f6578616d706c652e63
         6f6d2f752f63617468792d7761736869
         6e67746f6e
         # "mimi://example.com/u/cathy-washington"
      02                                # unsigned(2)
      78 25                             # text(37)
         6d696d693a2f2f6578616d706c652e63
         6f6d2f722f656e67696e656572696e67
         5f7465616d
         # "mimi://example.com/r/engineering_team"
   85                                   # array(5)
      02                                # unsigned(2)
      60                                # text(0)
                                        # ""
      01                                # unsigned(1)
      78 18                             # text(24)
         746578742f706c61696e3b6368617273
         65743d7574662d38
         # "text/plain;charset=utf-8"
      43                                # bytes(3)
         e29da4                         # "笑、"
```

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.

```
# MENTION
# message ID = h'018d825adf9f6be00dcafc5704c4102f
#               5022e74219d0b603e4ba7622654042af'
# timestamp  = 1644387243008 = 2022-02-08T22:14:03.008Z
[
  h'04f290e215d0f82d1750bfa8b7dc089d', # salt
  null,                                # replaces
  h'',                                 # topicId
  null,                                # expires
  h'017ce54837404c3696e0c747b985cb17   # inReplyTo =
    2716d0ed0a3d249ca63ace7d82a096f4', #   Original
  {                                    # extensions
    1: "mimi://example.com/u/cathy-washington",
    2: "mimi://example.com/r/engineering_team"
  },
  [                                    # body (NestedPart)
    1,                                   # disposition = render
    "",                                  # language
    1,                                   # cardinality = single
    "text/markdown;variant=GFM-MIMI",    # contentType
                                         # content
    'Kudos to [@Alice Smith](mimi://example.com/u/alice-smith)' +
    ' 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="mimi://example.com/u/alice-smith">' +
  '@Alice Smith</a> for making 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. If the same message is placed more than once, the replaces data field refers to the message ID of the first instance of the message. This allows maximum correlation of different version of the same message, even if the receiver was not privy to all of the original versions.

Here Bob Jones corrects a typo in his original message:

```
# EDIT
# message ID = h'014028c0deddbdea56bec26172f6ede9
#                53d11024cb82b8192b5e2aea62d7fb47'
# timestamp  = 1644387248621 = 2022-02-08T22:14:08.621Z
[
  h'b8c2e6d8800ecf45df39be6c45f4c042', # salt
  h'015354973c2b65ca937bf1e035ae53a5   # replaces =
    ab80e947afa43d46920d4202e5cc0b27', #   Reply
  h'',                                 # topicId
  null,                                # expires
  h'017ce54837404c3696e0c747b985cb17   # inReplyTo =
    2716d0ed0a3d249ca63ace7d82a096f4', #   Original
  {                                    # extensions
    1: "mimi://example.com/u/bob-jones",
    2: "mimi://example.com/r/engineering_team"
  },
  [                                    # body (NestedPart)
    1,                                   # disposition = render
    "",                                  # language
    1,                                   # cardinality = single part
    "text/markdown;variant=GFM-MIMI",    # contentType
                                         # content
    '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. To delete a message which was previously edited, the replaces header field refers to the message ID of the first instance of the message to be 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.

```
# DELETE
# message ID = h'011d9efc78d04d4dcf4d82b07d5199bb
#                ef37011c1f0c7e004b6111c6dda504b4'
# timestamp  = 1644387248621 = 2022-02-08T22:14:08.621Z
[
  h'0a590d73b2c7761c39168be5ebf7f2e6', # salt
  h'015354973c2b65ca937bf1e035ae53a5   # replaces =
    ab80e947afa43d46920d4202e5cc0b27', #   Reply
  h'',                                 # topicId
  null,                                # expires
  h'017ce54837404c3696e0c747b985cb17   # inReplyTo =
    2716d0ed0a3d249ca63ace7d82a096f4', #   Original
  {                                    # extensions
    1: "mimi://example.com/u/bob-jones",
    2: "mimi://example.com/r/engineering_team"
  },
  [                                    # body (NestedPart)
    1,                                   # disposition = render
    "",                                  # language
    0,                                   # cardinality = null part
  ]
]
```

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.

```
# UNLIKE
# message ID = h'013aadbb8f313253c8930f4e93c6ca54
#                b2ed06d258185bdcec3870534c8a4ec4'
# timestamp  = 1644387250389 = 2022-02-08T22:14:10.389Z
[
  h'c5ba86dc9fd272e58ca52ec805b79199', # salt
  h'0158c4288911e50a8f6be3f47746b668   # replaces =
    2f10fd91bc8c05557aa589a3157aff68', #   Reaction
  h'',                                 # topicId
  null,                                # expires
  h'017ce54837404c3696e0c747b985cb17   # inReplyTo =
    2716d0ed0a3d249ca63ace7d82a096f4', #   Original
  {                                    # extensions
    1: "mimi://example.com/u/cathy-washington",
    2: "mimi://example.com/r/engineering_team"
  },
  [                                    # body (NestedPart)
    2,                                   # disposition = reaction
    "",                                  # language
    0                                    # cardinality = null part
  ]
]
```

5.8. Expiring

There are two types of expiring messages in instant messaging systems. In the typical implementation, messages are deleted a specific amount of time relative to (after) when the receiving client reads the message. We will refer to this as relative expiration.

Absolute 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. The goal instead is to allow cooperating client that respect the convention to signal expiration times clearly.

The expires data field contains the absolute timestamp when, or relative amount of time after reading, after which 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.

```
   # EXPIRING
   # message ID = h'01e59db8173939facc2c8a4a0f0ae8d0
   #                c7a11a81239626630c9464a8d6717a03'
   # timestamp  = 1644389403227 = 2022-02-08T22:49:06.227Z
   [
     h'33be993eb39f418f9295afc2ae160d2d', # salt
     null,                                # replaces
     h'',                                 # topicId
     [                                    # expires
       false,                               # absolute, not relative
       1644390004          # expires = 10 minutes after it was sent
     ],
     null,                                # inReplyTo
     {                                    # extensions
       1: "mimi://example.com/u/alice-smith",
       2: "mimi://example.com/r/engineering_team"
     },
     [                                    # body (NestedPart)
       1,                                   # disposition = render
       "",                                  # language
       1,                                   # cardinality = single part
       "text/markdown;variant=GFM-MIMI",    # contentType
                                            # content
       '__*VPN GOING DOWN*__ I\'m rebooting the VPN in ten minutes' +
       ' unless anyone objects.'
     ]
   ]
```

```
   87                                  # array(7)
      50                               # bytes(16)
         33be993eb39f418f9295afc2ae160d2d
      f6                               # primitive(22)
      40                               # bytes(0)
                                       # ""
      82                               # array(2)
         f4                            # primitive(20)
         1a 62036674                   # unsigned(1644390004)
      f6                               # primitive(22)
      a2                               # map(2)
         01                            # unsigned(1)
         78 20                         # text(32)
            6d696d693a2f2f6578616d706c652e63
            6f6d2f752f616c6963652d736d697468
            # "mimi://example.com/u/alice-smith"
         02                            # unsigned(2)
         78 25                         # text(37)
            6d696d693a2f2f6578616d706c652e63
            6f6d2f722f656e67696e656572696e67
            5f7465616d
            # "mimi://example.com/r/engineering_team"
      85                               # array(5)
         01                            # unsigned(1)
         60                            # text(0)
                                       # ""
         01                            # unsigned(1)
         78 1e                         # text(30)
            746578742f6d61726b646f776e3b7661
            7269616e743d47464d2d4d494d49
            # "text/markdown;variant=GFM-MIMI"
         58 50                         # bytes(80)
            5f5f2a56504e20474f494e4720444f57
            4e2a5f5f2049276d207265626f6f7469
            6e67207468652056504e20696e207465
            6e206d696e7574657320756e6c657373
            20616e796f6e65206f626a656374732e
            # "__*VPN GOING DOWN*__ I’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.

```
# ATTACHMENT
# message ID = h'0176180c7d19a925021fe446d241134d
#                05c38e0d999cdc0f39c391d2377ed9d1'
# timestamp  = not specified in example
[
  h'18fac6371e4e53f1aeaf8a013155c166', # salt
  null,                                # replaces
  h'',                                 # topicId
  null,                                # expires = never
  null,                                # inReplyTo
  {                                    # extensions
    1: "mimi://example.com/u/bob-jones",
    2: "mimi://example.com/r/engineering_team"
  },
  [                                    # body (NestedPart)
    6,                                   # disposition = attachment
    "en",                                # language
    2,                                   # cardinality = external
    "video/mp4",                         # contentType
    # url
    "https://example.com/storage/8ksB4bSrrRE.mp4",
    0,                                   # expires
    708234961,                           # size
    1,                                   # encAlg = AES-128-GCM
    h'21399320958a6f4c745dde670d95e0d8', # key
    h'c86cf2c33f21527d1dd76f5b',         # nonce
    h'',                                 # aad
    1,                                   # hashAlg = sha256
    h'9ab17a8cf0890baaae7ee016c7312fcc   # content hash
      080ba46498389458ee44f0276e783163',
    "2 hours of key signing video",      # description
    "bigfile.mp4"                        # filename
  ]
]

87                                     # array(7)
   50                                  # bytes(16)
      18fac6371e4e53f1aeaf8a013155c166
   f6                                  # primitive(22)
   40                                  # bytes(0)
                                       # ""
   f6                                  # primitive(22)
   f6                                  # primitive(22)
   a2                                  # map(2)
      01                               # unsigned(1)
      78 1e                            # text(30)
         6d696d693a2f2f6578616d706c652e63
         6f6d2f752f626f622d6a6f6e6573
```

```
         # "mimi://example.com/u/bob-jones"
      02                                # unsigned(2)
      78 25                             # text(37)
         6d696d693a2f2f6578616d706c652e63
         6f6d2f722f656e67696e656572696e67
         5f7465616d
         # "mimi://example.com/r/engineering_team"
   8f                                   # array(15)
      06                                # unsigned(6)
      62                                # text(2)
         656e                           # "en"
      02                                # unsigned(2)
      69                                # text(9)
         766964656f2f6d7034             # "video/mp4"
      78 2b                             # text(43)
         68747470733a2f2f6578616d706c652e
         636f6d2f73746f726167652f386b7342
         346253727252452e6d7034
         # "https://example.com/storage/8ksB4bSrrRE.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)
         9ab17a8cf0890baaae7ee016c7312fcc
         080ba46498389458ee44f0276e783163
      78 1c                             # text(28)
         3220686f757273206f66206b65792073
         69676e696e6720766964656f
         # "2 hours of key signing video"
      6b                                # text(11)
         62696766696c652e6d7034         # "bigfile.mp4"
```

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 in an ExternalPart with a disposition of 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.

```
# CONFERENCING
# message ID = h'01496d15a8dba28d7397f9868b70768e
#                4a67f765d5b5b1ae9e03848c5fdeb0ba'
# timestamp  = not specified in example
[
  h'678ac6cd54de049c3e9665cd212470fa', # salt
  null,                                # replaces
  'Foo 118',                           # topicId
  null,                                # expires
  null,                                # inReplyTo
  {                                    # extensions
    1: "mimi://example.com/u/alice-smith",
    2: "mimi://example.com/r/engineering_team"
  },
  [                                    # body (NestedPart)
    7,                                   # disposition = session
    "",                                  # language
    2,                                   # cardinality = external
    "",                                  # contentType
    "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
    ""                                   # filename
  ]
]

87                                      # array(7)
   50                                   # bytes(16)
      678ac6cd54de049c3e9665cd212470fa
   f6                                   # primitive(22)
   47                                   # bytes(7)
      466f6f20313138                    # "Foo 118"
```

```
      f6                                  # primitive(22)
      f6                                  # primitive(22)
      a2                                  # map(2)
         01                               # unsigned(1)
         78 20                            # text(32)
            6d696d693a2f2f6578616d706c652e63
            6f6d2f752f616c6963652d736d697468
            # "mimi://example.com/u/alice-smith"
         02                               # unsigned(2)
         78 25                            # text(37)
            6d696d693a2f2f6578616d706c652e63
            6f6d2f722f656e67696e656572696e67
            5f7465616d
            # "mimi://example.com/r/engineering_team"
      8f                                  # array(15)
         07                               # unsigned(7)
         60                               # text(0)
                                          # ""
         02                               # unsigned(2)
         60                               # text(0)
                                          # ""
         78 1e                            # text(30)
            68747470733a2f2f6578616d706c652e
            636f6d2f6a6f696e2f3132333435
            # "https://example.com/join/12345"
         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)
            4a6f696e2074686520466f6f20313138
            20636f6e666572656e6365
            # "Join the Foo 118 conference"
         60                               # text(0)
```

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.

6. CBOR Data Model and Encoding Restrictions

6.1. Encoding restrictions

MIMI content documents MUST be encoded using CBOR "deterministic" serialization as defined in Section 4.2.1 of [RFC8949].

In summary, this consists of:

* using the shortest CBOR encoding of integers, tags, and floats, and the shortest encoding of the length of byte strings, text strings, arrays, and maps.
* representing bigints as regular integers when they fit
* never using indefinite length encoding of byte strings, text strings, arrays, and maps
* sorting map keys according to the third bullet point of that section

Most CBOR libraries automatically use the shortest form and use definite length. Support for bigints is optional and is not required by any if the values built into the MIMI Content specification. Most implementations of CBOR which support bigints can automatically reduce to regular ints.

The only map inside the MIMI content format is the extensions map. Many popular implementations of CBOR do not yet support map sorting in 4.2.1, but preserve map order. Therefore it should be straightforward for MIMI content extensions to present their maps in the correct order. Implementations MUST NOT send MIMI content in RFC7094 "canonical" order.

6.2. Data model restrictions

* map keys for MIMI content extensions MUST be either integers, or text strings up to 255 octets long

* map keys for any maps at any level of nesting inside a MIMI content extension MUST only be integers, text strings, or byte strings.
* integer keys MUST be representable as a double float without loss of precision: uints 0..9007199254740991 (2^53 - 1), and negints (-(2^53) + 1) -9007199254740991..-1
* the only floating point NaN values permitted are the half-width quiet NaN (CBOR encoding 0xF97E00), or floating point values inside tags 80-87 (representing arrays of various fixed-sized IEEE floating point types)
* text strings used anywhere inside MIMI content MUST be valid UTF-8 sequences.
* use of floating point values inside MIMI content is strongly discouraged

6.3. Depth restrictions

NestedParts MUST NOT be nested more than 4-levels deep.

MIMI content extensions MUST NOT contain any combination of tags, maps, or arrays at more than 4 levels of depth. Specifically the MIMI content extensions map shown below has 4 levels of depth. The extensions map is at level 1, the array inside map key 256 is at level 2, the array inside that array is at level 3, the URI inside tag 32 is at level 4.

```
{
    /sender URI/ 1: "mimi://a.example/u/alice",
    256: [
        [h'1234', 32("http://example.com")],
        [h'3456', 5771]
    ]
}
```

7. Support for Specific Media Types

7.1. 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-MIMI -- Github Flavored Markdown for MIMI, defined in Section 7.1.1

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
* 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.

7.1.1. Specifics of Github Flavored Markdown in MIMI

A MIMI content client supports GitHub Flavored Markdown as defined in [GFM], with two changes: the Autolink extension is not supported; and instead of the Disallowed Raw HTML extension (tagfilter), the No HTML extension (nohtml), which is defined here, is MANDATORY. (For clarity, a fixed list of supported extensions, is further described in the bullet points below.)

To implement the No HTML extension to GFM, the opening angle bracket (<) of any HTML tag (as defined in Section 6.10 of [GFM]) is replaced with < before sending. Any HTML tag in a received message is rendered as plain text. Note that HTML tag includes open tags, closing tags, HTML comments, processing instructions, declarations, and CDATA sections.

The following GitHub Flavored Markdown extensions are supported. No other extensions are allowed:

* Tables
* Task list items
* Strikethrough
* No HTML

This document specifies what can be sent inside a MIMI content message; it does not restrict or prescribe in any way how input from a user is interpreted by an Instant Messaging client that support MIMI, before any message resulting from that input is sent.

Note that rendering Markdown as plain text is an acceptable form of "support".

7.2. 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.

8. IANA Considerations

RFC EDITOR: Please replace XXXX throughout with the RFC number assigned to this document.

8.1. 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

Author:
See the Authors' Addresses section of RFC XXXX.

Change controller:
Internet Engineering Task Force (iesg@ietf.org).

8.2. "More Instant Messaging Interoperability (MIMI)" Group Heading

8.2.1. MIMI Content Extension Keys registry

This document requests the creation of a new MIMI Content Extension Keys registry. The registry should be under the group heading of "More Instant Messaging Interoperability (MIMI)".

The MIMI Content format defined in this document, contains an extensions map in each message. The keys in the extensions map can be (positive or negative) integers, or text strings. Text strings and negative integer keys are reserved for private use. Positive integer keys are assigned in the registry under the Expert Review policy [RFC8126]. Integer keys between 1 and 255 are restricted to IETF Stream RFCs, as specified by the IETF Review registration procedure defined in [RFC8126].

The columns in the registry are as follows:

* Key: The extension map key positive integer assigned to the MIMI content extension.
* Name: a short descriptive name for the MIMI content extension.
* Type: The CBOR data type of the value corresponding to the key. Applications with multiple, related data items are encouraged to register a map type that contains all the related fields.
* Recommended: Whether support for this MIMI content extension is recommended by the IETF. Valid values are "Y", "N", and "D", as described below. The default value of the "Recommended" column is "N". Setting the Recommended item to "Y" or "D", or changing an item whose current value is "Y" or "D", requires Standards Action [RFC8126].
  - Y: Indicates that the IETF has consensus, and that the item is RECOMMENDED. This only means that the associated mechanism is fit for the purpose for which it was defined. Careful reading of the documentation for the mechanism is necessary to understand the applicability of that mechanism. The IETF could recommend mechanisms that have limited applicability, but will provide applicability statements that describe any limitations of the mechanism or necessary constraints on its use.
  - N: Indicates that the item has not been evaluated by the IETF and that the IETF has made no statement about the suitability of the associated mechanism. This does not necessarily mean that the mechanism is flawed, only that no consensus exists. The IETF might have consensus to leave an item marked as "N" on the basis of it having limited applicability or usage constraints.
  - D: Indicates that the item is discouraged and SHOULD NOT or MUST NOT be used. This marking could be used to identify mechanisms that might result in problems if they are used, such as a weak cryptographic algorithm or a mechanism that might cause interoperability problems in deployment.
* Reference: The document where this MIMI content extension is defined

Initial Contents:

Key	Name	Description	Type	R	Reference
0	(reserved)	N/A	-	-	RFCXXXX
1	senderUri	the sender as a MIMI participant	tstr	Y	RFCXXXX
2	roomUri	the MIMI room URI	tstr	Y	RFCXXXX

Table 1

8.2.2. MIMI Content Disposition registry

This document requests the creation of a new MIMI Disposition registry. The registry should be under the group heading of "More Instant Messaging Interoperability (MIMI)".

The MIMI Content format defined in this document, contains a disposition field for each message part, represented by an integer in the range of 0 to 255. These values are assigned in the registry under the Expert Review policy [RFC8126].

The columns in the registry are as follows:

* Name: a short descriptive name for the MIMI disposition.
* Value: The positive integer assigned to the MIMI disposition.
* Semantics: The semantics of the MIMI disposition.
* Reference: The document where this MIMI disposition is defined

The initial contents of the registry are below:

Name	Value	Semantics	Reference
unspecified	0	no disposition hint provided	RFCXXXX
render	1	render to the user according to local policy	RFCXXXX
reaction	2	a single reaction to another message	RFCXXXX
profile	3	representing a profile of a participant or room	RFCXXXX
inline	4	rendered "inline" directly in the chat interface	RFCXXXX
icon	5	an icon such as an avatar image	RFCXXXX
attachment	6	intended to be downloaded instead of rendered immediately	RFCXXXX
session	7	a description of a multimedia session, or URI to join one	RFCXXXX
preview	8	a sender-generated preview, ex: image or link contents	RFCXXXX

Table 2

8.2.3. Expert Review

Expert Review [RFC8126] registry requests are registered after a three-week review period on the MIMI Designated Expert (DE) mailing list mimi-reg-review@ietf.org (mailto:mimi-reg-review@ietf.org) on the advice of one or more of the MIMI DEs.

Registration requests sent to the MIMI DEs' mailing list for review SHOULD use an appropriate subject (e.g., "Request to register value in MIMI Content Extensions Keys registry").

Within the review period, the MIMI DEs will either approve or deny the registration request, communicating this decision to the MIMI DEs' mailing list and IANA. Denials SHOULD include an explanation and, if applicable, suggestions as to how to make the request successful. Registration requests that are undetermined for a period longer than 21 days can be brought to the IESG's attention for resolution using the iesg@ietf.org (mailto:iesg@ietf.org) mailing list.

Criteria that SHOULD be applied by the MIMI DEs includes determining whether the proposed registration duplicates existing functionality, whether it is likely to be of general applicability or useful only for a single application, and whether the registration description is clear.

IANA MUST only accept registry updates from the MIMI DEs and SHOULD direct all requests for registration to the MIMI DEs' mailing list.

In cases where a registration decision could be perceived as creating a conflict of interest for a particular MIMI DE, that MIMI DE SHOULD defer to the judgment of the other MIMI DEs.

8.3. GFM-MIMI Markdown variant

This document registers a new Markdown variant in the IANA Markdown Variants registry. The registration template below conforms with [RFC7763].

Identifier: GFM-MIMI

Name: GitHub Flavored Markdown Subset for MIMI

Description:
GitHub Flavored Markdown, without Autolinks and with no embedded HTML

References: RFCXXXX

Contact Information:
IETF MIMI Working Group <mimi@ietf.org>

9. Security Considerations

9.1. 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
  - where the first octet of the message ID is an unknown hash algorithm.
* 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
* topicId
  - the topicId is very long (greater than 4096 octets)
* expires
  - refers to a date more than a year in the past (only possible with absolute expiration)
  - refers to a date more than a year in the future (possible with both relative and absolute expiration)
* 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

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 or replaces refer to an unknown message. Such a message could have been sent before the local client joined.
* 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.

9.2. Generating the random salt

To ensure a strong source of entropy for the per-message unique salt required in each message, the client can export a secret from the MLS key schedule, for example with the label salt_base_secret and calculate the salt as the first 16-octets of the HMAC of a locally generated nonce and the franking_base_secret.

```
hash_output = HMAC_SHA256( salt_base_secret, nonce )
salt = hash_output[0..15]
```

9.3. Rendering and authorization of edits and deletes

This content format allows clients to send new versions of previously sent messages, effectively replacing ("editing") or retracting ("deleting") another referenced message. The rendering of these "edits" and "deletes" are important from a security perspective.

For example, if Alice writes "Bob, could I borow a pen?", and Bob reacts with a thumbs up emoji, Alice might edit this message with no change in meaning (correcting the spelling of "borrow"), or a dramatic change in meaning ("Bob, could I borrow your Ferrari this month?"). The receiver SHOULD indicate clearly that a received message has been edited or retracted. The receiver might:

* offer an option to view previous versions of a message,
* show a summarized or thumbnail version of a message referenced in a reply,
* indicate clearly that a reply was to a previous version of a message than the most recent one.
* show a detailed view of reaction indicating that some reactions referred to a previous version of the message.

In addition, some groups may have special policies or permissions allowing specific types of edits or deletes. For example, a moderator in one room might be allowed to edit the topic of a message, but not modify the rest of the content. An administrator might be allowed to delete messages which violate the policy of the group. Receiving clients MUST NOT allow parties other than the original sender of a message to edit or delete that message, unless there is a specific, concrete authorization policy which allows it. Likewise, even the original sender of a message MUST NOT be able to change the semantics of any other portion of the message except for the contents of the NestedPart, without specific authorization.

9.4. 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. Note that the optional franking mechanism discussed in Section 5.4.1.2 of [I-D.ietf-mimi-protocol] prevents follower servers from modifying the timestamp.

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

9.5. 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.

9.6. 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 example, 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.

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

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.

10. References

10.1. Normative References

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

[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>.

[RFC2392] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, DOI 10.17487/RFC2392, August 1998, <https://www.rfc-editor.org/info/rfc2392>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

10.2. 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)", Work in Progress, Internet-Draft, draft-ietf-cbor-edn-literals-19, 16 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-edn-literals-19>.

[I-D.ietf-mimi-arch]
Barnes, R., "An Architecture for More Instant Messaging Interoperability (MIMI)", Work in Progress, Internet-Draft, draft-ietf-mimi-arch-02, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-mimi-arch-02>.

[I-D.ietf-mimi-protocol]
Barnes, R., Hodgson, M., Kohbrok, K., Mahy, R., Ralston, T., and R. Robert, "More Instant Messaging Interoperability (MIMI) using HTTPS and MLS", Work in Progress, Internet-Draft, draft-ietf-mimi-protocol-05, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-mimi-protocol-05>.

[I-D.ietf-mls-extensions]
Robert, R., "The Messaging Layer Security (MLS) Extensions", Work in Progress, Internet-Draft, draft-ietf-mls-extensions-08, 21 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-mls-extensions-08>.

[I-D.mahy-mimi-identity]
Mahy, R., "More Instant Messaging Interoperability (MIMI) Identity Concepts", Work in Progress, Internet-Draft, draft-mahy-mimi-identity-04, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-mahy-mimi-identity-04>.

[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>.

[RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <https://www.rfc-editor.org/info/rfc5545>.

[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>.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <https://www.rfc-editor.org/info/rfc6350>.

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

[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 = [
  salt: bstr .size 16,
  replaces: null / MessageId,
  topicId: bstr,
  expires: null / Expiration,
  inReplyTo: null / MessageId,
  mimiExtensions: extensions,
  nestedPart: NestedPart
```

```
   ]

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

   NullPart = ( cardinality: nullpart )

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

   ExternalPart = (
       cardinality: external,
       contentType: tstr,
       url: tstr,
       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,
       filename: tstr
   )

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

   Expiration = [
       relative: bool,
       time: uint .size 4
   ]

   baseDispos = &(
       unspecified: 0,
       render: 1,
       reaction: 2,
       profile: 3,
```

```
      inline: 4,
      icon: 5,
      attachment: 6,
      session: 7,
      preview: 8
  )
  ; Note: any ext_dispos take precedence
  unknownDispos = &( unknown: 9..255 )

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

  extensions = {
    ? &(senderUri: 1) ^ => tstr,
    ? &(roomUri: 2) ^ => tstr,
    * $$otherKnownExtensions,
    * unknownExtension
  }

  unknownExtension = (
     name => value
  )
  name = int / tstr .size (1..255)
  value = any

  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,
    hubAcceptedTimestamp: Timestamp,
    mlsGroupId: bstr,
    senderLeafIndex: uint .size 4,
    senderClientUrl: MsgUri,
    senderUserUrl: MsgUri,
    roomUrl: MsgUri
]

MsgUri = tstr
MessageId = bstr .size 32
Timestamp = MsecsSinceEpoch / ExtendedTime
; milliseconds since start of UNIX epoch
MsecsSinceEpoch = uint .size 8
; extended time from RFC9581
ExtendedTime = #6.1001({* name => value })
name = int / tstr .size (1..255)
value = any .size (0..4095)
```

Appendix B. Multipart examples

In a heterogeneous 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.

```
   # MULTIPART-1
   # message ID = h'01da5a515ec5db42cc4dcc19b90c3c31
   #               245d8a1cfcce11318f24eb11dce0990e'
   # timestamp  = not specified in example
   [
     h'261c953e178af653fe3d42641b91d814', # salt
     null,                                # replaces
     h'',                                 # topicId
     null,                                # expires = never
     null,                                # inReplyTo
     {                                    # extensions
       1: "mimi://example.com/u/alice-smith",
       2: "mimi://example.com/r/engineering_team"
     },
     [                                    # body (NestedPart)
                                            # partIndex = 0 (1st part)
       1,                                   # disposition = render
       "",                                  # language
       3,                                   # cardinality = multi
       0,                                   # partSemantics = chooseOne
       [
         [                                  # partIndex = 1
           1,                               # disposition = render
           "",                              # language
           1,                               # cardinality = single
           "text/markdown;variant=GFM-MIMI",# contentType
           '# Welcome!'                     # content
         ],
         [                                  # partIndex = 2
           1,                               # disposition = render
           "",                              # language
           1,                               # cardinality = single
           # contentType
           "application/vnd.examplevendor-fancy-im-message",
           # content
           h'dc861ebaa718fd7c3ca159f71a2001'
         ]
       ]
     ]
   ]
```

B.2. Multiple Reactions Example

This example shows sending a reaction with multiple separate emojis.

```
   # MULTIPART-2
   # message ID = h'01d65918c6c51c8e76546337276ae6f4
   #               bfd873d867d5cb57c76bcdca3d999dd7'
   # timestamp  = not specified in example
   [
     h'8528dc2d92e4f1944d62042907ab94d0', # salt
     null,                                # replaces
     h'',                                 # topicId
     null,                                # expires = never
     null,                                # inReplyTo
     {                                    # extensions
       1: "mimi://example.com/u/alice-smith",
       2: "mimi://example.com/r/engineering_team"
     },
     [                                    # body (NestedPart)
                                            # partIndex = 0 (1st part)
       2,                                   # disposition = reaction
       "",                                  # language
       3,                                   # cardinality = multi
       2,                                   # partSemantics = processAll
       [
         [                                  # partIndex = 1
           2,                               # disposition = reaction
           "",                              # language
           1,                               # cardinality = single
           "text/plain;charset=utf-8",      # contentType
           h'E2 9D A4'           # content = Unicode "heart"
         ],
         [                                  # partIndex = 2
           2,                               # disposition = reaction
           "",                              # language
           1,                               # cardinality = single
           "text/plain;charset=utf-8",      # contentType
           h'F0 9F A5 B3'        # content = Unicode "party face"
         ],
         [                                  # partIndex = 3
           2,                               # disposition = reaction
           "",                              # language
           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
```

```
# MULTIPART-3
# message ID = h'01cfebeadbdb83c1eefb6403ba4852da
#                f8bbbf9cd53bf5035a74d5d741950c9f'
# timestamp  = not specified in example
[
  h'b8362793168d18c049b882d4642a2274', # salt
  null,                                # replaces
  h'',                                 # topicId
  null,                                # expires = never
  null,                                # inReplyTo
  {                                    # extensions
    1: "mimi://example.com/u/alice-smith",
    2: "mimi://example.com/r/engineering_team"
  },
  [                                    # body (NestedPart)
                                         # partIndex = 0 (1st part)
    1,                                   # disposition = render
    "",                                  # language
    3,                                   # cardinality = multi
    0,                                   # partSemantics = chooseOne
    [
      [                                  # partIndex = 1
        1,                               # disposition = render
        "",                              # language
        3,                               # cardinality = multi
        2,                               # partSemantics = processAll
        [
          [                              # partIndex = 2
            1,                           # disposition = render
```

```
            "",                           # language
            3,                            # cardinality = multi
            0,                            # partSemantics = chooseOne
            [
              [                           # partIndex = 3
                1,                        # disposition = render
                "en",                     # language
                1,                        # cardinality = single
                # content type
                "text/html;charset=utf-8",
                # content
                '<html><body><h1>Welcome!</h1>\n' +
                '<img src="cid:5@local.invalid" ' +
                'alt="Welcome image"/>\n</body></html>'
              ],       # English HTML
              [                           # partIndex = 4
                1,                        # disposition = render
                "fr",                     # language
                1,                        # cardinality = single
                # content type
                "text/html;charset=utf-8",
                # content
                '<html><body><h1>Bienvenue!</h1>\n' +
                '<img src="cid:5@local.invalid" ' +
                'alt="Image bienvenue"/>\n</body></html>'
              ]        # French HTML
            ]
          ],         # English or French HTML (refers to GIF)
          [                               # partIndex = 5
            4,                            # disposition = inline
            "",                           # language
            1,                            # cardinality = single
            "image/gif",                  # content type
            # content
            h'dc861ebaa718fd7c3ca159f71a2001a7'
          ]          # GIF
        ]
      ],           # GIF version with English or French HTML
      [                                   # partIndex = 6
        1,                                # disposition = render
        "",                               # language
        3,                                # cardinality = multi
        2,                                # partSemantics = processAll
        [
          [                               # partIndex = 7
            1,                            # disposition = render
            "",                           # language
            3,                            # cardinality = multi
```

```
              0                           # partSemantics = chooseOne
              [
                [                         # partIndex = 8
                  1,                      # disposition = render
                  "en",                   # language
                  1,                      # cardinality = single
                  # content type
                  "text/html;charset=utf-8",
                  # content
                  '<html><body><h1>Welcome!</h1>\n' +
                  '<img src="cid:10@local.invalid" ' +
                  'alt="Welcome image"/>\n</body></html>'
                ],       # English HTML
                [                         # partIndex = 9
                  1,                      # disposition = render
                  "fr",                   # language
                  1,                      # cardinality = single
                  # content type
                  "text/html;charset=utf-8",
                  # content
                  '<html><body><h1>Bienvenue!</h1>\n' +
                  '<img src="cid:10@local.invalid" ' +
                  'alt="Image bienvenue"/>\n</body></html>'
                ]        # French HTML
              ]
            ],         # English or French HTML (refers to PNG)
            [                             # partIndex = 10
              4,                          # disposition = inline
              "",                         # language
              1,                          # cardinality = single
              "image/png",                # content type
              # content
              h'fa444237451a05a72bb0f67037cc1669'
            ]          # PNG
          ]
        ]            # PNG version 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

C.7. Changes between draft-ietf-mimi-content-04 and draft-ietf-mimi-content-05

* change message ID construction, and add salt
* remove partIndex and make it implied
* mention Content ID URI (cid:) and describe using it with the implicit partIndex
* discuss rendering and authorization issues for edit/delete in the security considerations
* include both absolute and relative expiration times
* add specificity about markdown support / create GFM-MIMI Markdown variant
* remove tag from URLs in ExternalPart and implied headers

C.8. Changes between draft-ietf-mimi-content-05 and draft-ietf-mimi-content-06

* remove lastSeen field
* improve guidance on processing MultiPart with content ID URIs
* add extra copy of the salt to Message ID construction to prevent SHA256 length extension attack
* future-proof timestamp format hub timestamp
* IANA register the MIMI extension keys used in the draft
* lots of changes for internal consistency
* rebuild all the examples from a script. make sure the EDN and CBOR correspond and the CDDL validates the CBOR.

C.9. Changes between draft-ietf-mimi-content-06 and draft-ietf-mimi-content-07

* fixed a bug in the generation of message IDs; regenerated them correctly.
* moved MIMI message status format into draft-mahy-mimi-message-status
* clarified that the salt is always 16 octets
* make examples uses the correct CBOR map key for room

C.10. Changes between draft-ietf-mimi-content-07 and draft-ietf-mimi-content-08

* add an explicit length before the sender URI and room URI in the input to the message ID hash input (Issue #61/PR#70)

* make the inline CDDL examples consistent with the complete CDDL (PR#69)
* add IANA registry for disposition values (Issue #64/PR#71)
* add CBOR encoding, data model, and depth restrictions section (Issue #76/PR#79)
* removed the maximum length of MIMI content extensions

Author's Address

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