Network Working Group
Request for Comments: 5438
Category: Standards Track

E. Burger
Unaffiliated
H. Khartabil
Ericsson Australia

February 2009

Instant Message Disposition Notification (IMDN)

Status of This Memo

Copyright Notice

Abstract

Instant Messaging (IM) refers to the transfer of messages between users in real-time. This document provides a mechanism whereby endpoints can request Instant Message Disposition Notifications (IMDN), including delivery, processing, and display notifications, for page-mode instant messages.

The Common Presence and Instant Messaging (CPIM) data format specified in RFC 3862 is extended with new header fields that enable endpoints to request IMDNs. A new message format is also defined to convey IMDNs.

This document also describes how SIP entities behave using this extension.

Table of Contents

1. Introduction

In many user-to-user message exchange systems, message senders often wish to know if the human recipient actually received a message or has the message displayed.

Electronic mail [RFC5321] offers a solution to this need with Message Disposition Notifications [RFC3798]. After the recipient views the message, her mail user agent generates a Message Disposition Notification, or MDN. The MDN is an email that follows the format prescribed by RFC 3798 [RFC3798]. The fixed format ensures that an automaton can process the message.

The Common Presence and Instant Messaging (CPIM) format, Message/CPIM [RFC3862], is a message format used to generate instant messages. The Session Initiation Protocol (SIP [RFC3261]) can carry instant messages generated using message/CPIM in SIP MESSAGE requests [RFC3428].

This document extends the Message/CPIM message format in much the same way Message Disposition Notifications extends electronic mail. This extension enables Instant Message Senders to request, create, and send Instant Message Disposition Notifications (IMDN). This mechanism works for page-mode as well as session-mode instant messages. This document only discusses page-mode. Session-mode is left for future standardisation efforts.

This specification defines three categories of disposition types: "delivery", "processing", and "display". Specific disposition types provide more detailed information. For example, the "delivery" category includes "delivered" to indicate successful delivery and "failed" to indicate failure in delivery.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This document refers generically to the sender of a message in the masculine (he/him/his) and the recipient of the message in the feminine (she/her/hers). This convention is purely for convenience and makes no assumption about the gender of a message sender or recipient.

3. Terminology

- IM: An Instant Message generated using the Message/CPIM format.
- IMDN: An Instant Message Disposition Notification generated using the Message/CPIM format that carries an IMDN XML document.
- Message: An IM or an IMDN generated using the Message/CPIM format.
- IM Sender: An endpoint (user agent) generating and sending an IM. Also, the endpoint request IMDNs for an IM. Quite often, the IM Sender is the IMDN Recipient. However, that is not always the case, since the IMDN uses the From header in the CPIM message. That value is often the IM Sender's Address of Record (AOR). This address may in fact resolve to different user agents.
- IM Recipient: An endpoint (user agent) that receives IMs. The IM Recipient, as the node that presumably renders the IM to the user, generates and sends delivery IMDNs to IMs, if requested by the IM Sender and allowed by the IM Recipient.
- Endpoint: An IM Sender or an IM Recipient.
- Intermediary: An entity in the network, most often an application server (including URI-List and store-and-forward servers), that forwards an IM to its final destination. Intermediaries also can generate and send processing IMDNs to IMs, if requested by the IM Sender and allowed by policy.

- Gateway: An intermediary that translates between different IM systems that use different protocols.

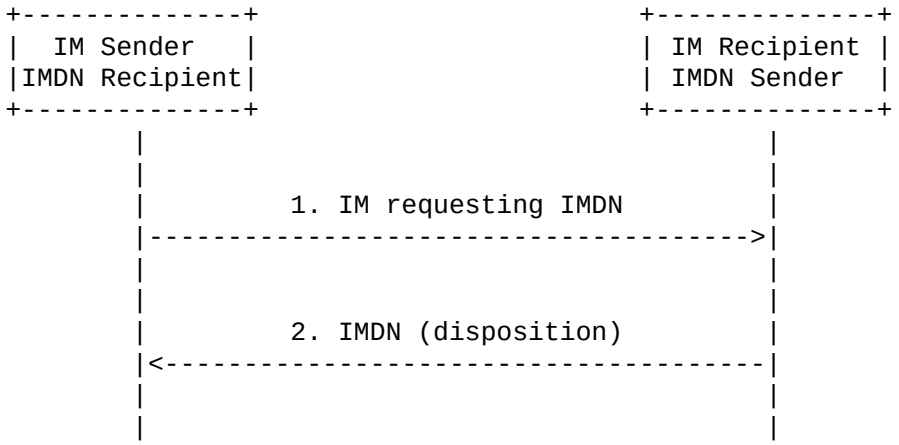
- IMDN payload: An XML document carrying the disposition notification information. In this specification, it is of MIME type "message/imdn+xml".

- Disposition type: This specification defines three categories of disposition types: "delivery", "processing", and "display".

- Transport Protocol Message: A SIP or other protocol message that contains an IM or IMDN.

4. Overview

The diagram below shows the basic protocol flow. An IM Sender creates an IM, adds IMDN request information that the IM Sender is interested in receiving, and then sends the IM. At a certain point in time, the IM Recipient or an intermediary determines that the user or application has received, did not receive, displayed, or otherwise disposed of the IM. The mechanism by which an IM Recipient determines its user has read an IM is beyond the scope of this document. At that point, the IM Recipient or intermediary automatically generates a notification message to the IM Sender. This notification message is the Instant Message Disposition Notification (IMDN).



Basic IMDN Message Flow

Note the recipient of an IMDN, in some instances, may not be the IM Sender. This is specifically true for page-mode IMs where the Address of Record (AOR) of the IM Sender, which is present in the IM, resolves to a different location or user agent than that from which

the IM originated. This could happen, for example, if resolving the AOR results in forking the request to multiple user agents. For simplicity, the rest of this document assumes that the IM Sender and the IMDN Recipient are the same and therefore will refer to both as the IM Sender.

5. Disposition Types

There are three broad categories of disposition states. They are delivery, processing, and display.

5.1. Delivery

The delivery notification type indicates whether or not the IM has been delivered to the IM Recipient. The delivery notification type can have the following states:

- "delivered" to indicate successful delivery.
- "failed" to indicate failure in delivery.
- "forbidden" to indicate denial for the IM Sender to receive the requested IMDN. The IM Recipient can send the "forbidden" state, but usually it is an intermediary that sends the message, if one configures it to do so. For example, it is possible the administrator has disallowed IMDNs.
- "error" to indicate an error in determining the fate of an IM.

5.2. Processing

The processing notification type indicates that an intermediary has processed an IM. The processing notification type can have the following states:

- "processed" to indicate that the intermediary has performed its task on the IM. This is a general state of the IM.
- "stored" to indicate that the intermediary stored the IM for later delivery.
- "forbidden" to indicate denial for the IM Sender to receive the requested IMDN. The "forbidden" state is sent by an intermediary that is configured to do so. For example, the administrator has disallowed IMDNs.
- "error" to indicate an error in determining the fate of an IM.

5.3. Display

The display notification type indicates whether or not the IM Recipient rendered the IM to the user. The display notification type can have the following states:

- "displayed" to indicate that the IM has been rendered to the user.
- "forbidden" to indicate denial, by the IM Recipient, for the IM Sender to receive the requested IMDN.
- "error" to indicate an error in determining the fate of an IM.

In addition to text, some IMs may contain audio, video, and still images. Therefore, the state "displayed" includes the start of rendering the audio or video file to the user.

Since there is no positive acknowledgement from the user, one cannot determine if the user actually read the IM. Thus, one cannot use the protocol described here as a service to prove someone actually read the IM.

6. New CPIM Header Fields

This specification extends the CPIM data format specified in RFC 3862 [RFC3862]. A new namespace is created as well as a number of new CPIM header fields.

6.1. CPIM Header Field Namespace

Per CPIM [RFC3862], this specification defines a new namespace for the CPIM extension header fields defined in the following sections. The namespace is:

urn:ietf:params:imdn

As per CPIM [RFC3862] requirements, the new header fields defined in the following sections are prepended, in CPIM messages, by a prefix assigned to the URN through the NS header field of the CPIM message. The remainder of this specification always assumes an NS header field like this one:

NS: imdn <urn:ietf:params:imdn>.

Of course, clients are free to use any prefix while servers and intermediaries must accept any legal namespace prefix specification.

6.2. Disposition-Notification

The IM Sender MUST include the Disposition-Notification header field to indicate the desire to receive IMDNs from the IM Recipient for that specific IM. Section 10 defines the syntax.

6.3. Message-ID

The IM Sender MUST include the Message-ID header field in the IM for which he wishes to receive an IMDN. The Message-ID contains a globally unique message identifier that the IM Sender can use to correlate received IMDNs. Because the Message-ID is used by the sender to correlate IMDNs with their respective IMs, the Message-ID MUST be selected so that:

- There is a minimal chance of any two Message-IDs accidentally colliding during the time period within which an IMDN might be received.

- It is prohibitive for an attacker who has seen one or more valid Message-IDs to generate additional valid Message-IDs.

The first requirement is a correctness requirement to ensure correct matching by the sender. The second requirement prevents off-path attackers from forging IMDNs. In order to meet both of these requirements, it is RECOMMENDED that Message-IDs be generated using a cryptographically secure, pseudo-random number generator and contain at least 64 bits of randomness, thus reducing the chance of a successful guessing attack to n/2^64, where n is the number of outstanding valid messages.

When the IM Sender receives an IMDN, it can compare its value with the value of the <message-id> element present in the IMDN payload. IMDNs also carry this header field. Note that since the IMDN is itself an IM, the Message-ID of the IMDN will be different than the Message-ID of the original IM. Section 10 defines the syntax.

6.4. Original-To

An intermediary MAY insert an Original-To header field into the IM. The value of the Original-To field MUST be the address of the IM Receiver. The IM Recipient uses this header to indicate the original IM address in the IMDNs. The IM Recipient does this by populating the <original-recipient-uri> element in the IMDN. The intermediary MUST insert this header if the intermediary changes the CPIM To header field value. The header field MUST NOT appear more than once in an IM. The intermediary MUST NOT change this header field value if it is already present. Section 10 defines the syntax.

6.5. IMDN-Record-Route

An intermediary MAY insert an IMDN-Record-Route header field to the IM. This enables the intermediary to receive and process the IMDN on its way back to the IM Sender. The value of the IMDN-Record-Route header field MUST be the address of the intermediary. Multiple IMDN-Record-Route header fields can appear in an IM. Section 10 defines the syntax.

6.6. IMDN-Route

The IMDN-Route header field provides routing information by including one or more addresses to which to route the IMDN. An intermediary that needs the IMDN to flow back through the same intermediary MUST add the IMDN-Record-Route header. When the IM Recipient creates the corresponding IMDN, the IM Recipient copies the IMDN-Record-Route headers into corresponding IMDN-Route header fields. Section 10 defines the syntax.

7. Endpoint Behaviour

7.1. IM Sender

7.1.1. Constructing Instant Messages

An IM is constructed using the CPIM message format defined in RFC 3862 [RFC3862].

7.1.1.1. Adding a Message-ID Header Field

If the IM Sender requests the reception of IMDNs, the IM Sender MUST include a Message-ID header field. This header field enables the IM Sender to match any IMDNs with their corresponding IMs. See Section 6.3 for Message-ID uniqueness requirements.

7.1.1.2. Adding a DateTime Header Field

Some devices are not able to retain state over long periods. For example, mobile devices may have memory or battery limits. Such limits mean these devices may not be able to, or may choose not to, keep sent messages for the purposes of correlating IMDNs with sent IMs. To make some use of IMDN in this case, we add a time stamp to the IM to indicate when the user sent the message. The IMDN returns this time stamp to enable the user to correlate the IM with the IMDN at the human level. We use the DateTime CPIM header field for this purpose. Thus, if the IM Sender would like an IMDN, the IM Sender MUST include the DateTime CPIM header field.

7.1.1.3. Adding a Disposition-Notification Header Field

The Disposition-Notification conveys the type of disposition notification requested by the IM Sender. There are three types of disposition notification: delivery, processing, and display. The delivery notification is further subdivided into failure and success delivery notifications. An IM Sender requests failure delivery notification by including a Disposition-Notification header field with value "negative-delivery". Similarly, a success notification is requested by including a Disposition-Notification header field with value "positive-delivery". The IM Sender can request both types of delivery notifications for the same IM.

The IM Sender can request a processing notification by including a Disposition-Notification header field with value "processing".

The IM Sender can also request a display notification. The IM Sender MUST include a Disposition-Notification header field with the value "display" to request a display IMDN.

The absence of this header field or the presence of the header field with an empty value indicates that the IM Sender is not requesting any IMDNs. Disposition-Notification header field values are comma-separated. The IM Sender MAY request more than one type of IMDN for a single IM.

Future extensions may define other disposition notifications not defined in this document.

Section 10 describes the formal syntax for the Disposition-Notification header field. The following is an example CPIM body of an IM where the IM Sender requests positive and negative delivery notifications, but not display notification or processing notifications:

```
From: Alice <im:alice@example.com>
To: Bob <im:bob@example.com>
NS: imdn <urn:ietf:params:imdn>
imdn.Message-ID: 34jk324j
DateTime: 2006-04-04T12:16:49-05:00
imdn.Disposition-Notification: positive-delivery, negative-delivery
Content-type: text/plain
Content-length: 12

Hello World
```

7.1.2. Matching IMs with IMDNs

An IM Sender matches an IMDN to an IM by matching the Message-ID header field value in the IM with the <message-id> element value in the body of the IMDN. If the IM was delivered to multiple recipients, the IM Sender uses the <recipient-uri> element and the <original-recipient-uri> element in the XML body of the IMDN it received to determine if the IM was sent to multiple recipients and to identify the IM Recipient that sent the IMDN.

An IM Sender can determine an IMDN is a disposition notification by noting if the Content-Disposition in the IMDN is "notification". This does mean the IM Sender MUST understand the Content-Disposition MIME header in CPIM messages.

7.1.3. Keeping State

This specification does not mandate the IM Sender to keep state for a sent IM.

Once an IM Sender sends an IM containing an IMDN request, it MAY preserve the IM context (principally the Message-ID), other user-identifiable information such as the IM subject or content, and the date and time it was sent. Without preservation of the IM context, the IM Sender will not be able to correlate the IMDN with the IM it sent. The IM Sender may find it impossible to preserve IM state if it has limited resources or does not have non-volatile memory and then loses power.

There is, however, the concept of a "Sent Items" box in an application that stores sent IMs. This "Sent Items" box has the necessary information and may have a fancy user interface indicating the state of a sent IM. A unique Message-ID for this purpose proves to be useful. The length of time for items to remain in the "Sent Items" box is a user choice. The user is usually free to keep or delete items from the "Sent Items" box as she pleases or as the memory on the device reaches capacity.

Clearly, if an IM Sender loses its sent items state (for example, the user deletes items from the "Sent Items" box), the client may use a different display strategy in response to apparently unsolicited IMDNs.

This specification also does not mandate an IM Sender to run any timers waiting for an IMDN. There are no time limits regarding when IMDNs may be received.

IMDNs may legitimately never be received, so the time between the sending of an IM and the generation and ultimate receipt of the IMDN may simply take a very long time. Some clients may choose to purge the state associated with the sent IM. This is the reason for adding the time stamp in the IM and having it returned in the IMDN. This gives the user some opportunity to remember what IM was sent. For example, if the IMDN indicates that the IM the user sent at 2 p.m. last Thursday was delivered, the user has a chance to remember that they sent an IM at 2 p.m. last Thursday.

7.1.4. Aggregation of IMDNs

An IM Sender may send an IM to multiple recipients in one Transport Protocol Message (typically using a URI-List server [RFC5365]) and request IMDNs. An IM Sender that requested IMDNs MUST be prepared to receive multiple aggregated or non-aggregated IMDNs. See Section 8.3 for details.

7.2. IM Recipient

7.2.1. Constructing IMDNs

IM Recipients examine the contents of the Disposition-Notification header field of the CPIM message to determine if the recipient needs to generate an IMDN for that IM. Disposition-Notification header fields of CPIM messages can include one or more values. IM Recipients may need to generate zero, one, or more IMDNs for that IM, for example, a delivery notification as well as a display notification. In this case, the IM Recipient MUST be able to construct multiple IMDNs per IM. An IM Recipient MUST NOT construct more than one IMDN per disposition type. That is, it must not generate a delivery notification indicating "delivered" followed by a delivery notification indicating "failed" for the same IM. If the IM Sender requested only failure notifications and the IM was successfully delivered, then no IMDNs will be generated. If the IM Recipient does not understand a value of the Disposition-Notification header field, the IM Recipient ignores that value.

The IM Recipient MUST NOT generate "processing" notifications.

A Disposition-Notification header field MUST NOT appear in an IMDN since it is forbidden to request an IMDN for an IMDN. An IM Sender MUST ignore a delivery notification request in an IMDN if present. The IM Sender MUST NOT send an IMDN for an IMDN.

An IMDN MUST contain a Message-ID header field. The same rules of uniqueness for the Message-ID header field that appears in an IM apply to an IMDN. The Message-ID header field in the IMDN is different and unrelated to the one in the IM.

An IM may contain an IMDN-Record-Route header field (see Section 8 for details). If IMDN-Record-Route header fields appear in the IM, the IM Recipient constructing the IMDN MUST copy the contents of the IMDN-Record-Route header fields into IMDN-Route header fields in the IMDN and maintain their order. The IMDN is then sent to the URI in the top IMDN-Route header field. IMDN-Record-Route header fields do not make sense in an IMDN and therefore MUST NOT be placed in an IMDN. IMDN Recipients MUST ignore it if present.

If there is no IMDN-Record-Route header field, the IM Recipient MUST send the IMDN to the URI in the From header field.

As stated in CPIM [RFC3862], CPIM messages may need to support MIME headers other than Content-type. IM Recipients MUST insert a Content-Disposition header field set to the value "notification". This indicates to the IM Sender that the message is an IMDN to an IM it has earlier sent.

7.2.1.1. Constructing Delivery Notifications

The IM Recipient constructs a delivery notification in a similar fashion as an IM, using a CPIM body [RFC3862] that carries a Disposition Notification XML document formatted according to the rules specified in Section 11. The MIME type of the Disposition Notification XML document is "message/imdn+xml".

Section 10 defines the schema for an IMDN.

The following is an example CPIM body of an IMDN:

```
From: Bob <im:bob@example.com>
To: Alice <im:alice@example.com>
NS: imdn <urn:ietf:params:imdn>
imdn.Message-ID: d834jied93rf
Content-type: message/imdn+xml
Content-Disposition: notification
Content-length: ...

<?xml version="1.0" encoding="UTF-8"?>
<imdn xmlns="urn:ietf:params:xml:ns:imdn">
      <message-id>34jk324j</message-id>
      <datetime>2008-04-04T12:16:49-05:00</datetime>
     <recipient-uri>im:bob@example.com</recipient-uri>
```

```
        <original-recipient-uri
          >im:bob@example.com</original-recipient-uri>
        <delivery-notification>
           <status>
              <delivered/>
           </status>
        </delivery-notification>
      </imdn>
```

7.2.1.2. Constructing Display Notifications

The IM Recipient constructs a display notification in a similar fashion as the delivery notification. See Section 7.2.1.1 for details.

Section 10 defines the schema for an IMDN.

The following is an example:

```
From: Bob <im:bob@example.com>
To: Alice <im:alice@example.com>
NS: imdn <urn:ietf:params:imdn>
imdn.Message-ID: dfjkleriou432333
Content-type: message/imdn+xml
Content-Disposition: notification
Content-length: ...

<?xml version="1.0" encoding="UTF-8"?>
<imdn xmlns="urn:ietf:params:xml:ns:imdn">
      <message-id>34jk324j</message-id>
      <datetime>2008-04-04T12:16:49-05:00</datetime>
     <recipient-uri>im:bob@example.com</recipient-uri>
      <original-recipient-uri
         >im:bob@example.com</original-recipient-uri>
      <display-notification>
         <status>
            <displayed/>
         </status>
      </display-notification>
    </imdn>
```

There are situations where the IM Recipient cannot determine if or when the IM has been displayed. The IM Recipient in this case generates a display notification with a <status> value of "error" to indicate an internal error by the server. Note that the IM Recipient may choose to ignore any IMDN requests and not send any IMDNs. An IM

Recipient may not wish to let a sender know whether or not a particular message has been displayed to her. This could be a per-message, per-sender, or programmed policy choice.

8. Intermediary Behaviour

In this context, intermediaries are application servers (including URI-List and store-and-forward servers) and gateways. A gateway is a server that translates between different IM systems that use different protocols.

A URI-List server may change the IM Recipient address from its own to the address of the final recipient of that IM for every copy it makes that it sends to the list members (see [RFC5365] for details). In this case, if the IM Sender is requesting an IMDN, the intermediary SHOULD add an Original-To header field to the IM, populating it with the address that was in the CPIM To header field before it was changed. That is, the intermediary populates the Original-To header field with the intermediary address. Of course, one may configure an intermediary to restrict it from rewriting or populating the Original-To field. An intermediary MUST NOT add an Original-To header field if one already exists. An intermediary MAY have an administrative configuration to not reveal the original Request-URI, and as such, MUST NOT add an Original-To header.

An IM reply for a page-mode IM is not linked in any way to the initial IM and can end up at a different user agent from where the initial IM originated, depending on how the recipient URI gets resolved. Therefore, IM replies may traverse different intermediaries. An IMDN, on the other hand, needs to traverse the same intermediaries as the IM itself since those intermediaries may be required to report negative delivery notifications if the IM was not delivered successfully. Some of those intermediaries are, for example, store-and-forward servers that may report that an IM has been processed and later report that the IM has failed to be delivered.

For the reasons stated above, an intermediary MAY choose to remain on the path of IMDNs for a specific IM. It can do so by adding a CPIM IMDN-Record-Route header field as the top IMDN-Record-Route header field. The value of this field MUST be the intermediary's own address. An intermediary that does not support this extension will obviously not add the IMDN-Record-Route header field. This allows IMDNs to traverse directly from the IM Recipient to the IM Sender even if the IM traversed an intermediary not supporting this extension.

An intermediary receiving an IMDN checks the top IMDN-Route header field. If that header field carries the intermediary address, the intermediary removes that value and forwards the IMDN to the address indicated in the new top IMDN-Route header field. If no additional IMDN-Route header fields are present, the IMDN is forwarded to the address in the CPIM To header field.

An intermediary MUST remove any information about the final recipients of a list if the list membership is not disclosed. The intermediary does that by removing the <recipient-uri> element and/or <original-recipient-uri> element from the body of the IMDN before forwarding it to the IM Sender.

8.1. Constructing Processing Notifications

Intermediaries are the only entities that construct processing notifications. They do so only if the IM Sender has requested a "processing" notification by including a Disposition-Notification header field with value "processing".

The intermediary can create and send "processing" notifications indicating that an IM has been processed or stored. The intermediary MUST NOT send more than one IMDN for the same disposition type -- i.e., it must not send a "processing" notification indicating that an IM is being "processed" followed by another IMDN indicating that the same IM is "stored".

An intermediary constructs a "processing" notification in a similar fashion as the IM Recipient constructs a delivery notification. See Section 7.2.1.1 for details.

The following is an example:

```
Content-type: Message/CPIM

From: Bob <im:bob@example.com>
To: Alice <im:alice@example.com>
Content-type: message/imdn+xml
Content-Disposition: notification
Content-length: ...

<?xml version="1.0" encoding="UTF-8"?>
<imdn xmlns="urn:ietf:params:xml:ns:imdn">
      <message-id>34jk324j</message-id>
      <datetime>2008-04-04T12:16:49-05:00</datetime>
     <recipient-uri>im:bob@example.com</recipient-uri>
      <original-recipient-uri
         >im:bob@example.com</original-recipient-uri>
```

```
        <processing-notification>
           <status>
              <processed/>
           </status>
        </processing-notification>
      </imdn>
```

There are situations where the intermediary cannot know the fate of an IM. The intermediary in this case generates a processing notification with a <status> value of "error" to indicate so.

8.2. Constructing Delivery Notifications

Intermediaries MAY construct negative delivery notifications. They do so only if the IM Sender has requested a "negative-delivery" notification by including a Disposition-Notification header field with value "negative-delivery" AND an error was returned for that IM.

The intermediary can create and send "negative-delivery" notifications indicating that an IM has failed to be delivered. The intermediary MUST NOT send more than one IMDN for the same disposition type -- i.e., it must not send a "failed" notification indicating that an IM has failed followed by another IMDN indicating that an IMDN is "forbidden".

An intermediary constructs a "negative-delivery" notification much like the IM Recipient. See Section 7.2.1.1 for details.

8.3. Aggregation of IMDNs

As previously described, URI-List servers are intermediaries.

A URI-List server may choose (using local policy) to aggregate IMDNs or it may send individual IMDNs instead. When a URI-List server receives an IM and decides to aggregate IMDNs, it can wait for a configurable period of time or until all recipients have sent the IMDN, whichever comes first, before it sends an aggregated IMDN. Note that some IMDNs, for example "displayed" notifications, may never come due to user settings. How long to wait before sending an aggregated IMDN and before a URI-List server removes state for that IM is an administrator configuration and implementation issue.

A URI-List server MAY choose to send multiple aggregated IMDNs. A timer can be started, and when it fires, the URI-List server can aggregate whatever IMDNs it has so far for that IM, send the aggregated IMDN, and restart the timer for the next batch. This is needed for scenarios where the IM Sender has requested more than one

IMDN for a specific IM -- for example, delivery notifications as well as display notifications -- or when the URI-List server is short on resources and chooses to prioritise forwarding IMs over IMDNs.

A second timer can be running, and when it fires, the state of the IM is deleted. In this case, the URI-List server consumes any IMDNs that might arrive after that time.

Please note the references to timers in the above paragraphs are not normative and are only present to help describe one way one might implement aggregation.

A URI-List server MAY aggregate IMDNs for the case where the list membership information is not disclosed. There may be scenarios where the URI-List server starts sending aggregated IMDNs and switches to individual ones or visa versa. A timer firing often may in fact have that effect.

The aggregated IMDN is constructed using the multipart/mixed MIME type and including as individual payloads all the IMDNS that were received as message/imdn+xml.

Below is an example of aggregated IMDNs.

```
From: Bob <im:bob@example.com>
To: Alice <im:alice@example.com>
NS: imdn <urn:ietf:params:imdn>
imdn.Message-ID: d834jied93rf
Content-type: multipart/mixed;
                    boundary="imdn-boundary"
Content-Disposition: notification
Content-length: ...

--imdn-boundary
Content-type: message/imdn+xml

<?xml version="1.0" encoding="UTF-8"?>
<imdn xmlns="urn:ietf:params:xml:ns:imdn">
      <message-id>34jk324j</message-id>
      <datetime>2008-04-04T12:16:49-05:00</datetime>
     <recipient-uri>im:bob@example.com</recipient-uri>
      <original-recipient-uri
        >im:bob@example.com</original-recipient-uri>
      <delivery-notification>
         <status>
            <delivered/>
```

```
            </status>
          </delivery-notification>
        </imdn>

   --imdn-boundary
   Content-type: message/imdn+xml

   <?xml version="1.0" encoding="UTF-8"?>
   <imdn xmlns="urn:ietf:params:xml:ns:imdn">
         <message-id>34jk324j</message-id>
         <datetime>2008-04-04T12:16:49-05:00</datetime>
        <recipient-uri>im:bob@example.com</recipient-uri>
         <original-recipient-uri
            >im:bob@example.com</original-recipient-uri>
         <display-notification>
            <status>
               <displayed/>
            </status>
         </display-notification>
       </imdn>

   --imdn-boundary
```

9. Identifying Messages

Messages are typically carried in a transport protocol like SIP [RFC3261]. If the payload carried by the transport protocol does not contain any parts of type Message/CPIM, then the message is an IM. If the payload contains any parts of type Message/CPIM, and none of those parts contains a payload that is of type "message/imdn+xml", the message is an IM. It is not valid to attempt to carry both an IM and an IMDN in a multipart payload in a single transport protocol message.

A message is identified as a delivery notification by examining its contents. The message is a delivery notification if the Content-type header field present has a value of "message/imdn+xml", the Content-Disposition header field has a value of "notification", and the <delivery-notification> element appears in that XML body.

A message is identified as a processing notification or display notification in a similar fashion as a delivery notification. The difference is that, for a processing notification, the <processing-notification> element appears in the XML body. For a display notification, the <display-notification> element appears in the XML body.

10. Header Fields Formal Syntax

The following syntax specification uses the message header field syntax as described in Section 3 of RFC 3862 [RFC3862].

Header field syntax is described without a namespace qualification. Following the rules in RFC 3862 [RFC3862], header field names and other text are case sensitive and MUST be used as given, using exactly the indicated upper-case and lower-case letters.

```
Disposition-Notification =
    "Disposition-Notification" ": "
    [(notify-req *(COMMA notify-req))]

notify-req =
    ("negative-delivery" / "positive-delivery" /
     "processing" / "display" / Token) *(SEMI disp-notify-params)

disp-notify-params = Ext-param

Message-ID = "Message-ID" ": " Token

Original-To = "Original-To" ": "  [ Formal-name ] "<" URI ">"

IMDN-Record-Route =
    "IMDN-Record-Route" ": "  [ Formal-name ] "<" URI ">"

IMDN-Route = "IMDN-Route" ": "  [ Formal-name ] "<" URI ">"

SEMI    =  *SP ";" *SP ; semicolon

COMMA   =  *SP "," *SP ; comma
```

11. IMDN Format

11.1. Structure of an XML-Encoded IMDN Payload

An IMDN payload is an XML document [XML] that MUST be well-formed and MUST be valid according to schemas, including extension schemas, available to the validater and applicable to the XML document. The IMDN payload MUST be based on XML 1.0 and MUST be encoded using UTF-8.

The schema allows qualified extension elements in several positions other than the "urn:ietf:params:xml:ns:imdn" namespace. To maintain forwards compatibility (i.e., newer instance documents can be used by existing consumers), the new specifications MUST NOT extend the allowable content of this specification. The backwards compatibility

(i.e., existing instance documents can also be used by updated, new consumers) MAY break if there are conflicts with the existing qualified names of extension elements and possible future specifications. The IETF MAY specify new extension elements within the "sub-namespace" of "urn:ietf:params:xml:ns:" for this message/imdn+xml MIME type.

Possible future specifications can add new element definitions with the combine="interleave" pattern. When multiple elements of this new type are then allowed, the new definition MUST contain the <zeroOrMore> cardinality rule. If the new specification does allow only a single new element, the <optional> cardinality rule MUST be used. These cardinality requirements maintain the backwards compatibility of existing instance documents with newer consumers. Also, the new specification MUST then redefine either the "anyIMDN" extension or the individual extension points that reference it, so that new element definitions do not match with this redefined and more limited wildcard pattern.

The namespace identifier for elements defined by this specification is a URN [URN], using the namespace identifier 'ietf' defined by [URN_NS] and extended by [IANA]. This urn is: urn:ietf:params:xml:ns:imdn.

This namespace declaration indicates the namespace on which the IMDN is based.

The root element is <imdn>. The <imdn> element has sub-elements, namely <message-id>, <datetime>, <recipient-uri>, <original-recipient-uri>, <subject>, and one of <delivery-notification>, <processing-notification>, or <display-notification>. A <status> also appears as a sub-element of <delivery-notification>, <processing-notification>, and <display-notification>. The elements are described in detail in the following sections.

<imdn> can be extended in the future to include new disposition notification types or other elements, as described in Section 11.1.9.

11.1.1. The <message-id> Element

The <message-id> element is mandatory according to the XML schema and carries the message ID that appeared in the Message-ID header field of the IM.

11.1.2. The <datetime> Element

The <datetime> element is mandatory and carries the date and time the IM was sent (not the IMDN). This information is obtained from the DateTime header field of the IM.

11.1.3. The <recipient-uri> Element

The <recipient-uri> element is optional and carries the URI of the final recipient. This information is obtained from the CPIM To header field of the IM.

11.1.4. The <original-recipient-uri> Element

The <original-recipient-uri> element is optional and carries the URI of the original recipient. It MUST be present if the IM carried the Original-To header field. This information is obtained from the Original-To header field of the IM.

11.1.5. The <subject> Element

The <subject> element is optional. If present, it MUST carry the text and language attributes that were in the Subject header field, if any. This allows for a human-level correlation between an IM and an IMDN. If there are more than one Subject header fields in an IM, selecting any one of them to place in the IMDN payload <subject> element will suffice. The sender then needs to compare Subject header fields until a match or not match is determined.

11.1.6. The <delivery-notification>, <processing-notification>, and <display-notification> Elements

The appearance of one of the <delivery-notification>, <processing-notification>, and <display-notification> elements is mandatory and carries the disposition type that the IM Sender requested and is being reported. It carries the sub-element <status>.

11.1.7. The <status> Element

The <status> element is mandatory and carries the result of the disposition request. For notification type <delivery-notification>, it can carry one of the sub-elements <delivered>, <failed>, <forbidden>, or <error>. For notification type <display-notification>, it can carry one of the sub-elements <displayed>, <forbidden>, or <error>. For notification type <processing-notification>, it can carry one of the sub-elements <processed>,

<stored>, <forbidden>, or <error>. <forbidden> means the disposition was denied. <error> means internal server error. The <status> element can also be extended to carry any other status extension.

11.1.8. MIME Type for IMDN Payload

The MIME type for the IMDN payload is "message/imdn+xml". The IMDN MUST identify the payload as MIME type "message/imdn+xml" in the Content-type header field.

11.1.9. The RelaxNG Schema

```
<?xml version="1.0" encoding="UTF-8"?>
     <grammar
       xmlns="http://relaxng.org/ns/structure/1.0"
       xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
       datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
       ns="urn:ietf:params:xml:ns:imdn">

         <start>
             <element name="imdn">
                 <element name="message-id">
                     <data type="token"/>
                 </element>
                 <element name="datetime">
                     <data type="string"/>
                 </element>
                 <optional>
                     <element name="recipient-uri">
                         <data type="anyURI"/>
                     </element>
                     <element name="original-recipient-uri">
                         <data type="anyURI"/>
                     </element>
                     <optional>
                         <element name="subject">
                             <data type="string"/>
                         </element>
                     </optional>
                 </optional>
                 <choice>
                     <ref name="deliveryNotification"/>
                     <ref name="displayNotification"/>
                     <ref name="processingNotification"/>
                     <empty/>
                 </choice>
                 <ref name="imdnExtension"/>
             </element>
```

```
        </start>

        <define name="deliveryNotification">
            <element name="delivery-notification">
                <element name="status">
                    <choice>
                        <element name="delivered">
                            <empty/>
                        </element>
                        <element name="failed">
                            <empty/>
                        </element>
                        <ref name="commonDispositionStatus"></ref>
                    </choice>
                    <ref name="deliveryExtension"/>
                  </element>
             </element>
        </define>

        <define name="displayNotification">
            <element name="display-notification">
                <element name="status">
                    <choice>
                        <element name="displayed">
                            <empty/>
                        </element>
                        <ref name="commonDispositionStatus"></ref>
                    </choice>
                    <ref name="displayExtension"/>
                </element>
            </element>
        </define>

        <define name="processingNotification">
            <element name="processing-notification">
                <element name="status">
                    <choice>
                        <element name="processed">
                            <empty/>
                        </element>
                        <element name="stored">
                            <empty/>
                        </element>
                        <ref name="commonDispositionStatus"></ref>
                    </choice>
                    <ref name="processingExtension"/>
                 </element>
            </element>
```

```
    </define>

    <define name="commonDispositionStatus">
        <choice>
            <element name="forbidden">
                <empty/>
            </element>
            <element name="error">
                <empty/>
            </element>
        </choice>
    </define>

    <!-- <imdn> extension point for the extension schemas to add
         new definitions with the combine="interleave" pattern.
         Extension schemas should add proper cardinalities.  For
         example, the <zeroOrMore> cardinality should be used if
         the extension is to allow multiple elements, and the
         <optional> cardinality should be used if the extension
         is to allow a single optional element. -->

    <define name="imdnExtension">
        <zeroOrMore>
            <ref name="anyIMDN"/>
        </zeroOrMore>
    </define>

    <!-- delivery-notification <status> extension point -->
    <define name="deliveryExtension">
        <zeroOrMore>
            <ref name="anyIMDN"/>
        </zeroOrMore>
    </define>

    <!-- display-notification <status> extension point -->
    <define name="displayExtension">
        <zeroOrMore>
            <ref name="anyIMDN"/>
        </zeroOrMore>
    </define>

    <!-- processing-notification <status> extension point -->
    <define name="processingExtension">
        <zeroOrMore>
            <ref name="anyIMDN"/>
        </zeroOrMore>
    </define>
```

```
        <!-- wildcard definition for complex elements (of mixed type)
             unqualified or qualified in the imdn namespace.
             Extension schemas MUST redefine this or the
             individual, previous definitions that use this definition.
             In other words, the extension schema MUST reduce the
             allowable content in order to maintain deterministic
             and unambiguous schemas with the interleave pattern. -->
        <define name="anyIMDN">
            <element>
                <anyName>
                    <except>
                        <nsName ns="urn:ietf:params:xml:ns:imdn"/>
                        <nsName ns=""/>
                    </except>
                </anyName>
                <ref name="anyExtension"/>
            </element>
        </define>

       <!-- the rest of the "anyIMDN" wildcard definition -->
        <define name="anyExtension">
            <zeroOrMore>
               <choice>
                   <attribute>
                      <anyName/>
                   </attribute>
                   <ref name="any"/>
               </choice>
            </zeroOrMore>
        </define>

        <!-- wildcard type for complex elements (of mixed type)
             without any namespace or content restrictions -->
        <define name="any">
            <element>
                <anyName/>
                <zeroOrMore>
                   <choice>
                      <attribute>
                         <anyName/>
                      </attribute>
                      <text/>
                      <ref name="any"/>
                   </choice>
                </zeroOrMore>
```

```
            </element>
        </define>

    </grammar>
```

12. Transporting Messages Using SIP

12.1. Endpoint Behaviour

12.1.1. Sending Requests

The IM Sender constructs a SIP MESSAGE request using RFC 3428 [RFC3428]. The Content-type header field indicates the MIME type of the request payload. When using this extension, the Content-type header field MUST be of MIME type "message/cpim" [RFC3862] for both IMs and IMDNs. The IM Sender constructs the payload according to Section 7.

The IM Sender constructs a SIP MESSAGE request to multiple recipients in a similar manner as a SIP MESSAGE request to a single recipient. "Multiple-Recipient MESSAGE Requests in SIP" [RFC5365] describes the differences.

IM Senders can remain anonymous. For example, the sender can set the SIP From header field of the SIP message to an anonymous URI. As there is no return address, anonymous IM Senders SHOULD NOT request disposition notifications. An IM Recipient MAY ignore such a request if the IM Sender is anonymous.

12.1.2. Sending Responses

An endpoint receiving a SIP MESSAGE request constructs a SIP response according to RFC 3428 [RFC3428]. Of course, an endpoint will send a SIP response to the MESSAGE request regardless of the type of message (IM or IMDN) it has received or the disposition type for which it has been asked.

12.1.3. Receiving Requests

12.1.3.1. Instant Message

A SIP MESSAGE request is identified as an IM by examining its contents according to Section 9.

If an IM Recipient received a SIP MESSAGE request that is an IM requesting a positive-delivery notification, and that IM Recipient has constructed and sent a SIP 2xx class response, it MAY generate a positive-delivery notification after making sure that the IM has been

delivered to the user or application. A gateway, for example, can generate a 2xx response before the final recipient received the IM. The IM Recipient constructs a positive-delivery notification according to Section 7.2.1.1. The IM Recipient places the message as the payload in a SIP MESSAGE request.

If an IM Recipient received a SIP MESSAGE request that is an IM requesting a negative-delivery, and that IM Recipient has constructed and sent a 2xx class response, it SHOULD generate a negative-delivery notification if it learnt that the final recipient or application did not receive the IM (a gateway, for example, can generate a 2xx response before it has an error response from downstream or before any internal timers fire waiting for a response). The negative-delivery notification is constructed according to Section 7.2.1.1. The message is then placed as the payload in a SIP MESSAGE request.

If an IM Recipient received a SIP MESSAGE request that is an IM requesting a negative-delivery notification, and the IM Recipient has constructed and sent a non-2xx final response, it MUST NOT generate a negative-delivery notification.

If an IM Recipient received a SIP MESSAGE request that is an IM requesting a display notification, and that IM Recipient has constructed and sent a SIP 2xx class response, it MAY generate a display notification after making sure that the IM has been presented to the user or application. It is outside the scope of this document to discuss how a determination can be made whether the IM has been read. Note that the decision whether or not to send a display notification can be left to the user. An application may allow a user to configure such a choice. The IM Recipient constructs the display notification according to Section 7.2.1.2. The IM Recipient places the message as the payload in a SIP MESSAGE request.

For IMDNs, the IM Recipient populates the SIP Request-URI and the SIP To header field using the address that appeared in the SIP From header field in the IM.

12.1.3.2. Delivery Notification

A SIP MESSAGE request is identified as a delivery notification by examining its contents according to Section 9.

12.1.3.3. Display Notification

A SIP MESSAGE request is identified as a display notification by examining its contents according to Section 9.

12.2. Intermediary Behaviour

In this context, intermediaries include application servers (including URI-List and store-and-forward servers) and gateways. SIP Proxies MUST NOT generate IMDNs but MUST forward them like any other SIP request.

Intermediaries forward a SIP MESSAGE request to multiple recipients according to [RFC5365].

If an intermediary receives an IM, the intermediary examines the body. If the body is of type "message/cpim", the intermediary then looks for a Disposition-Notification CPIM header field in the message. If the Disposition-Notification CPIM header field has either the value "positive-delivery" or "negative-delivery", and, in processing the IM, the intermediary generates a SIP 2xx class response to the MESSAGE request, then the intermediary performs the following actions.

If the Disposition-Notification header field contains a value of "positive-delivery", the intermediary MUST NOT generate a delivery notification if it receives a SIP 2xx class response for the sent IM. Just because a downstream entity received a MESSAGE request does not mean the message was relayed to its ultimate destination or was delivered. Thus, the intermediary cannot say delivery occurred just because it received a 2xx response.

If the Disposition-Notification header field contains a value of "negative-delivery", the intermediary SHOULD generate a delivery notification if it receives a SIP 4xx, 5xx, or 6xx class final response for the sent IM. If it has received a SIP 2xx class response followed by a negative-delivery notification, the intermediary forwards that negative-delivery notification or aggregates it.

If the Disposition-Notification header field contains a value of "processing", the intermediary MAY generate a processing notification after it has forwarded or stored the IM. The rest of the procedures in Section 8.1 apply.

The procedure for generating such an IMDN is the same as that of an IM Recipient (Section 7.2.1.1).

The <recipient-uri> element of the XML body is populated with the URI of the IM Recipient.

If an intermediary receives a SIP MESSAGE request carrying a positive delivery notification or a display notification, it forwards it using the rules in Section 8.

13. Transporting Messages using MSRP

The Message Session Relay Protocol (MSRP) [RFC4975] already provides a built-in mechanism to supply positive and negative delivery reports. These reports do not provide built-in display or processing notifications. However, these notifications in session-mode are not as useful as they are for page-mode. This is because the base use case for MSRP is that the recipient user agent immediately renders SEND requests sequentially, providing the session experience. This is unlike page-mode requests where a user has to actively initiate the display of the message. That is, they need to click on a button, open a message, and so on to read the message.

If new requirements arise in the future determining the need for IMDN in MSRP, new specifications can be drafted.

14. Security Considerations

IMDNs provide a fine-grained view of the activity of the IM Recipient, and thus deserve particularly careful confidentiality protection so that only the intended recipient of the IMDN will receive the IMDN. In most cases, the intended recipient of the IMDN is the IM Sender.

Since the IM transport protocol carries the IMDN, all security considerations of the underlying IM protocol also apply to the IMDNs.

The threats in the IMDN system, over and beyond the threats inherent to IM, include the following:

- A malicious endpoint attempts to send messages to a user that would normally not wish to receive messages from that endpoint by convincing the IMDN system to "bounce" an IMDN from an unsuspecting endpoint to the user.

- A malicious endpoint attempts to flood an IM Sender with IMDNs by convincing a URI-List server to send IMDNs to an unsuspecting IM Sender.

- A malicious intermediary or node attempts to flood a target node with IMDNs by inserting the target’s address in the From field or IMDN-Record-Route field.

- A malicious node in the network attempts to modify an IMDN from an IM Recipient.

- A malicious intermediary attempts to forward an IMDN from an IM Recipient to the IM Sender, where the IM Recipient would not normally forward the IMDN to that IM Sender if the IM Recipient knew the identity of the IM Sender.

- A malicious endpoint attempts to discover the Request-URI of an endpoint beyond an intermediary, where the endpoint would normally wish to keep its identity private from the malicious endpoint.

- A malicious node in the network attempts to eavesdrop on IMDN traffic to, for example, learn Request-URI or traffic pattern information.

- A malicious node in the network attempts to stage a denial-of-service attack on an intermediary by requesting a large list expansion.

The protocol cannot protect against attacks that include the following:

- A malicious intermediary directly revealing the identity of a downstream endpoint that would not normally wish its identity revealed. Keeping such information private is an intermediary implementation issue.

- A malicious IM Recipient alters the time of the IMDN. There is no protocol mechanism for ensuring that the IM Recipient does not lie about the time or purposely holds an IMDN for transmission to make it appear that the IM displayed to the user was read later than it actually was.

- A deletion attack on an IMDN. This is a trade-off between privacy and security. The privacy considerations allow the IM Recipient to silently ignore an IMDN request. Any mechanism that would reliably indicate that a malicious node deleted an IM Recipient's IMDN would also serve the purpose of detecting an IM Recipient that chose not to issue an IMDN.

To combat eavesdropping, modification, and man-in-the-middle attacks, we require some level of authentication and integrity protections. That said, there are circumstances where strong integrity would be overkill. The presumption is that the IM Sender has, and sets the expectation for, the level of protection. The procedures for integrity protection are as follows.

- If the IM Recipient has a certificate, it MUST sign the IMDN. Signing the IMDN provides integrity protection. While an intermediary can replace the IMDN body, the IM Sender (the recipient of the IMDN) can validate the signature and note the IMDN does not come directly from the IM Receiver. This is not a problem if the IM Sender trusts the intermediary. Likewise, an IMDN in response to a signed IM without a signature indicates something bad might have happened.

- If the IM is encrypted, the IM Recipient or intermediary MUST encrypt the IMDN body, as an attacker may attempt to discern the user's activity profile and identity from sniffing IMDNs on the network.

- The two above rules are cumulative.

The IM Recipient or intermediary MUST be capable of accessing the IM Sender's public certificate in order to verify the signature in the IM.

CPIM security considerations [RFC3862] apply here, as this is an extension of CPIM. In order to make the IMDN mechanism independent of the transport protocol, the Working Group made the design choice of putting routing information into the IMDN application-layer payload. One consequence of this choice is it eliminates the possibility of having end-to-end encryption.

An attacker can mount a distributed denial-of-service attack on a node by sending lots of IMs to the node with IMDN requests. Note that this is the same problem as there is without IMDN; IMDN simply linearly increases the load on the node under attack. One can mitigate, but not eliminate, this threat by the endpoint immediately ignoring requests that are not authenticated.

One way to address the potential for a malicious node to use the IMDN system to anonymize attacks is to log all IMDN requests on the IM Recipient user agent. This allows for tracking of attacks, if only after they occur. Note this also puts a burden on the IM Recipient user agent host. Limited user agents may not be able to preserve much of a log.

Likewise, an attacker can mount a denial-of-service attack on an intermediary by asking the intermediary to explode a large list.

The following security considerations apply when using IMDNs.

14.1. Forgery

IMs can be forged. To protect against that, an IM can be signed. An intermediary that receives a signed message and needs to modify any part of it that is included in the signature (like adding an Original-To header field to the CPIM header fields) MUST consume the IM and create a new copy of it that the intermediary signs itself.

IMDNs may be forged as easily as ordinary IMs. Endpoints and intermediaries that wish to make automatic use of IMDNs should take appropriate precautions to minimize the potential damage from denial-of-service attacks. Security threats related to forged IMDNs include the sending of a falsified IMDN when the indicated disposition of the IM has not actually occurred. For example, display notification could be forged to indicate that an IM has been displayed to the Recipient. Unsolicited IMDNs is also another form of forgery.

14.2. Confidentiality

There may be cases where an IM Recipient does not wish to reveal that she has received, or in fact read, the IM. In this situation, it is acceptable for the IM Recipient to silently ignore requests for an IMDN. It is strongly RECOMMENDED that the IM Recipient obtain the user's consent before sending an IMDN. Circumstances where the IM Recipient does not ask for the user's consent include IM systems that, for regulatory reasons, are required to issue an IMDN, such as in the health care field or financial community.

An IM Recipient can obtain such consent by a prompt or dialog box on a per-IM basis, globally through the user's setting of a preference, or another, user-configurable mechanism. The user might also indicate globally that IMDNs are never to be sent or that a "forbidden" IMDN status is always sent in response to a request for an IMDN.

There are situations where a user sends an IM and requests IMDNs to a list whose member information is not disclosed. In this situation, the user will learn of the list members. Therefore, in this case, the URI-List server MUST remove any information about list members. If the number of members in the list is also not disclosed, the URI-List server MUST only deliver one aggregated IMDN. Alternatively, the URI-list server MAY reject the IM.

It is possible for a list server to not understand IMDN. IM Recipients may note the To header field is a list name and not the IM Recipient's name. In this case, the IM Recipient can take the appropriate action if it wishes to keep its identity private.

An unencrypted IMDN could reveal confidential information about an encrypted IM. The same level of security applied to an IM MUST be applied to its IMDNs. For example, if an IM is signed and encrypted, the IMDN must be signed and encrypted.

14.3. IMDN as a Certified Delivery Service

IMDNs cannot be relied on as a guarantee that an IM was or was not seen by the user. Even if IMDNs are not actively forged, they may be lost in transit. Moreover, the IM Recipient may bypass the IMDN issuing mechanism through policy or manipulation of their user agent Server.

15. IANA Considerations

15.1. message/imdn+xml MIME TYPE

This document registers a new MIME type "message/imdn+xml", and registers a new XML namespace.

This specification follows the guidelines of RFC 3023 [RFC3023].

MIME media type: message

MIME subtype name: imdn+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [RFC3023].

Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [RFC3023].

Security considerations: See Section 10 of RFC 3023 [RFC3023] and Section 14 of this document.

Interoperability considerations: none

Published specification: This document

Applications which use this media type: This media type is used to support CPIM-based instant Messaging.

Additional information: none

Magic number: none

File extension: .cl or .xml

Macintosh file type code: "TEXT"

Personal and email address for further information: Hisham Khartabil (hisham.khartabil@gmail.com)

Intended Usage: COMMON

Author/change controller: The IETF

15.2. XML Registration

This section registers a new XML namespace and schema, as per guidelines in the IETF XML Registry [IANA].

URI: urn:ietf:params:xml:ns:imdn

XML: The schema for this namespace is in Section 11.1.9 above.

Registrant Contact: IETF, SIMPLE working group, Hisham Khartabil (hisham.khartabil@gmail.com)

15.3. URN Registration for IMDN Header Parameters

Per [RFC3553], please establish the following registry. New entries to the registry are Specification Required.

Registry name: urn:ietf:params:imdn

Specification: RFC 5438. Additional values may be defined by a Standards Action [RFC5226] that updates or obsoletes RFC 5438.

Repository: RFC 5438

Index value: Values subordinate to urn:ietf:params:imdn require RFC publication. The index value is the IMDN header name. The index value must follow the rules for a legal IMDN header name. In particular, the IMDN header name, and thus the index value to register, must be a string of octets taken from the restricted set of US-ASCII characters per Section 3.1 of [RFC3553]. The index value is case sensitive.

URN Formation: The URI for a header is formed from its name by

a) replacing any non-URN characters (as defined by RFC 2141 [URN]) with the corresponding '%hh' escape sequence (per RFC 3986 [RFC3986]) and

b) prepending the resulting string with 'urn:ietf:params:imdn:'.

Thus, the URI corresponding to the CPIM message IMDN header 'Disposition-Notification:' would be 'urn:ietf:params:imdn:Disposition-Notification'.

Initial values:

Index Value	Reference
Disposition-Notification	RFC5438 Section 6.2
Message-ID	RFC5438 Section 6.3
Original-To	RFC5438 Section 6.4
IMDN-Record-Route	RFC5438 Section 6.5
IMDN-Route	RFC5438 Section 6.6

15.4. Content-Disposition: notification

This document registers one new Content-Disposition header field "disposition-types": notification, which has been recorded in the IANA registry for Mail Content Dispositions.

Descriptions of this "disposition-types", including motivation and examples, are given in Section 7.2.1.1 and Section 9.

Short descriptions suitable for the IANA registry are:

notification: the payload of the message carrying this Content-Disposition header field value is an Instant Message Disposition Notification as requested in the corresponding Instant Message.

16. Acknowledgements

Special thanks to Jari Urpalainen for the thorough review and suggestions for the RelaxNG schema.

The authors would also like to thank Paul Kyzivat, Ben Campbell, Adam Roach, Gonzalo Camarillo, Frank Ellermann, Sean Olson, Eva Leppanen, Miguel Garcia, Eric McMurry, Jon Peterson, and Robert Sparks for their comments and support. In addition, we would like to thank the Gen-Art reviewer extraordinaire, Spencer Dawkins.

17. References

17.1. Normative References

[IANA] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[RFC3862] Klyne, G. and D. Atkins, "Common Presence and Instant Messaging (CPIM): Message Format", RFC 3862, August 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[URN] Moats, R., "URN Syntax", RFC 2141, May 1997.

[XML] Bray, T., "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C CR CR-xml11-20011006, October 2000.

17.2. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, June 2003.

[RFC3798] Hansen, T. and G. Vaudreuil, "Message Disposition Notification", RFC 3798, May 2004.

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5365] Garcia-Martin, M. and G. Camarillo, "Multiple-Recipient MESSAGE Requests in the Session Initiation Protocol (SIP)", RFC 5365, October 2008.

[URN_NS] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

Authors' Addresses

Eric Burger
Unaffiliated
New Hampshire
USA

Phone:
Fax: +1 603 457 5933
EMail: eburger@standardstrack.com

Hisham Khartabil
Ericsson Australia
Melbourne
Australia

Phone: +61 416 108 890
EMail: hisham.khartabil@gmail.com