

Neda Communications, Inc.  
DRAFT 1234  
Category: Informational

M. Banan  
Neda Communications, Inc.

February 1999

## **DRAFT-RFC-Embedded Response Mime Message Type**

### **STATUS OF THIS MEMO**

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

### **ABSTRACT**

In its current form, this paper is an early draft of a minimum subset of MIME content type/subtypes to be used by EMSD.

This paper first introduces the concept of automated processing of typed information. It then enumerates EMSD Message types to be supported in the EMSD domain. Issues relevant to both origination and processing are discussed.

This paper is far from being complete. The embedded response section currently focuses on the formal specification. More detailed description will be provided in later versions of this paper.

The following topics are discussed in some detail:

- EMSD Transfer of Typed Information
- EMSD Message Types
- Registry of MIME Content Types Used by EMSD
- Specification of New MIME Content Types
- Issues



# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	About This Paper . . . . .	5
1.2	Requirements . . . . .	6
<b>2</b>	<b>EMSD Transfer of Typed Information</b>	<b>9</b>
2.1	About Application Integrated Messaging . . . . .	9
2.1.1	Automated processing of application data . . . . .	9
2.2	Use of MIME . . . . .	10
2.2.1	Multipurpose Internet Mail Extensions (MIME) . . . . .	10
<b>3</b>	<b>EMSD Message Types</b>	<b>13</b>
3.1	Categories of EMSD Message Types . . . . .	13
3.1.1	Public Message Types . . . . .	14
3.1.2	Internal Application Message Types . . . . .	14
3.1.3	Administrative Message Types . . . . .	14
3.2	Text Message . . . . .	14
3.3	Text with Embedded Response Message . . . . .	15
3.3.1	1-Part Text with Embedded Response Message . . . . .	15
3.3.2	2-Part Text with Embedded Response Message . . . . .	15
3.4	Response Message . . . . .	15
3.4.1	General Rules for Forming a Response Message . . . . .	16
3.5	Forwarded RFC 822 E-Mail Message . . . . .	16
3.6	Binary Message . . . . .	16
3.7	Fax Notification Message . . . . .	17
3.8	Voice Mail Notification Message . . . . .	17
3.9	Calendar Operation . . . . .	17
3.10	Address Book Operation . . . . .	17
3.11	Configuration Request . . . . .	18
3.12	Configuration Response . . . . .	18
3.13	Location Request ????? needed ?????? . . . . .	18
3.14	Location Response ????? needed ?????? . . . . .	18
3.15	Icons . . . . .	19
<b>4</b>	<b>Registry of MIME Content Types Used by EMSD</b>	<b>21</b>
4.1	Content Types . . . . .	21
4.2	Character Sets . . . . .	22
4.3	Content Transfer Encoding . . . . .	22
<b>5</b>	<b>Specification of New MIME Content Types Used by EMSD</b>	<b>23</b>
5.1	application/lsm-body-part . . . . .	23

5.1.1	The application/lsm-body-part Language . . . . .	24
5.1.2	The Tagged Data List (TDL) Syntax . . . . .	24
5.1.3	The Lisp-List (LL) Syntax . . . . .	27
5.1.4	Calendar Operation Examples . . . . .	29
5.1.5	Address Book Operation Examples . . . . .	30
5.1.6	Configuration Request/Response Examples . . . . .	30
5.1.7	Embedded Response Specification Examples . . . . .	31
<b>6</b>	<b>Issues</b>	<b>39</b>
6.1	Encoding of EMSD Content-Type . . . . .	39
6.2	Origination Facilities . . . . .	39
<b>A</b>	<b>MIME Registration of Content-type/subtype Values with IANA</b>	<b>41</b>
<b>B</b>	<b>MIME Assigned Numbers</b>	<b>43</b>

# Chapter 1

## Introduction

This paper analyzes issues surrounding the transfer of typed information through EMSD.

EMSD can transfer various forms of data without regard to its type.

When transferring data through EMSD, the ability to automatically identify body part types within a message, and to have that type recognizable to a receiving application, greatly increases the utility of the information to its recipient.

The ability to automatically identify a body part type may increase the level of transparent interoperability between mail senders and receivers, potentially of any platform, by allowing automatic processing.

In order to realize the benefits of transparent interoperability, many infra-structural elements need to be in place. These include global identification and recognition of application data which is a critical element in the evolution towards transparent interoperability.

Currently, a technical mechanism exists for identifying application data types in EMSD, it is called MIME. However, in order to guarantee complete interoperability, the specific definition of all EMSD MIME content types should be fully specified and enforced. This paper attempts to do that.

### 1.1 About This Paper

In its current form, this paper is an early draft of a minimum subset of MIME content type/subtypes to be used by EMSD.

This paper first introduces the concept of automated processing of typed information. It then enumerates EMSD Message types to be supported in the EMSD domain. Issues relevant to both origination and processing are discussed.

This paper is far from being complete. The embedded response section currently focuses on the

formal specification. More detailed description will be provided in later versions of this paper.

The following topics are discussed in some detail:

- EMSD Transfer of Typed Information
- EMSD Message Types
- Registry of MIME Content Types Used by EMSD
- Specification of New MIME Content Types
- Issues

## 1.2 Requirements

This paper has not been based on formal requirements.

Natasha product requirements document and various informal discussions have been the basis of the choices represented in this paper.

Summary of some of these discussions are kept in this section.

Our hope is that through informal iterative reviews we will be able to make sure that this paper in fact addresses all the requirements.

### Quick List

Here is a quick list of entities required for different types of messages per the Natasha requirements. They influence both the EMSD requirements and TBD broadcast services.

Point to Point Content Types:  
Embedded Replies - the message also contains replies.

Info Service Notification - the message is a notification of info services.  
(point to point)

Info Service Type - the type of service (news, weather, sports) or possibly name of the service to direct it to a folder.

Configuration Data - the message contains configuration information for the device (the list of configurable items could go on ad finitum).

Fax Notification - the message is a notification of a fax received at fax server.

Vmail Notification - the message is a notification of a voicemail received at the vmail server.

Address Book Update Notification - the message is a notification of a change to the Address Book.

Address Book Change type - Add, Delete, Replace Entry; Replace All, Backup All

Address Book Data - the actual data for the change

Calendar Update Notification - the message is a notification of a change to the Address Book.

Calendar Change type - Add, Delete, Replace Entry; Replace All, Backup All

Calendar Data - the actual data for the change

Location Request - a request for the RSSI values of known MDDBS's

Location Data - the data returned from the device from a location request.

(there may also be the opportunity to have specific operations occur to facilitate provisioning and customer experience, such as being able to activate the device from the device, request billing info or make changes to your account with your device, etc. )

- Broadcast Content types:  
Embedded Responses - see above.  
Info Service Notification - see above.  
Info Service Type - see above.

Regards,

DV



## Chapter 2

# EMSD Transfer of Typed Information

The ability to identify, in a standard and globally unique way, types of enclosed application data is desirable to increase the utility of the application data to its recipients and also to facilitate implementation of mail-enabled and mail-aware applications. For example, a meeting scheduler might define its application data as a standard representation for information about proposed meeting dates. An intelligent user agent would use this information to conduct a dialog with the user, and might then send further mail based on that dialog.

### 2.1 About Application Integrated Messaging

In order to transparently include arbitrary application data within an e-mail message, and also to have that data successfully translated back to its original form upon delivery of the e-mail message, there must be some way of uniquely identifying the included data.

#### 2.1.1 Automated processing of application data

UA implementations may provide different mechanisms for selecting what application should be launched upon receipt of the particular application data contained in a message. This mechanism may be fixed (static) or extensible (dynamic). UA implementations offering mechanisms that allow for dynamic – “on the fly” – configuration of a UA’s table of known extended body part types provide more flexibility.

Figure 2.1 depicts an example of a UA implementation capable of dynamic recognition and processing of extended body part types. In this example, the “UA’s Extended Body Part Table” may consist of an extensible number of entries, each of which associate a body part type – identified by an Object Identifier – with a specific application program that can process the information object contained in the body part. After receiving a message from the MTS, the UA consults its table and decides what application to invoke to process the extended body part contained in the message.

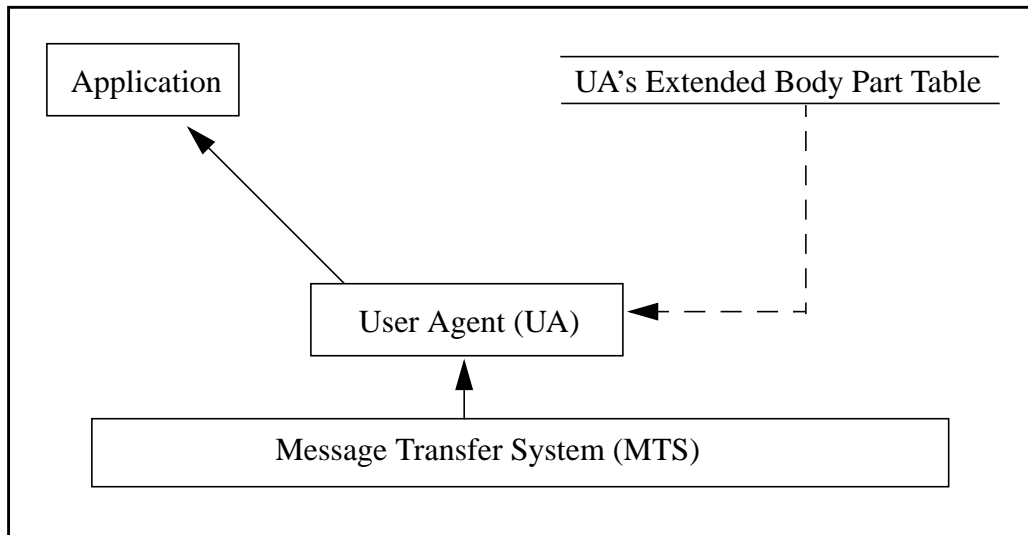


Figure 2.1: Automated processing of Application Data

From a theoretical perspective, using the MIME feature, UAs can automate the invocation of any application on the application data contained in messages. While this can provide enormous benefits, it is also important to recognize the serious security implications of this capability.

## 2.2 Use of MIME

The format of mail used in Internet is defined by RFC-822, [2]. EMSD is patterned after RFC-822. The format of headers is quite simple. The entire header is encoded in 7 bit ASCII as lines of text. Furthermore, RFC-822 only specifies a format for messages and does not specify any delivery mechanisms.

Message body parts are structured using the MIME mechanism.

### 2.2.1 Multipurpose Internet Mail Extensions (MIME)

The Internet Engineering Task Force (IETF) has developed a document titled "Multipurpose Internet Mail Extensions" or MIME, RFC-1341, [1] and 1342 [3]. The title is actually somewhat misleading.

MIME is the official proposed standard format for multimedia Internet mail encapsulated inside standard Internet RFC 822 messages. Facilities include sending multiple objects in a single message, character sets other than US-ASCII, multi-font text messages, non-textual material such as images and audio fragments, and other extensions.

MIME defines structure for Internet message bodies through enhancements to the Content-Type

field.

In MIME, the term “content-type” is used to refer to an information object contained in the body of a message.

MIME, the Multi-purpose Internet Mail Extensions, is a freely available specification that offers a way to interchange multi-media e-mail among many different computer systems. MIME supports not only several pre-defined types of non-textual message contents, such as audio, GIF files and PostScript images, but also permits you to define your own types of message parts.

### Pre-Defined MIME Types

Each part of a multimedia message identifies what type of information is carried in the message part. For example, a message part containing audio data might have either type `audio/basic` or type `audio/x-next`. Both are `audio` types; the *subtypes* are `basic` and `x-next`.

An entire MIME message—as opposed to an individual part of a multipart message—can also have a type. For example, a message might have the type `text/plain`, and consist entirely of plain text. A MIME message containing parts of different types has the umbrella type `multipart/mixed`.

Here are brief summaries of the pre-defined types and subtypes in MIME version 1.0.

**application/octet-stream** is for “other” kinds of data, either uninterpreted binary data or information to be processed by a mail-based application.

**application/postscript** is for PostScript programs (typically documents represented in PostScript form).

**audio/basic** is for audio data encoded using 8-bit ISDN (Pulse Code Modulation). A sample rate of 8000/second and a single channel is assumed.

**image/gif** is for GIF (graphical interchange format) image data.

**image/jpeg** is for JPEG (joint picture experts group) image data.

**message/external-body** is for specifying large bodies by reference to an external data source.

**message/partial** is for partial messages, to permit the fragmented transmission of bodies that are thought to be too large to be passed through mail transport facilities.

**message/rfc822** is an encapsulated RFC 822 conformant message which may have its own type.

**multipart/alternative** represents the same data in multiple formats.

**multipart/digest** is for multipart entities in which each part is an encapsulated message.

**multipart/mixed** is the primary way to represent a MIME message containing parts of various different types.

**multipart/parallel** is for data intended to be viewed simultaneously.

**text/plain** indicates plain (unformatted) text.

**video/mpeg** is for MPEG (motion picture experts group) video data. Video requires the capability to display moving images, typically including specialized hardware and software.

There may be other registered types and subtypes down the road. MIME also allows arbitrary subtypes whose names are prefixed with "x-", but anything else is reserved for registered types.

RFC 1341, [1], contains more detailed explanations of required or optional attributes to be used with particular types.

## Chapter 3

# EMSD Message Types

Use of EMSD Message types is not always within the same scope. Some message types are intended to be originated and processed by the the entire messaging community. Other EMSD message types have a narrower scope of use.

EMSD Message Types enumerated in this paper do not include any of broadcast related services types.

Each EMSD message type falls into one of three message categories. These categories are listed in the next section. The following sections then go on to describe each EMSD message type and its associated MIME structure(s). Some of the MIME content types are new and defined expressly for EMSD. These new content types are specified in chapter 5.

For EMSD specific new content types in addition to the syntax specified for use by EMSD devices – Tagged Data List (TDL) –, an experimenatl temporary syntax called “Lisp-List (LL) syntax” has also been defined.

NOTE: “Lisp-List (LL) syntax” is not an integral part of this specification and future releases of this specification will not include it.

### 3.1 Categories of EMSD Message Types

Each EMSD message type is classified as one of following message categories:

- Public Message Types
- Internal Applications Message Types
- Administrative Message Types

### 3.1.1 Public Message Types

EMSD defines the following types of messages enumerated below. For the purposes of this document each type of message is given an abbreviation in parenthesis.

- Text Message (TxtMsg)
- Text with Embedded Response Message (TxtEmbrRspMsg)
- Response Message (RspMsg)
- Forwarded RFC 822 E-Mail Message (FwdMsg)
- Binary Message (BinMsg)
- Fax Notification Message (FaxNotificationMsg)
- Voice Mail Notification Message (VmailNotificationMsg)

### 3.1.2 Internal Application Message Types

- Address Book Operation (AddrBookMsg)
- Calendar Operation (CalReqMsg)
- Location Request (LocReqMsg)
- Location Response (LocRspMsg)

### 3.1.3 Administrative Message Types

- Configuration Request (ConfigReqMsg)
- Configuration Response (ConfigRspMsg)

## 3.2 Text Message

An EMSD Text Message (TxtMsg) has a body part that consists of plain text or richly formatted text. The structure of a 'TxtMsg' is:

```
Content-Type:      text/plain
```

### 3.3 Text with Embedded Response Message

An EMSD Text-with-Embedded-Response Message (TxtEmbRspMsg) is a message comprising two pieces of information, namely:

- A Text Message
- An Embedded Response Specification (ERS)

The first piece is a text message. The second piece is an embedded response specification (ERS) which is associated with the text message, and is also provided by the sender. The ERS is processed by the recipient's user agent (UA) software. The software provides the appropriate presentation of the particular ERS, allowing the recipient to respond.

See section 5.1.1, for the specification of this language.

From a MIME message structuring perspective, a TxtEmbRspMsg can be represented in two ways, namely: either as a single part (TxtEmbRspMsg\_1Part) message or a two part (TxtEmbRspMsg\_2Part) message.

#### 3.3.1 1-Part Text with Embedded Response Message

In the case of a TxtEmbRspMsg\_1Part message, the text message is contained entirely in the **Subject:** line of the TxtEmbRspMsg message header. The ERS is expressed in the message body. Its MIME structure is:

```
Content-Type: application/lsm-body-part
```

#### 3.3.2 2-Part Text with Embedded Response Message

In the case of a TxtEmbRspMsg\_2Part message, the use of the subject line is not defined by this specification. Its MIME structure is:

```
Content-Type: multipart/mixed
  Content-Type: text/plain
  Content-Type: application/lsm-body-part
```

### 3.4 Response Message

The response to a TxtEmbRspMsg (both the TxtEmbRspMsg\_1Part and the TxtEmbRspMsg\_2Part versions) is an ERS Response Message (RspMsg).

The ERS of a `TxtEmbrspMsg` is intended to be processed by the recipient's mail reading User Agent. This allows the recipient to issue an `RspMsg` based upon the ERS if and when he or she so chooses.

The structure of a 'RspMsg' sent by the respondent's User Agent is:

```
Content-Type: text/plain
```

### 3.4.1 General Rules for Forming a Response Message

If the `TxtEmbrspMsg` has multiple recipients, the response message to the originator will "Cc:" all recipients of the `TxtEmbrspMsg`.

A `RspMsg` will include the embedded response of the user in the "Subject:" line of the message header; or in the message body; or in both.

A `RspMsg` will include the following pieces of information in its message body:

- the text message of the associated `TxtEmbrspMsg`
- the ERS of the associated `TxtEmbrspMsg` may be included in the message body in a reformatted, human readable form. This form is not defined.
- the user's response based on the ERS
- the time of the user's response

The original ERS may optionally be sent with the `RspMsg` as an **application/lsm-body-part** MIME body part.

## 3.5 Forwarded RFC 822 E-Mail Message

The structure of a 'FwdMsg' is:

```
Content-Type: message/rfc822
```

## 3.6 Binary Message

The structure of a 'BinMsg' is:

```
Content-Type: application/octet-stream
```

### 3.7 Fax Notification Message

The structure of a 'FaxNotificationMsg' is:

Content-Type: message/external-body

### 3.8 Voice Mail Notification Message

The structure of a 'VmailNotificationMsg' is:

Content-Type: message/external-body

### 3.9 Calendar Operation

An EMSD Calendar Operation Message (CalReqMsg) specifies an action on the user's calendar application. The structure of a 'CalReqMsg' is:

Content-Type: application/lsm-body-part

The details of calendar operation in this content type is expressed either as a <tdl-cal-action> in the Tagged Data List (TDL) syntax, or as <ll-cal-action> in the Lisp-List (LL) syntax. See section 5.1.2 and see section 5.1.3 for the specification of this language.

### 3.10 Address Book Operation

An EMSD Address Book Operation Message (AddrBookMsg) specifies an action on the user's address book application. The structure of a 'AddrBookMsg' is:

Content-Type: application/lsm-body-part

The details of address book operation in this content type is expressed as a <tdl-addr-action> in TDL syntax, or as <ll-addr-action> in LL syntax.

See section 5.1.2 and see section 5.1.3, for the specification of this language.

### 3.11 Configuration Request

An EMSD Configuration Request Message (ConfigReqMsg) specifies an action on the configuration application. The structure of a 'ConfigReqMsg' is:

Content-Type: application/lsm-body-part

A configuration request is expressed as a <tdl-config-action> in TDL syntax, or as a <ll-config-action> in LL syntax. See section 5.1.2 and see section 5.1.3, for the specification of this language.

### 3.12 Configuration Response

An EMSD Configuration Response message is used to report the outcome of a configuration request. The "Subject:" line of the response records the outcome while the body contains the original request.

The structure of a 'ConfigRspMsg' is:

Content-Type: application/lsm-body-part

See section 5.1.2 and see section 5.1.3, for the specification of this language.

### 3.13 Location Request ????? needed ???????

The structure of a 'LocReqMsg' is:

An EMSD Location Operation Message (LocReqMsg) specifies an action on the user's location reporting application. The structure of a 'LocReqMsg' is:

Content-Type: application/lsm-body-part

### 3.14 Location Response ????? needed ???????

The structure of a 'LocRspMsg' is:

Content-Type: application/lsm-body-part

### **3.15 Icons**

TBD.



## Chapter 4

# Registry of MIME Content Types Used by EMSD

In order to guarantee interoperability amongst all EMSD devices and create a reliable messaging environment, it is necessary that a well defined minimum set of EMSD Content Types be specified. This section enumerates those:

### 4.1 Content Types

#### EMSD MIME TYPES

Content-Type =====	Content-Subtype =====	Used in EMSD Message Type(s) =====
text	plain	TxtMsg, TxtEmbRspMsg_2Part, RspMsg
message	rfc822	FwdMsg
message	external-body	VmailNotificationMsg, FaxNotificationMsg
application	lsm-body-part (new)	AddrBookMsg, CalReqMsg, LocReqMsg, LocRspMsg ConfigReqMsg, ConfigRspMsg TxtEmbRspMsg_1Part, TxtEmbRspMsg_2Part
application	octet-stream	BinMsg
multipart	mixed	TxtEmbRspMsg_2Part

Note that the multipart mixed is only used in TxtEmbrSpMsg\_2Part and only as top-level MIME body part. EMSD does not support nested multipart messages.

## 4.2 Character Sets

Character Sets  
=====

Type	Description
----	-----
US-ASCII	the default character set

## 4.3 Content Transfer Encoding

Content Transfer Encoding  
=====

Type	Description
----	-----
7BIT	
8BIT	

The above Content Transfer Encodings apply to EMSD devices.

The message center may convert the content encoding from any valid MIME Content-Transfer-Encodings (e.g., BASE64, QUOTED-PRINTABLE) into the ones specified above.

## Chapter 5

# Specification of New MIME Content Types Used by EMSD

The MIME content types used by the various EMSD messages types include the following new subtype:

- **application/lsm-body-part**

New content-type/subtype used by EMSD will need to go through the steps outlined in RFC 1521.

### 5.1 application/lsm-body-part

This content type is defined in order to provide a flexible and growing base for incorporation of a variety of applications.

MIME type name:	APPLICATION
MIME subtype name:	EMSD-BODY-PART
Required parameters:	none
Optional parameters:	none
Encoding considerations:	tbd
Security considerations:	tbd
Published specification:	This document. See the following sections.
Person & email address to contact for further information:	Mohsen Banan <mohsen@neda.com>

### 5.1.1 The application/lsm-body-part Language

The **application/lsm-body-part** MIME subtype is used in the following EMSD messages types:

- AddrBookMsg (See section 3.10)
- CalReqMsg (See section 3.9)
- ConfigReqMsg (See section 3.11)
- ConfigRspMsg (See section 3.12)
- TxtEmbrRspMsg (See section 3.3)

For EMSD specific new content types in addition to the syntax specified for use by EMSD devices – Tagged Data List (TDL) –, an experimental temporary syntax called “Lisp-List (LL) syntax” has also been defined.

NOTE: “Lisp-List (LL) syntax” is not an integral part of this specification and future releases of this specification will not include it.

This section presents the formal language for expressing **application/lsm-body-part** content types. This content type is used to specify actions to be taken

Two syntaxes are defined for this language:

- Tagged Data List (TDL) Syntax
- Lisp-List (LL) Syntax

The content of an **application/lsm-body-part** is a **<lsm-body-part>** which can be expressed in TDL syntax as a **<tdl-lsm-body-part>**; or in LL syntax as a **<ll-lsm-body-part>**.

```
<lsm-body-part> ::=      <tdl-lsm-body-part> |
                          <ll-lsm-body-part>
```

### 5.1.2 The Tagged Data List (TDL) Syntax

In this syntax, the content of an **application/lsm-body-part** body part is expressed as an action on an object along with an optional number of attribute-value pairs. Here is an example of a calendar action:

```
object=cal&action=add&date=02/12/96&start=10a&duration=1:30&subject=re: meeting
```

which expresses:

Object: user's calendar  
 Action: add entry

Date: February 12, 1996  
 Start: 10 AM  
 Duration: 1.5 hours  
 Subject: re: meeting

The grammar for such actions is presented here:

```
<tdl-lsm-body-part> ::= <tdl-cal-action> |
                        <tdl-addr-action> |
                        <tdl-config-action> |
                        <tdl-embedded-response-spec>
```

### TDL for Calendar Operations

```
<tdl-cal-action> ::= object=cal&action=<tdl-cal-action-type><tdl-cal-attr-val-l
<tdl-cal-action-type> ::= add
<tdl-cal-attr-val-list> ::= &<tdl-cal-attr-val>[&<tdl-cal-attr-val>]*
<tdl-cal-attr-val> ::= <tdl-cal-attr>=<tdl-cal-val>
<tdl-cal-attr> ::= date=<cal-val-date> |
                   start=<cal-val-time> |
                   duration=<cal-val-dur> |
                   subject=<string>
<cal-val-date> ::= mm/dd/yy | mm/dd
<cal-val-time> ::= hh:mmA | hhA | hh:mmP | hhP
<cal-val-dur> ::= hh:mm | hh
```

### TDL for Address Book Operations

```
<tdl-addr-tdl-action> ::= object=addr&action=<tdl-addr-action-type><tdl-addr-attr-va
<tdl-addr-action-type> ::= add | update
```

```

<tdl-addr-attr-val-list> ::=      &<tdl-addr-attr-val>[&<tdl-addr-attr-val>]*
<tdl-addr-attr-val>      ::=      <tdl-addr-attr>=<tdl-addr-val>
<tdl-addr-attr>         ::=      lastname=<string> |
                                   firstname=<string> |
                                   email=<string> |
                                   phone=<string> |
                                   fax=<string>

```

### TDL for Configuration Operations

```

<tdl-config-tdl-action> ::=      object=cal&action=<tdl-config-action-type><tdl-config-attr
<tdl-config-action-type> ::=      query | update
<tdl-config-attr-val-list> ::=      &msg_ctr_pswd=<string>[&<tdl-config-attr-val>]*
<tdl-config-attr-val>   ::=      TBD

```

### TDL for Embedded Response Specifications

The embedded response feature of a TxtEmbrSpMsg allows a message to be sent to a recipient along with a limited number of possible responses. The possible responses are expressed in the embedded response specification (ERS) portion of a TxtEmbrSpMsg. See section 3.3.

The mail reading user agent of the recipient of a TxtEmbrSpMsg processes it resulting in some user-input presentation being offered the mail reader. The specific information presented and the manner of presentation is derived from the TxtEmbrSpMsg's contents.

```

<tdl-embedded-response-spec> ::=      <tdl-ers-choose-one> |
                                       <tdl-ers-choose-multi> |
                                       <tdl-ers-yes-no> |
                                       <tdl-ers-ack> |
                                       <tdl-ers-user-input>
<tdl-ers-choose-one>      ::=      object=ers&action=choose-one<tdl-ers-c-1-choices>
<tdl-ers-c-1-choices>    ::=      &choice=<ers-string>[&choice=<ers-string>]+

```

```

<tdl-ers-choose-multi> ::= object=ers&action=choose-multi<tdl-ers-c-n-choices>
<tdl-ers-c-n-choices> ::= &choice=<string>[&choice=<tdl-string>]+

<tdl-ers-yes-no> ::= object=ers&action=yes-no
<tdl-ers-ack> ::= object=ers&action=acknowledge

<tdl-ers-user-input> ::= object=ers&action=<tdl-user-input-string>
<ers-string> ::= <string> | <user-input-string>
<string> ::= " <any character sequence. cannot have %[uitdm]%" "
<user-input-string> ::= " <any character sequence,
                        with one or more %u%, %i%, %t% %d% %m%> "

%u% ::= {recipient is prompted for input to be incorporated
        into the string}
%i% ::= {recipient is prompted for an integer}
%t% ::= {recipient is prompted for time of day}
%d% ::= {recipient is prompted for day of week}
%m% ::= {recipient is prompted for date month/year}

```

### 5.1.3 The Lisp-List (LL) Syntax

In this syntax, the content of an **application/lsm-body-part** body part is expressed using a Lisp-like list notation.

The LL syntax for expressing an **application/lsm-body-part** is intended to be simple yet extensible. A grammar for the language using the LL syntax is presented here. Examples of embedded response specifications are provided in following sections which describe each type of specification (e.g., ers-choose-one) in more detail.

```

<ll-lsm-body-part> ::= <ll-cal-action> |
                       <ll-addr-action> |
                       <ll-config-action> |
                       <ll-embedded-response-spec>

```

**LL for Calendar Operations**

```
<ll-cal-action> ::= (cal-add <cal-action-plist>)
<cal-action-plist> ::= (subject <string>
                        date <cal-val-date>
                        time <cal-val-time>
                        duration <cal-val-dur>)
```

**LL for Address Book Operations**

```
<ll-addr-action> ::= (addr-add <addr-action-plist> |
                      (addr-update <addr-action-plist>))
<addr-action-plist> ::= (lastname <string>
                        firstname <string>
                        email <string>
                        phone <string>
                        fax <string>)
```

**LL for Configuration Operations**

```
<ll-config-action> ::= (config-query <config-action-plist> |
                      (config-update <config-action-plist> |
                      (msg-cntr-password <string>
                        :
                        TBD
                        :
                      )
                      )
```

**LL for Embedded Response Specifications**

```

<ll-embedded-response-spec> ::= <ll-ers-choose-one> |
                                <ll-ers-choose-multi> |
                                <ll-ers-yes-no> |
                                <ll-ers-ack> |
                                <ll-ers-user-input>

<ll-ers-choose-one> ::= (ers-choose-one (choices <ers-string> <ers-string>+))
<ll-ers-choose-multi> ::= (ers-choose-multi (choices <ers-string> <ers-string>+))
<ll-ers-yes-no-spec> ::= (ers-yes-or-no)
<ll-ers-ack> ::= (ers-acknowledge)
<ll-ers-user-input> ::= (ers-user <user-input-string>)
<ers-string> ::= <string> | <user-input-string>
<string> ::= " <any character sequence. cannot have %[uitdm]%" "
<user-input-string> ::= " <any character sequence,
                        with one or more %u%, %i%, %t% %d% %m%" "
%u% ::= {recipient is prompted for input to be incorporated
        into the string}
%i% ::= {recipient is prompted for an integer}
%t% ::= {recipient is prompted for time of day}
%d% ::= {recipient is prompted for day of week}
%m% ::= {recipient is prompted for date month/year}

```

#### 5.1.4 Calendar Operation Examples

Following is an example consider the example of an user's calendar operation in TDL syntax:

```

=====
To: Mike Jones <206.555.1212@lsm.attws.com>
Subject: Calendar update
MIME-Version: 1.0
Content-Type: application/lsm-body-part; charset=us-ascii

object=cal&action=add&date=02/12/96&start=10a&duration=1:30&subject=re: meeting

```

=====

Following is the same example as above, this time in LL syntax:

=====

To: Mike Jones <206.555.1212@lsm.attws.com>  
Subject: Address Book record for John Yin  
MIME-Version: 1.0  
Content-Type: application/lsm-body-part; charset=us-ascii

```
(cal-add (date ``02/12/96``  
          start ``10a``  
          duration ``1:30``  
          subject ``re: meeting``))
```

=====

### 5.1.5 Address Book Operation Examples

Following is an example consider the example of an Address Book operation in TDL syntax:

=====

To: Mike Jones <206.555.1212@lsm.attws.com>  
Subject: Address Book record for John Yin  
MIME-Version: 1.0  
Content-Type: application/lsm-body-part; charset=us-ascii

```
object=addr&action=update&lastname=Yin&firstname=John Z&phone=(407)9954801
```

=====

Following is the same example as above, this time in LL syntax:

=====

To: Mike Jones <206.555.1212@lsm.attws.com>  
Subject: Address Book record for John Yin  
MIME-Version: 1.0  
Content-Type: application/lsm-body-part; charset=us-ascii

```
(addr-update (lastname ``Yin``  
             firstname ``John Z``  
             phone ``(407)9954801``))
```

=====

### 5.1.6 Configuration Request/Response Examples

Attributes of configuration request messages are TBD.

### 5.1.7 Embedded Response Specification Examples

This section presents examples of the various embedded response specification (ERS) types.

#### ers-choose-one

This embedded response specification presents the user with a set of choices of which one is to be picked for the response.

#### Examples of (ers-choose-one ...) ERS

Here is an example of a `TxtEmbrSpMsg_2Part` message with an "ers-choose-one" ERS expressed in LL Syntax:

```
=====
From: John Doe <john@neda.com>
To: Jane Smith <jane@neda.com>
Subject: Fishing
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="--2754242114249"
```

This is a multi-part message in MIME format.

```
--2754242114249
Content-Transfer-Encoding: 7bit
Content-Type: text/plain; charset=us-ascii
```

I hear fishing is good in Oregon this week, would you like to join me?

```
--2754242114249
Content-Transfer-Encoding: 7bit
Content-Type: application/lsm-body-part; charset=us-ascii
```

```
(ers-choose-one (choice ``Mon' ``Wed'' ``Sorry, not this week...'))
```

```
--2754242114249--
```

Here is the same example of a `TxtEmbrSpMsg_2Part` message with an "ers-choose-one" ERS expressed in TDL Syntax:

```
=====
From: John Doe <john@neda.com>
To: Jane Smith <jane@neda.com>
Subject: Fishing
```

Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="--2754242114249"

This is a multi-part message in MIME format.

--2754242114249
Content-Transfer-Encoding: 7bit
Content-Type: text/plain; charset=us-ascii

I hear fishing is good in Oregon this week, would you like to join me?

--2754242114249
Content-Transfer-Encoding: 7bit
Content-Type: application/lsm-body-part; charset=us-ascii

object=ers&action=choose-one&choice=Mon&choice=Wed&choice=Sorry, not this week...

--2754242114249--
=====

The following is an ERS in LL syntax for an RSVP invitation to a party.

(ers-choose-one (choices "Party of %u% will attend." "Unable to attend."))

The following is the same ERS in TDL syntax.

object=ers&action=choose-one&choice=Party of %u% will attend.&choice=Unable to attend.

RspMsg for (ers-choose-one ...) ERS

=====  
From: Jane Smith <jane@neda.com>  
To: John Doe <john@neda.com>  
Subject: EMSD Response [Sorry, not this week...]  
Mime-Version: 1.0  
Content-Type: text/plain

TEXT MESSAGE:

Subject: Fishing
I hear fishing is good in Oregon this week, would you like to join me?

ERS:

(ers-choose-one (choice ``Mon' ``Wed'' ``Fri'' ``Sorry, not this week...'))

RESPONSE:

```Sorry, not this week...```

RESPONSE SENT AT:

Thu Sep 21 22:36:47 1995

=====

In the above example, if the original message was in TDL syntax, then the **ERS** would read:

`object=ers&action=choose-one&choice=Mon&choice=Wed&choice=Sorry, not this week...`

**ers-choose-multi**

This embedded response specification presents the user with a set of choices of which one or more may be picked for the response.

**Examples of (ers-choose-multi ...) ERS**

Here is an example of a `TxtEmbRspMsg_2Part` message with an "ers-choose-multi" ERS:

=====

From: John Doe <john@neda.com>  
To: Jane Smith <jane@neda.com>  
Subject: Travel Information  
Mime-Version: 1.0  
Content-Type: multipart/mixed; boundary="--2754242114249"

This is a multi-part message in MIME format.

--2754242114249  
Content-Transfer-Encoding: 7bit  
Content-Type: text/plain; charset=us-ascii

Which countries would you like travel information on?

--2754242114249  
Content-Transfer-Encoding: 7bit  
Content-Type: application/lsm-body-part; charset=us-ascii

`(ers-choose-multi (choices ``Italy`` ``Austria`` ``France``))`

--2754242114249--  
=====

The following may be an ERS for a message to find out possible meeting times:

```
(ers-choose-multi (choices ``Tue 9/26 @ 10:00 AM PDT``
                    ``Thu 9/28 @ 10:00 AM PDT``
                    ``Fri 9/29 @ 10:30 AM PDT``))
```

**RspMsg for (ers-choose-multi ...) ERS**

```
=====
From: Jane Smith <jane@neda.com>
To: John Doe <john@neda.com>
Subject: EMSD Response [``Italy``, ``Austria``]
Mime-Version: 1.0
Content-Type: text/plain
```

TEXT MESSAGE:

Subject: Which countries would you like travel information on?

ERS:

```
(ers-choose-multi (choices ``Italy`` ``Austria`` ``France``))
```

RESPONSE:

```
``Italy``, ``Austria``
```

RESPONSE SENT AT:

Thu Sep 21 22:36:47 1995

```
=====
```

**ers-yes-or-no**

This embedded response specification presents the user with a simple “yes” or “no” choice.

**Examples of (ers-yes-or-no ...) ERS**

Here is an example of a TxtEmbRspMsg\_1Part message with an “ers-yes-or-no” ERS in LL syntax:

```
=====
From: John Doe <john@neda.com>
To: Jane Smith <jane@neda.com>
Subject: Does Grandpa need a wheelchair?
```

Mime-Version: 1.0  
Content-Transfer-Encoding: 7bit  
Content-Type: application/lsm-body-part; charset=us-ascii

(ers=yes-or-no)

=====

Here is the previous example in TDL syntax:

=====

From: John Doe <john@neda.com>  
To: Jane Smith <jane@neda.com>  
Subject: Does Grandpa need a wheelchair?  
Mime-Version: 1.0  
Content-Transfer-Encoding: 7bit  
Content-Type: application/lsm-body-part; charset=us-ascii

object=ers&action=yes-no

=====

**RspMsg for (ers=yes-or-no ...) ERS**

=====

From: Jane Smith <jane@neda.com>  
To: John Doe <john@neda.com>  
Subject: EMSD Response [no]  
Mime-Version: 1.0  
Content-Type: text/plain

TEXT MESSAGE:

Subject: Does Grandpa need a wheelchair?

ERS:

(ers=yes-or-no)

RESPONSE:

no

RESPONSE SENT AT:

Thu Sep 21 22:36:47 1995

=====

**ers-acknowledge**

This embedded response specification presents the user with the opportunity to acknowledge receipt of the message.

**Examples of (ers-acknowledge ...) ERS**

Following is an example in TDL syntax.

```

=====
From: John Doe <john@neda.com>
To: Jane Smith <jane@neda.com>
Subject: The meeting on Thursday 9/28 is cancelled.
Mime-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Type: application/lsm-body-part; charset=us-ascii

object=ers&action=acknowledge
=====

```

Following is the previous example, in LL syntax.

```

=====
From: John Doe <john@neda.com>
To: Jane Smith <jane@neda.com>
Subject: The meeting on Thursday 9/28 is cancelled.
Mime-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Type: application/lsm-body-part; charset=us-ascii

(ers-acknowledge)
=====

```

**RspMsg for (ers-acknowledge ...) ERS**

```

=====
From: Jane Smith <jane@neda.com>
To: John Doe <john@neda.com>
Subject: EMSD Response [acknowledged]
Mime-Version: 1.0
Content-Type: text/plain

TEXT MESSAGE:

Subject: The meeting on Thursday 9/28 is cancelled.

```

ERS:

(ers-acknowledge)

RESPONSE:

acknowledged

RESPONSE SENT AT:

Thu Sep 21 22:36:47 1995

=====

**ers-user**

This embedded response specification presents the user with the opportunity to enter one or more text strings given a template provided in the ERS.

**Examples of (ers-user ...) ERS**

Here is an example of a TxtEmbRspMsg\_1Part message with an "ers-user" ERS:

=====

From: John Doe <john@neda.com>  
To: Jane Smith <jane@neda.com>  
Subject: Give me your social security number...  
Mime-Version: 1.0  
Content-Transfer-Encoding: 7bit  
Content-Type: application/lsm-body-part; charset=us-ascii

(ers-user "My SSN# is %u%")

=====

**RspMsg for (ers-user ...) ERS**

=====

From: Jane Smith <jane@neda.com>  
To: John Doe <john@neda.com>  
Subject: EMSD Response [My SSN# is 123-45-6789]  
Mime-Version: 1.0  
Content-Type: text/plain

TEXT MESSAGE:

DRAFT 1234

Embedded Response

February 1999

Subject: Give me your social security number...

ERS:

(ers-user ``My SSN# is %u%``)

RESPONSE:

My SSN# is 123-45-6789

RESPONSE SENT AT:

Thu Sep 21 22:36:47 1995

=====

# Chapter 6

## Issues

### 6.1 Encoding of EMSD Content-Type

Use of MIME from non-EMSD environment requires little justification.

However, because of the non-compact form of mime parameter specification it may be necessary that we define encoding definitions for MIME parameters in the EMSD environment. We propose that this be done as an optimization activity after we have gained field experience.

### 6.2 Origination Facilities

- Strategies for making the origination software widely available.



## Appendix A

# MIME Registration of Content-type/subtype Values with IANA

The following section has been reproduced from RFC 1341, [1].

Note that MIME is generally expected to be extended by subtypes. If a new fundamental top-level type is needed, its specification should be published as an RFC or submitted in a form suitable to become an RFC, and be subject to the Internet standards process.

To: IANA@isi.edu  
Subject: Registration of new MIME content-type/subtype

MIME type name:

(If the above is not an existing top-level MIME type,  
please explain why an existing type cannot be used.)

MIME subtype name:

Required parameters:

Optional parameters:

Encoding considerations:

Security considerations:

Published specification:

(The published specification must be an Internet RFC or RFC-to-be if a new top-level type is being defined, and must be a publicly available specification in any case.)

Person & email address to contact for further information:

## Appendix B

# MIME Assigned Numbers

The following section has been reproduced from RFC 1340, [4].

RFC 1340

Assigned Numbers

July 1992

### MIME TYPES

RFC-1341 [169] specifies that Content Types, Content Subtypes, Character Sets, Access Types, and Conversion values for MIME mail will be assigned and listed by the IANA.

#### Content Types and Subtypes

-----

Type	Subtype	Description	Reference
----	-----	-----	-----
text	plain richtext		[169,NSB]
multipart	mixed alternative digest parallel		[169,NSB]
message	rfc822		[169,NSB]

	partial external-body	
application	octet-stream postscript oda	[169,NSB]
image	jpeg gif	[169,NSB]
audio	basic	[169,NSB]
video	mpeg	[169,NSB]

## Character Sets

-----

Type	Description	Reference
----	-----	-----
US-ASCII	the default character set	[169,NSB]
ISO-8859-1	see ISO_8859-1:1987 below	[169,NSB]
ISO-8859-2	see ISO_8859-2:1987 below	[169,NSB]
ISO-8859-3	see ISO_8859-3:1988 below	[169,NSB]
ISO-8859-4	see ISO_8859-4:1988 below	[169,NSB]
ISO-8859-5	see ISO_8859-5:1988 below	[169,NSB]
ISO-8859-6	see ISO_8859-6:1987 below	[169,NSB]
ISO-8859-7	see ISO_8859-7:1987 below	[169,NSB]
ISO-8859-8	see ISO_8859-8:1988 below	[169,NSB]
ISO-8859-9	see ISO_8859-9:1989 below	[169,NSB]

## Access Types

-----

Type	Description	Reference
----	-----	-----
Jun FTP		[169,NSB]
ANON-FTP		[169,NSB]
TFTP		[169,NSB]
AFS		[169,NSB]
LOCAL-FILE		[169,NSB]
MAIL-SERVER		[169,NSB]

## Conversion Values

-----

Conversion values or Content Transfer Encodings.

Type	Description	Reference
------	-------------	-----------

-----  
7BIT  
8BIT  
BASE64  
BINARY  
QUOTED-PRINTABLE

-----  
[ 169 , NSB ]  
[ 169 , NSB ]  
[ 169 , NSB ]  
[ 169 , NSB ]  
[ 169 , NSB ]



# Bibliography

- [1] N. Borenstein and N. Freed. MIME (multipurpose internet mail extensions): Mechanisms for specifying and describing the format of internet message bodies. Request for Comments (Proposed Standard) 1341, Internet Engineering Task Force, June 1992. (Obsoleted by RFC1521).
- [2] D. Crocker. Standard for the format of ARPA internet text messages. Request for Comments (Standard) STD 11, 822, Internet Engineering Task Force, August 1982. (Obsoletes RFC733); (Updated by RFC987); (Updated by RFC1327).
- [3] K. Moore. Representation of Non-ASCII text in internet message headers. Request for Comments (Proposed Standard) 1342, Internet Engineering Task Force, June 1992. (Obsoleted by RFC1522).
- [4] J. Reynolds and J. Postel. Assigned numbers. Request for Comments (Standard) STD 2, 1340, Internet Engineering Task Force, July 1992. (Obsoletes RFC1060); (Obsoleted by RFC1700).