INTERNET-DRAFT
draft-nielsen-dime-02

Henrik Frystyk Nielsen
Henry Sanders
Microsoft,
Russell Butek
Simon Nash
IBM

Expires December 2002

June 17, 2002

Direct Internet Message Encapsulation (DIME)

Status of this Memo

Please send comments to the "dime@discuss.develop.com" mailing list, which is archived at "http://discuss.develop.com/dime.html".

Abstract

Direct Internet Message Encapsulation (DIME) is a lightweight, binary message format that can be used to encapsulate one or more application-defined payloads of arbitrary type and size into a single message construct. Each payload is described by a type, a length, and an optional identifier. Both URIs and MIME media type

constructs are supported as type identifiers. The payload length is
an integer indicating the number of octets of the payload. The

optional payload identifier is a URI enabling cross-referencing between payloads. DIME payloads may include nested DIME messages or chains of linked chunks of unknown length at the time the data is generated. DIME is strictly a message format: it provides no concept of a connection or of a logical circuit, nor does it address head-of-line problems.

Table of Contents

1 Introduction

Direct Internet Message Encapsulation (DIME) is a lightweight, binary message format designed to encapsulate one or more application-defined payloads into a single message construct. A DIME message contains one or more DIME records each carrying a payload of arbitrary type and up to 2^32-1 octets in size. Records can be chained together to support larger payloads. A DIME record carries three parameters for describing its payload: the payload length, the payload type, and an optional payload identifier. The purpose of these parameters is as follows:

The payload length

The payload length indicates the number of octets in the payload (see section 2.3.1). By providing the payload length within the first 12 octets of a record, efficient record boundary detection is possible.

The payload type

The DIME payload type identifier indicates the type of the payload. DIME supports both URIs [10] as well as MIME media type constructs [7] as type identifiers (see section 2.3.2). By indicating the type of a payload, it is possible to dispatch the payload to the appropriate user application.

The payload identifier

A payload may be given an optional identifier in the form of an absolute or relative URI (see section 2.3.3). The use of an identifier enables payloads that support URI linking technologies to cross-reference other payloads.

In addition, each record contains a version number (see section 2.2) and a slot for optional data in the form of option elements (see section 2.4).

1.1 Notational 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 [9].

1.2 Conformance Requirement

An implementation is not DIME compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements defined in this specification. A DIME implementation MUST be conformant in order to parse or generate a DIME message defined by this specification.

1.3 Design Goals

Because of the large number of existing message encapsulation formats, record marking protocols and multiplexing protocols, it is best to be explicit about the design goals of DIME and, in particular, about what is outside the scope of DIME.

The design goal of DIME is to provide an efficient and simple message format that can accommodate the following:

1. Encapsulating arbitrary documents and entities, including encrypted data, XML documents, XML fragments, image data like GIF and JPEG files, etc.

2. Encapsulating documents and entities initially of unknown size. This capability can be used to encapsulate dynamically generated content or very large entities as a series of chunks.

3. Aggregating multiple documents and entities that are logically associated in some manner into a single message. For example, DIME can be used to encapsulate a SOAP message and a set of attachments referenced from that SOAP message.

In order to achieve efficiency and simplicity, the mechanisms provided by this specification have been deliberately limited to serve these purposes. DIME has not been designed as a general message description or document format such as MIME or XML.

Instead, DIME-based applications can take advantage of such formats by encapsulating them in DIME messages.

The following list identifies what is outside the scope of DIME:

1. DIME does not make any assumptions about the types of payloads that are carried within DIME messages or about the message exchange patterns of such messages.

2. DIME does not in any way introduce the notion of a connection or of a logical circuit (virtual or otherwise).

3. DIME does not attempt to deal with head-of-line blocking problems that might occur when using stream-oriented protocols like TCP.

1.4 DIME Terminology

DIME message

The basic message construct defined by this specification. A DIME message contains one or more DIME records (see section 2.1.1).

DIME record

A DIME record contains a payload described by a type, a length, and an optional identifier (see section 2.1.2).

DIME record chunk

A DIME record that has been marked as containing a chunk of a payload rather than a full payload (see section 2.1.3).

DIME version

A number used to identify the format of a DIME record (see section 2.2).

DIME payload

The data carried within a DIME record defined by a user application.

DIME chunked payload

A payload that has been partitioned into multiple DIME record chunks. This can be used to carry dynamically generated content or very large entities that don't fit into a single DIME record (see section 2.1.3).

DIME payload length

The size of the payload indicated in number of octets (see section 2.3.1).

DIME payload type

An identifier that indicates the type of the payload. This specification supports both URIs [10] as well as MIME media type constructs [11] as type identifiers (see section 2.3.2).

DIME payload identifier

A URI that optionally can be used to identify a payload (see section 2.3.3).

DIME options

A DIME record might contain zero or more optional pieces of data in the form of DIME option elements. This can be used to carry additional information about the payload or information which otherwise may be of benefit to the DIME parser parsing the DIME message (see section 2.4).

DIME option element

An optional piece of data that may be carried in a DIME record as part of the DIME options (see section 2.4).

DIME user application

The logical, higher-layer application that uses DIME for encapsulating messages.

DIME generator

An entity or module that encapsulates user application-defined payloads within DIME messages.

DIME parser

An entity or module that parses DIME messages and hands off the payloads to a DIME user application.

1.5 Intended Usage

The intended usage of DIME is as follows: A user application wants to encapsulate one or more related documents into a single DIME message. For example, this can be a SOAP message along with a set of attachments. The DIME generator encapsulates each document in DIME records as payload or chunked payload, indicating the type and length of the payload along with an optional identifier. The DIME records are then put together to form a single DIME message. The DIME parser deconstructs the DIME message and hands the payloads to a (potentially different) user application.

DIME can be used in combination with most protocols that support the exchange of binary data as long as the DIME message can be exchanged in its entirety. A DIME message can be carried as a MIME entity using the media type "application/dime" (see section 6 for IANA media type registration of "application/dime").

DIME records can encapsulate documents of any type. It is possible to carry MIME messages in DIME records by using a media type such as "message/rfc822". A DIME message can be encapsulated in a DIME record by using the media type "application/dime" (see section 6).

It is important to note that although MIME entities are supported, there are no assumptions in DIME that a record payload is MIME; DIME makes no assumption concerning the type of the payloads carried in a DIME message.

DIME provides no support for error handling. It is up to the DIME parser to determine the implications of receiving a malformed DIME message not conforming to this specification (see section 2.2 for a description of DIME version numbers). It is the responsibility of the user applications involved to provide any additional functionality such as QoS that they may need as part of the overall system in which they participate.

2 The DIME Mechanisms

This section describes the mechanisms used in DIME. The specific syntax for these mechanisms is defined in section 3.

2.1 DIME Encapsulation Constructs

2.1.1 Message

A DIME message is composed of one or more DIME records. The first record in a message is marked with the MB (Message Begin) flag set and the last record in the message is marked with the ME (Message End) flag set (see section 3.2.1 and 3.2.3). The minimum message length is one record which is achieved by setting both the MB and the ME flag in the same record. Note that at least two record chunks are required in order to encode a chunked payload (see section 2.1.3). The maximum number of DIME records that can be carried in a DIME message is unbounded.

DIME messages MUST NOT overlap; that is, the MB and the ME flags MUST NOT be used to nest DIME messages. DIME messages can be nested by carrying a full DIME message within a DIME record with the type "application/dime" (see section 6).

```
<--------------------- DIME message ---------------------->
+---------+     +---------+     +---------+     +---------+
| R1 MB=1 | ... | Rr      | ... | Rs      | ... | Rt ME=1 |
+---------+     +---------+     +---------+     +---------+
```

Figure 1: Example of a DIME message with a set of records. The message head is to the left and the tail to the right, with the logical record indexes t > s > r > 1. The MB (Message Begin) flag is set in the first record (index 1) and the ME (Message End) flag is set in the last record (index t).

Note that actual DIME records do not carry an index number; the ordering is implicitly given by the order in which the records are serialized.

2.1.2 Record

A record is the unit for carrying a payload within a DIME message. Each payload is described by its own set of parameters (see section 2.3).

2.1.3 Record Chunks

A record chunk is a DIME record that contains a chunk of a payload. Chunked payloads can be used to partition dynamically generated content or very large entities into multiple subsequent record chunks serialized within the same DIME message.

Chunking is not a mechanism for introducing multiplexing into DIME. It is a mechanism to limit the need for outbound buffering on the generating side. This is similar to the message chunking mechanism defined in HTTP/1.1 [11].

A DIME message can contain zero or more chunked payloads. A chunked payload is encoded as an initial record chunk followed by zero or more middle record chunks followed by a terminating record chunk. Each record chunk is encoded using the following encoding rules:

1. The initial record chunk is a DIME record with the CF (Chunk Flag) flag set (see section 3.2.4). The type of the entire chunked payload MUST be indicated in the TYPE field regardless of whether the DATA_LENGTH field value is zero or not. The ID field MAY be used to carry an identifier of the entire chunked payload. The DATA_LENGTH field indicates the size of the data carried in the DATA field (see section 2.3.1).

2. Each middle record chunk is a DIME record with the CF flag set indicating that this record chunk contains the next chunk of data of the same type and with the same identifier as the initial record chunk. The value of the TYPE_LENGTH and the ID_LENGTH fields MUST be zero and the TYPE_T field value MUST be 0x00 (see section 3.2.8). The DATA_LENGTH field indicates the size of the data carried in the DATA field (see section 2.3.1).

3. The terminating record chunk is a DIME record with the CF flag cleared indicating that this record chunk contains the last

chunk of data of the same type and with the same identifier as the initial record chunk. As with the middle record chunks, the value of the TYPE_LENGTH and the ID_LENGTH fields MUST be zero and the TYPE_T field value MUST be 0x00 (see section

3.2.8). The DATA_LENGTH field indicates the size of the data carried in DATA field (see section 2.3.1).

A chunked payload MUST be entirely encapsulated within a single DIME message. That is, a chunked payload MUST NOT span multiple DIME messages. As a result, neither an initial nor a middle record chunk can have the ME (Message End) flag set.

2.2 DIME Version Number

A DIME record contains a version number that indicates the format of the record. The DIME version number is incremented when the format of a DIME message is changed. Version numbers are considered to be "major" rather than "minor". That is, there is no assumption of compatibility between any two versions.

All DIME records in a DIME message including record chunks MUST be of the same version. A DIME parser encountering different DIME version numbers in different DIME records in the same DIME message MUST discard that message as faulty.

In order to parse a DIME record of a given version, a DIME parser MUST be compliant with that version (see section 1.2). A DIME implementation MUST NOT attempt to parse or generate a DIME record with a version that the implementation does not comply with. A DIME implementation MAY but NEED NOT support multiple DIME versions.

This document defines version 1 (0x01) (see section 3.2.1). Any new version of DIME MUST be published as a standard-track RFC following IETF consensus.

2.3 DIME Payload Description

Each record contains information about the payload carried within it. This section introduces the mechanisms by which these payloads are described.

2.3.1 Payload Length

Regardless of the relationship of a record to other records, the payload length always indicates the length of the payload encapsulated in THIS record. The length of the payload is indicated in number of octets in the DATA_LENGTH field (see section 3.2.10). Note that zero is a valid length.

2.3.2 Payload Type

The payload type of a record indicates the kind of data being carried in the payload of that record. This may be used to guide the processing of the payload at the discretion of the user application. The type of the first record, by convention, provides the processing context not only for the first record but for the whole DIME message. Additional context for processing the message may be provided by the transport service port (TCP, UDP, etc) at which the message was received and by other communication parameters.

It is important to emphasize that DIME mandates no specific processing model for DIME messages. The usage of the payload types is entirely at the discretion of the user application. The comments regarding usage above should be taken as guidelines for building processing conventions, including mappings of higher level application semantics onto DIME.

The structure and format of the TYPE field value is indicated using the TYPE_T field (see section 3.2.5). This specification supports TYPE field values in the form of absolute URIs and MIME media type constructs. The former allows for decentralized control of the value space and the latter allows DIME to take advantage of the already very large and successful media type value space maintained by IANA [3].

The media type registration process is outlined in RFC 2048 [8]. Use of non-registered media types is discouraged. The URI scheme registration process is described in RFC 2717 [13]. It is recommended that only well-known URI schemes registered by IANA be used (see [17] for a current list).

URIs can be used for message types that are defined by URIs. Records that carry a payload with an XML-based message type MAY use the XML namespace identifier of the root element as the TYPE field value. A SOAP/1.1 message, for example, may be represented by the URI

http://schemas.xmlsoap.org/soap/envelope/

Records that carry a payload with an existing, registered media type SHOULD carry a TYPE field value of that media type. Note that the TYPE field indicates the type of the payload; it does NOT refer to a MIME message that contains an entity of the given type. For example, the media type image/jpeg

indicates that the payload is a JPEG image. Similarly, the media type

message/http

indicates that the payload is an HTTP message as defined by RFC 2616 [11]. A value of

application/xml; charset="utf-16"

indicates that the payload is an XML document as defined by RFC 3023 [16].

2.3.3 Payload Identification

The optional payload identifier allows user applications to identify a payload within a DIME record. By providing a payload identifier, it becomes possible for other payloads supporting URI-based linking technologies to refer to that payload. DIME does not mandate any particular linking mechanism but leaves this to the user application to define in the language it prefers.

It is important that payload identifiers are maintained so that references to those payloads are not broken. If records are repackaged, for example, by an intermediate application, then that application MUST ensure that the payload identifiers are preserved.

2.4 DIME Options

DIME has provisions for carrying additional information in a DIME record as option elements. A DIME record (including a record chunk) can carry zero or more such option elements each containing information about the payload or information which otherwise may be of benefit to a DIME parser.

An option element contains two parameters describing its contents: a type and a length. The meaning of these parameters is as follows:

- The option element type indicates the structure and format of the data carried in that element (see section 3.2.11).

o The option element length indicates the size of the data carried in that element in number of octets (see section 3.2.11).

The structure and format of each element is entirely determined by the option element type. This specification does not define any option element types. DIME option elements are defined in a centralized manner controlled by IANA (see section 6.2 for IANA guidelines).

Option elements are set on a per DIME record basis. A DIME generator MAY generate different option elements for different DIME records in the same DIME message. Use of option data is OPTIONAL by DIME generators.

DIME option element types are defined independently of each other; support for an element type does not imply support for other element types. That is, a DIME parser that recognizes option element type 5 might not recognize type 4 or 6.

A DIME parser conforming to this specification MAY but NEED NOT support any option element types. A DIME parser SHOULD ignore unrecognized option element types.

3 The DIME Specifications

3.1 Data Transmission Order

The order of transmission of the DIME record described in this document is resolved to the octet level. For diagrams showing a group of octets, the order of transmission of those octets is first left to right and then top to bottom, as they are read in English. For example, in the diagram in Figure 2, the octets are transmitted in the order they are numbered.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Octet 1    |    Octet 2    |    Octet 3    |    Octet 4    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Octet 5    |    Octet 6    |    Octet 7    |    Octet 8    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: DIME octet ordering

Whenever an octet represents a numeric quantity, the leftmost bit in the diagram is the high order or most significant bit. That is, the bit labeled 0 is the most significant bit.

For each multi-octet field representing a numeric quantity defined by DIME, the leftmost bit of the whole field is the most significant bit. Such quantities are transmitted in a big-endian manner with the most significant octet transmitted first.

3.2 Record Layout

DIME records are variable length records with a common format illustrated in Figure 3. In the following sections, the individual record fields are described in more detail.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         |M|M|C|       |       |                               |
| VERSION |B|E|F| TYPE_T| RESRVD|         OPTIONS_LENGTH        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ID_LENGTH          |           TYPE_LENGTH         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          DATA_LENGTH                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               /
/                     OPTIONS + PADDING                         /
/                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               /
/                          ID + PADDING                         /
/                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               /
/                        TYPE + PADDING                         /
/                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               /
/                        DATA + PADDING                         /
/                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: DIME Record Layout. The use of "/" indicates a field length which is a multiple of 4 octets.

3.2.1 Version

An unsigned 5-bit integer that indicates the format of the DIME record (see section 2.2). This document defines version 1 (0x01). A DIME generator conforming to this specification MUST generate DIME messages with a VERSION field value of 0x01. A DIME parser conforming to this specification MUST verify that the VERSION field has a value of 0x01.

3.2.2 MB (Message Begin)

The MB flag is a 1 bit field that when set indicates the start of a DIME message (see section 2.1.1).

3.2.3 ME (Message End)

The ME flag is a 1 bit field that when set indicates the end of a DIME message (see section 2.1.1). Note, that in case of a chunked payload, the ME flag is set only in the terminating record chunk of the last chunked payload (see section 2.1.3).

3.2.4 CF (Chunk Flag)

The CF flag is a 1 bit field indicating that this is either the first record chunk or a middle record chunk of a chunked payload (see section 2.1.3 for a description of how to encode a chunked payload).

3.2.5 TYPE_T

The TYPE_T field value indicates the structure and format of the value of the TYPE field (see section 2.3.2 for a description of the TYPE field and section 4 for a description of internationalization issues related to the TYPE field). The TYPE_T field is a 4 bit field with values defined in Table 1:

TYPE_T	Value
Unchanged (see section 2.1.3)	0x00
media-type as defined in RFC 2616 [11]	0x01
absoluteURI as defined in RFC 2396 [10]	0x02
Unknown	0x03
None	0x04
Reserved	0x05-0x0F

Table 1: DIME TYPE_T field values.

The value 0x00 (Unchanged) MUST be used in all middle record chunks and terminating record chunks used in chunked payloads (see section 2.1.3). It MUST NOT be used in any other record. When used, the TYPE_LENGTH field value MUST be zero.

The value 0x01 (media-type) indicates that the TYPE field contains a value that follows the "media-type" BNF construct defined by RFC 2616 [11] (see section 2.3.2).

The value 0x02 (absoluteURI) indicates that the TYPE field contains a value that follows the "absoluteURI" BNF construct defined by RFC 2396 [10] (see section 2.3.2).

The value 0x03 (Unknown) SHOULD be used to indicate that the type of the payload is unknown. This is similar to the "application/octet-stream" media type defined by MIME [7]. When used, the TYPE_LENGTH field value MUST be zero. Regarding implementation, it is RECOMMENDED that a DIME parser receiving a DIME record of this type provides a mechanism for storing but not processing the payload (see section 5).

The value 0x04 (None) indicates that there is no type or payload

associated with this record. When used, the value of the
TYPE_LENGTH and the DATA_LENGTH fields MUST be zero. This TYPE_T

value can be used whenever an empty record is needed, for example in order to terminate a DIME message in cases where there is no payload defined by the user application.

There is no default value for the TYPE_T field. Reserved TYPE_T field values are for future use and MUST NOT be used. A DIME parser that receives a DIME record with an unknown TYPE_T field value SHOULD treat the payload as if it had been marked with a value of 0x03 (Unknown). Note, that in this case the TYPE_LENGTH is not required to be zero.

3.2.6 RESRVD

The RESRVD field is reserved for future use and MUST be set to 0x00. A DIME parser that receives a DIME record with a RESRVD field value other than 0x00 MUST discard that message as faulty.

3.2.7 OPTIONS_LENGTH

An unsigned 16 bit integer that specifies the length in octets of the OPTIONS field excluding any padding used to achieve a 4 octet alignment of the OPTIONS field (see section 2.4).

3.2.8 ID_LENGTH

An unsigned 16 bit integer that specifies the length in octets of the ID field excluding any padding used to achieve a 4 octet alignment of the ID field (see section 2.3.3).

3.2.9 TYPE_LENGTH

An unsigned 16 bit integer that specifies the length in octets of the TYPE field excluding any padding used to achieve a 4 octet alignment of the TYPE field (see section 2.3.2).

3.2.10 DATA_LENGTH

The DATA_LENGTH field is an unsigned 32 bit integer that specifies the length in octets of the DATA field excluding any padding used to achieve a 4 octet alignment of the DATA field (see section

2.3.1).

A payload size of 0 octets is allowed. Payloads larger than 2^32-1 octets can be accommodated by using chunked payloads (see section 2.1.3).

3.2.11 OPTIONS

The OPTIONS field contains 0 or more option elements where each element follows the layout in Figure 4 (see section 2.4 for a description of option elements):

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ELEMENT_T          |         ELEMENT_LENGTH        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               /
/                         ELEMENT_DATA                          /
/                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: DIME option element layout. The use of "/" indicates a field length which is a multiple of 4 octets.

All DIME records MAY have a non-zero OPTIONS field. A DIME parser receiving a DIME record with an unrecognized option element type SHOULD ignore that element (see section 6.2 for IANA guidelines for registration of new option element types).

The length of each element does not have to be a multiple of 4 octets and there is no padding between elements. However, the size of the OPTIONS field MUST be a multiple of 4 octets. If the length of all the elements is not a multiple of 4 octets, the generator MUST pad the OPTIONS field value with all zero octets. Padding is not included in the OPTIONS_LENGTH field (see section 3.2.7).

A DIME generator MUST NOT pad the OPTIONS field with more than 3 octets. A DIME parser MUST ignore the padding octets.

3.2.12 ID

The value of the ID field is an identifier in the form of a URI

[10] (see section 2.3.3 and 3.3). The required uniqueness of the message identifier is guaranteed by the generator. The URI can be either relative or absolute; DIME does not define a base URI which

means that user applications using relative URIs MUST provide an actual or a virtual base URI (see [10]).

With the exception of subsequent record chunks (see section 2.1.3), all records MAY have a non-zero ID field.

The length of the ID field MUST be a multiple of 4 octets. If the length of the payload id value is not a multiple of 4 octets, the generator MUST pad the value with all zero octets. Padding is not included in the ID_LENGTH field (see section 3.2.8).

A DIME generator MUST NOT pad the ID field with more than 3 octets. A DIME parser MUST ignore the padding octets.

3.2.13 TYPE

The value of the TYPE field is an identifier describing the type of the payload (see section 2.3.2). The value of the TYPE field MUST follow the structure implied by the value of the TYPE_T field (see section 3.2.5).

The length of the TYPE field MUST be a multiple of 4 octets. If the length of the payload type value is not a multiple of 4 octets, the generator MUST pad the value with all zero octets. Padding is not included in the TYPE_LENGTH field (see section 3.2.9).

A DIME generator MUST NOT pad the TYPE field with more than 3 octets. A DIME parser MUST ignore the padding octets.

A DIME parser receiving a DIME record with a known TYPE_T field value but an unknown TYPE field value SHOULD interpret the type identifier of that record as if the TYPE_T field value was 0x03 (Unknown).

It is STRONGLY RECOMMENDED that the identifier be globally unique and maintained with stable and well-defined semantics over time.

3.2.14 DATA

The DATA field carries the payload intended for the DIME user application. Any internal structure of the data carried within the DATA field is opaque to DIME.

The length of the DATA field MUST be a multiple of 4 octets. If the length of the payload is not a multiple of 4 octets, the generator MUST pad the value with all zero octets. Padding is not included in the DATA_LENGTH field (see section 3.2.10).

A DIME generator MUST NOT pad the DATA field with more than 3 octets. A DIME parser MUST ignore the padding octets.

3.3 Use of URIs in DIME

DIME uses URIs [10] for some identifiers. To DIME, a URI is simply a formatted string that identifiesûvia name, location, or any other characteristicûa resource on the Web.

The use of IP addresses in URIs SHOULD be avoided whenever possible (see RFC 1900 [5]). However, when used, the literal format for Ipv6 addresses in URIs as described by RFC 2732 [15] SHOULD be supported.

DIME does not define any equivalence rules for URIs in general as these are defined by the individual URI schemes and by RFC 2396 [10]. However, because of inconsistencies with respect to some URI equivalence rules in many current URI parsers, it is STRONGLY RECOMMENDED that generators of DIME messages only rely on the most rudimentary equivalence rules defined by RFC 2396.

The size of URIs used as values in the ID field and the TYPE field is limited by the maximum size of these fields which is 2^16-1 octets. DIME parsers and generators MUST be able to deal with URIs of this size.

4 Internationalization Considerations

Identifiers used in DIME such as URIs and MIME media type constructs provide different levels of support for internationalization. It is STRONGLY RECOMMENDED that the definitions and guidelines for internationalization support of these values be followed when used in DIME. In particular, the following fields require special attention:

o For the ID field, implementers are referred to RFC 2718 [14] for internationalization considerations of URIs.

- For a TYPE_T value of 0x01 (media types), implementers are referred to RFC 2046 [7] for internationalization considerations of MIME media types.

- For a TYPE_T value of 0x02 (absolute URI), implementers are referred to RFC 2718 [14] for internationalization considerations of URIs.

For ELEMENT_T values and TYPE_T values not defined by this specification, implementers are referred to the documentation of such features for specific internationalization considerations.

5 Security Considerations

Implementers should pay special attention to the security implications of any record types that can cause the remote execution of any actions in the recipient's environment. Before accepting records of any type, an application should be aware of the particular security implications associated with that type.

Security considerations for media types in general are discussed in RFC 2048 [8] and in the context of the "application/postscript" and the "message/external-body" media type in RFC 2046 [7].

> Note: This specification does not presently define any mechanisms for providing security for DIME messages and header information. Future revisions of this specification will address this open issue.

6 IANA Considerations

This draft describes a new media type, "application/dime" for which section 6.1 contains a registration application following the guidelines in RFC 2048 [8].

Section 6.2 contains guidelines for definition and registration of additional DIME options (see section 2.4).

6.1 Media Type Registration: application/dime

MIME media type name: application

MIME subtype name: dime

Required parameters: none

Optional parameters: none

Encoding considerations:

This media type MAY be encoded as appropriate for the charset and the capabilities of the underlying MIME transport. For 7-bit transports, data using 8-bit or higher MUST be encoded in quoted-printable or base64 content-transfer-encodings. For 8-bit clean transport (e.g., 8BITMIME [2] ESMTP [4] or NNTP [1]), 8-bit data such as UTF-8 does not need to be encoded. Over HTTP [11], no content-transfer-encoding is necessary regardless of the encoding.

Security considerations: See section 5

Interoperability considerations: n/a

Published Specification: this specification

Applications which use this media type:

Applications that choose to use DIME as the packaging mechanism for encapsulating one or more application-defined payloads of arbitrary type and size into a single message construct.

Additional information: none

Magic number(s): none

File extension(s):

.dim
.dime

Macintosh File Type Code(s):

DIME

Person and email address for further information: see section 10

Intended usage:

COMMON

Author/Change controller:

The DIME specification is an individual Internet Draft submission. It is not the product of an IETF Working Group. The IETF has change control over the DIME specification.

6.2 Guidelines for Registration of DIME Option Element Types

The registration process of DIME option element types follows the guidelines for "IETF Consensus" as defined in RFC 2434 [11] where new ELEMENT_T values are assigned through the IETF consensus process. Specifically, new assignments are made via RFCs approved by the IESG. Typically, the IESG will seek input on prospective assignments from appropriate persons (e.g., a relevant Working Group if one exists).

The following process is designed to ensure that new DIME option elements are reviewed for technical correctness and appropriateness and that their description is complete and published before an ELEMENT_T value is assigned by IANA.

1. The author(s) document(s) the option element, leaving the ELEMENT_T value as "To Be Determined" (TBD). It is important that security and internationalization concerns for the option element be addressed. It is STRONGLY RECOMMENED that the documentation be published as an Internet Draft.

2. The author(s) submit(s) the Internet Draft for review by the IESG and any relevant working groups (IETF or otherwise).

3. The specification of the new option element is reviewed by the IESG, the IETF, and other relevant groups identified in 2). If

the option element is accepted for inclusion in the DIME specification, the specification of the option is published as either a standards-track or a non-standards-track RFC.

4. At the time of publication as an RFC, IANA assigns a DIME ELEMENT_T value for the new option element. The option is not to be used in published implementations before IANA has assigned an ELEMENT_T value.

7 Intellectual Property

The following notice is copied from RFC 2026 [6], Section 10.4, and describes the position of the IETF concerning intellectual property claims made against this document.

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use other technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the procedures of the IETF with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

8 Acknowledgements

Special thanks go to Paul H. Gleichauf and Krishna Sankar of Cisco for their input on this specification.

9 References

[1] B. Kantor, P. Lapsley, "Network News Transfer Protocol", RFC 977, U.C. San Diego, U.C. Berkeley, February 1986

[2] J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker, "SMTP Service Extension for 8bit-MIMEtransport", RFC 1652, MCI, Innosoft, Dover Beach Consulting, Inc., Network Management Associates, Inc., Silicon Graphics, Inc., July 1994.

[3] Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.

[4] J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker, " SMTP Service Extensions", RFC 1869, MCI, Innosoft International, Inc., Dover Beach Consulting, Inc., Network Management Associates, Inc., Brandenburg Consulting, Inc., November 1995.

[5] B. Carpenter, Y. Rekhter, "Renumbering Needs Work", RFC 1900, IAB, February 1996

[6] S. Bradner, "The Internet Standards Process û Revision 3", RFC 2026, Harvard University, October 1996

[7] N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types" RFC 2046, Innosoft First Virtual, November 1996

[8] N. Freed, J. Klensin, J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", RFC 2048, Innosoft, MCI, ISI, November 1996

[9] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997

[10] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

[11] T. Narten, H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[12] R. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, T. Berners-Lee, "Hypertext Transfer Protocol û HTTP/1.1", RFC 2616, U.C. Irvine, DEC W3C/MIT, DEC, W3C/MIT, W3C/MIT, January 1997

[13] R. Petke, I. King, "Registration Procedures for URL Scheme Names", BCP: 35, RFC 2717, UUNET Technologies, Microsoft Corporation, November 1999

[14] L. Masinter, H. Alvestrand, D. Zigmond, R. Petke, "Guidelines for new URL Schemes", RFC 2718, Xerox Corporation, Maxware, Pirsenteret, WebTV Networks, Inc., UUNET Technologies, November 1999

[15] R. Hinden, B. Carpenter, L. Masinter, "Format for Literal Ipv6 Addresses in URL's", RFC 2732, Nokia, IBM, AT&T, December 1999

[16] M. Murata, S. St.Laurent, D. Kohn, "XML Media Types" RFC 3023, IBM Tokyo Research Laboratory, simonstl.com, Skymoon Ventures, January 2001

[17] List of Uniform Resource Identifier (URI) schemes registered

by IANA is available at
"http://www.iana.org/assignments/uri-schemes"

10 Authors' Addresses

Henrik Frystyk Nielsen
Microsoft
One Microsoft Way, Redmond, WA 90852
Email: henrikn@microsoft.com

Henry Sanders
Microsoft
One Microsoft Way, Redmond, WA 90852
Email: henrysa@microsoft.com

Russell Butek
IBM
11501 Burnet Road, Austin, TX 78758
Email: butek@us.ibm.com

Simon Nash
IBM
Hursley Park, Winchester, UK
Email: nash@hursley.ibm.com