INTERNET-DRAFT
draft-nielsen-dime-00

Expires May 2002

Henrik Frystyk Nielsen
Henry Sanders
Erik Christensen
Microsoft

November 2001

Direct Internet Message Encapsulation (DIME)

Status of this Memo

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, provides no concept of a connection or of a logical circuit, and does not address head-of-line problems.

Table of Contents

1 Introduction....3
1.1 Notational Conventions....3
1.2 Design Goals....4
1.3 DIME Terminology....5
1.4 Intended Usage....6

2 The DIME Mechanisms....7
2.1 DIME Encapsulation Constructs....7
2.1.1 Message....7
2.1.2 Record....8
2.1.3 Chunked Records....8
2.2 DIME Payload Description....9
2.2.1 Payload Length....9
2.2.2 Payload Type....9
2.2.3 Payload Identification....10

3 The DIME Specifications....11
3.1 Data Transmission Order....11
3.2 Record Layout....11
3.2.1 MB (Message Begin)....12
3.2.2 ME (Message End)....12
3.2.3 CF (Chunked Flag)....12
3.2.4 ID_LENGTH....12
3.2.5 TNF (Type Name Format)....13
3.2.6 TYPE_LENGTH....13
3.2.7 DATA_LENGTH....13
3.2.8 ID....14
3.2.9 TYPE....14
3.2.10 DATA....15
3.3 Use of URIs in DIME....15

4 Internationalization Considerations....15

5 Security Considerations....16

6 IANA Considerations....16

7 Intellectual Property....16

8 Acknowledgements....17

9 References...17

10 Authors' Addresses.......................................18

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 2.2.1). By providing the payload length within the first 8 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 [8] as well as MIME media type constructs [5] as type identifiers (see section 2.2.2). Based on the indicated type of a payload, it is possible to dispatch the payload to the appropriate user application.

The payload identifier

A payload can optionally be given an identifier in the form of an absolute or relative URI (see section 2.2.3). This enables payloads that support URI linking technologies to cross-reference other payloads.

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 [5].

1.2 Design Goals

Because of the large number of existing message encapsulation formats, record marking protocols and multiplexing protocols, we would like 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 explicitly 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 must 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 must not in any way introduce the notion of a connection or of a logical circuit (virtual or otherwise).

3. DIME must not attempt to deal with head-of-line blocking

problems that might occur when using stream-oriented protocols like TCP.

1.3 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 chunked record

A DIME record that contains parts of dynamically generated content or very large entities that have been partitioned into multiple DIME records within the same DIME message (see section 2.1.3).

DIME payload

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

DIME payload length

The size of the payload indicated in number of octets (see section 2.2.1).

DIME payload type

An identifier that indicates the type of the payload. This specification supports both URIs [8] as well as MIME media type constructs [5] as type identifiers (see section 2.2.2).

DIME payload identifier

A URI that can optionally be used to identify a payload (see section 2.2.3)

DIME user application

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

DIME generator

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

DIME parser

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

1.4 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 within a DIME record, indicating the type and length of the payload along with an optional identifier. These 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 considerations of "application/dime").

DIME records can encapsulate any message 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. 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 3.2.2). The minimum message length is one record which is achieved by setting both the MB and the ME flag in the same record. There is no maximum number of DIME records that can be carried within a single DIME message.

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).

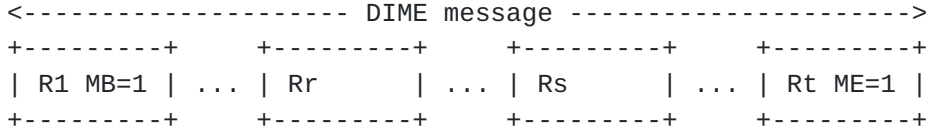

Figure 1: Example of a DIME message with a set of records. The message head is to the left and the tail to the right with the logical record indexes t > s > r > 1. The MB (Message Begin) flag is set in the first record (index 1) and the ME (Message End) flag is set in the last record (index t).

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 2.2).

2.1.3 Chunked Records

Chunked records can be used to partition dynamically generated content or very large entities into multiple subsequent DIME records 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 [9].

A DIME message can contain zero or more chunked record series. A chunked record series contains an initial record followed by zero or more middle records followed by a terminating record. The encoding rules are as follows:

1. The initial record is indicated by having the CF (Chunked Flag) flag set (see section 3.2.3). The type of the payload MUST be indicated in the TYPE field regardless of whether the DATA_LENGTH field value is zero or not. The DATA_LENGTH field indicates the size of THIS record's DATA field (see section 2.2.1). The ID field MAY be used to carry an identifier of the payload.

2. Each middle record is marked with the CF flag set indicating that this record contains the next chunk of data of the same type and (if provided) with the same identifier (see section 3.2.3). The value of the TYPE_LENGTH and the ID_LENGTH fields MUST be zero and the TNF (Type Name Format) field MUST be set to 0x00 (see section 3.2.4). The DATA_LENGTH field indicates the size of THIS record's DATA field (see section 2.2.1).

3. The terminating record in a chunked record series is indicated by having the CF flag cleared. Again, the value of the TYPE_LENGTH and the ID_LENGTH fields MUST be 0 and the TNF (Type Name Format) field MUST be set to 0x00 (see section

3.2.4). The DATA_LENGTH field indicates the size of THIS record's DATA field (see section 2.2.1).

A chunked record series MUST be entirely encapsulated within a single DIME message. That is, a chunked record series MUST NOT span multiple DIME messages.

2.2 DIME Payload Description

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

2.2.1 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.

2.2.2 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 format of the TYPE field value is indicated using the TNF (Type Name Format) 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 [2].

The media type registration process is outlined in RFC 2048 [6]. Use of non-registered media types is discouraged. The URI scheme registration process is described in RFC 2717 [10]. It is recommended that only well-known URI schemes registered by IANA be used (see [14] 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 [9]. A value of

application/xml; charset="utf-16"

indicates that the payload is an XML document as defined by RFC 3023 [13].

2.2.3 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 this in the language that 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.

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 left to right, 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.

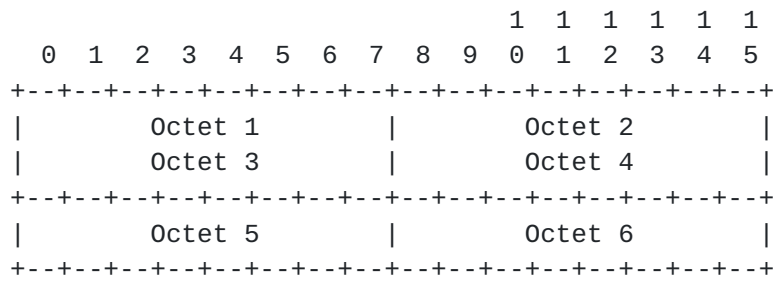

Figure 2: DIME octet ordering

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.

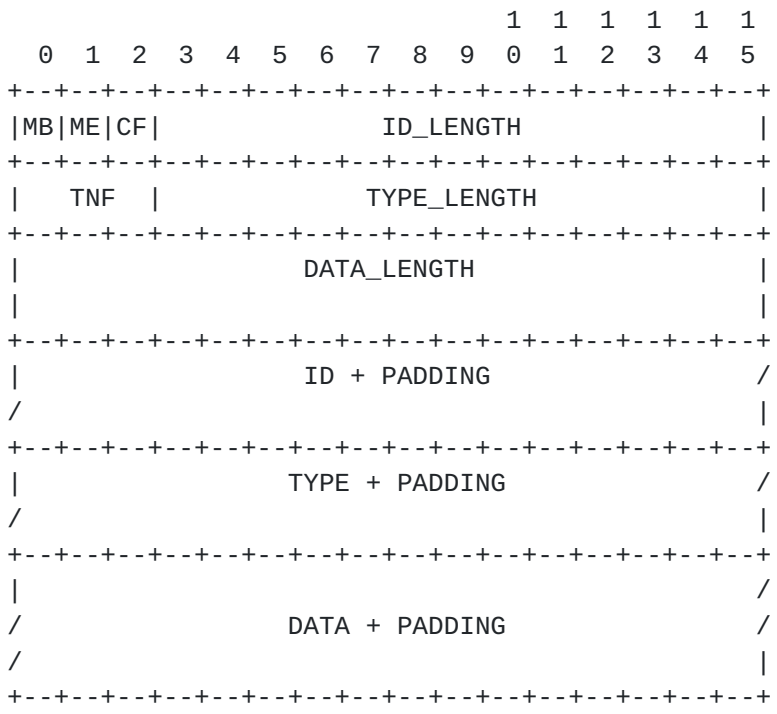

Figure 3: DIME Record Layout. The use of "/" indicates a field length which is a multiple of 4 octets.

3.2.1 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).

3.2.2 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).

3.2.3 CF (Chunked Flag)

The CF flag is a 1 bit field indicating a chunked record. See section 2.1.3 for a description of how to encode a chunked record series.

3.2.4 ID_LENGTH

An unsigned 13 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 2.2.3).

3.2.5 TNF (Type Name Format)

The TNF field value indicates the structure of the value of the TYPE field (see section 2.2.2). The TNF field is a 3 bit field with values defined in Table 1:

Type Name Format	Value
None	0x00
media-type as defined in RFC 2616 [9]	0x01
Absolute URI as defined in RFC 2396 [8]	0x02
Reserved	0x03-0x07

Table 1: DIME TNF field values

Reserved TNF field values are reserved for future use and MUST NOT be used. The value 0x00 MUST be used in all but the first chunk in a chunked record series (see section 2.1.3). It MUST NOT be used in any other record.

3.2.6 TYPE_LENGTH

An unsigned 13 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 2.2.2).

3.2.7 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 2.2.1).

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

2.1.3).

3.2.8 ID

The value of the ID field is an identifier in the form of a URI [8] (see section 2.2.3 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 [8]).

With the exception of subsequent chunked records (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 3.2.4).

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

3.2.9 TYPE

The value of the TYPE field is an identifier of structure indicated by the TNF field describing the type of the payload (see section 2.2.2). With the exception of subsequent chunked records (see section 2.1.3), all records MUST have a non-zero TYPE_LENGTH field and a TYPE field value that follows the rules implied by the value of the TNF field. There is no default value for the TYPE field.

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 3.2.6).

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

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

3.2.10 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 3.2.7).

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 [8] for some identifiers. To DIME, a URI is simply a formatted string that identifiesùvia name, location, or any other characteristicùa resource on the Web.

The use of IP addresses in URIs SHOULD be avoided whenever possible (see RFC 1900 [2]). However, when used, the literal format for IPv6 addresses in URIs as described by RFC 2732 [12] 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 [8]. 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 2^13-1 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 may provide different levels of support for

internationalization. Implementers are referred to RFC 2718 [11] for internationalization consideration of URIs and RFC 2046 [5] for internationalization considerations of MIME media types.

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 [6] and in the context of the "application/postscript" and the "message/external-body" media type in RFC 2046 [5].

6 IANA Considerations

This draft describes a new content type, "application/dime", which must be registered with IANA following the guidelines in RFC 2048 [6] (see [15] for the online registration form of media types).

7 Intellectual Property

The following notice is copied from RFC 2026, 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] J. B. Postel, "Simple Mail Transfer Protocol", RFC 821, ISI, August 1982

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

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

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

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

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

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

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

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

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

[11] 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

[12] R. Hinden, B. Carpenter, L. Masinter, "Format for Literal IPv6 Addresses in URL's", RFC 2732, Nokia, IBM, AT&T, December 1999

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

[14] List of Uniform Resource Identifier (URI) schemes registered by IANA is available at "http://www.iana.org/assignments/uri-schemes"

[15] Online form for MIME media type registration
http://www.iana.org/cgi-bin/mediatypes.pl

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

Erik Christensen
Microsoft
One Microsoft Way, Redmond, WA 90852
Email: erikc@microsoft.com