MiniMIME Reference Manual

Generated by Doxygen 1.5.1

Thu Mar 29 17:59:08 2007

Contents

Chapter 1

MiniMIME Module Index

1.1 MiniMIME Modules

Here is a list of all modules:

Chapter 2

MiniMIME File Index

2.1 MiniMIME File List

Here is a list of all documented files with brief descriptions:

Chapter 3

MiniMIME Page Index

3.1 MiniMIME Related Pages

Here is a list of all related documentation pages:

Chapter 4

MiniMIME Module Documentation

4.1 Manipulating MiniMIME codecs

Codec manipulation

- int **mm_codec_hasdecoder** (const char *encoding)
- int **mm_codec_hasencoder** (const char *encoding)
- int **mm_codec_isregistered** (const char *encoding)
- int **mm_codec_register** (const char *encoding, char *(*encoder)(char *data, u_int32_t i), char *(*decoder)(char *data))
- int **mm_codec_unregister** (const char *encoding)
- int **mm_codec_unregisterall** (void)
- void **mm_codec_registerdefaultcodecs** (void)

4.1.1 Function Documentation

4.1.1.1 int mm_codec_hasdecoder (const char * *encoding*)

Looks up whether a context has an decoder installed for a given encoding

Parameters:

encoding The encoding specifier to look up

Returns:

1 if a decoder is installed or 0 if not

4.1.1.2 int mm_codec_hasencoder (const char * *encoding*)

Looks up whether a context has an encoder installed for a given encoding

Parameters:

ctx A valid MIME context

encoding The encoding specifier to look up

Returns:

1 if an encoder is installed or 0 if not

4.1.1.3 int mm_codec_isregistered (const char * *encoding*)

Looks up whether a codec for a given encoding is installed to a context

Parameters:

encoding The encoding specifier to look up

Returns:

1 if a codec was found or 0 if not

4.1.1.4 int mm_codec_register (const char * *encoding*, char *(*)(char *data, u_int32_t i) *encoder*, char *(*)(char *data) *decoder*)

Registers a codec with the MiniMIME library

Parameters:

encoding The encoding specifier for which to register the codec
encoder The encoder function for this encoding
decoder The decoder function for this encoding

Returns:

1 if successfull or 0 if not

This function registers a codec for a given MiniMIME context. The codec may provide an decoder, an encoder or both (but not none). If there is a codec already installed for this encoding, the function will puke.

4.1.1.5 void mm_codec_registerdefaultcodecs (void)

Registers the default codecs to a MiniMIME context

This functions registers the codecs for the following encodings to a MiniMIME context:

- Base64
- (TODO:) Quoted-Printable

4.1.1.6 int mm_codec_unregister (const char * *encoding*)

Unregisters a MiniMIME codec

Parameters:

encoding The encoding specifier which to unregister

Returns:

0 if unregistered successfully, or -1 if there was no such codec

4.1.1.7 int mm_codec_unregisterall (void)

Unregisters all codecs within a context

Parameters:

ctx A valid MiniMIME context

Returns:

0 if all codecs were unregistered successfully or -1 if an error occured.

Note:

Foobar

4.2 Accessing and manipulating Content-Type objects

Functions for manipulating Content-Type objects

- mm_content * **mm_content_new** (void)
- void **mm_content_free** (struct mm_content *ct)
- int **mm_content_attachparam** (struct mm_content *ct, struct mm_param *param)
- char * **mm_content_getparambyname** (struct mm_content *ct, const char *name)
- mm_param * **mm_content_getparamobjbyname** (struct mm_content *ct, const char *name)
- int **mm_content_setmaintype** (struct mm_content *ct, char *value, int copy)
- char * **mm_content_getmaintype** (struct mm_content *ct)
- char * **mm_content_getsubtype** (struct mm_content *ct)
- char * **mm_content_gettype** (struct mm_content *ct)
- int **mm_content_setsubtype** (struct mm_content *ct, char *value, int copy)
- int **mm_content_settype** (struct mm_content *ct, const char *fmt,...)
- int **mm_content_iscomposite** (struct mm_content *ct)
- int **mm_content_isvalidencoding** (const char *encoding)
- int **mm_content_setencoding** (struct mm_content *ct, const char *encoding)
- int **mm_content_getencoding** (struct mm_content *ct, const char *encoding)
- char * **mm_content_paramstostring** (struct mm_content *ct)
- char * **mm_content_tostring** (struct mm_content *ct)

Variables

- int **mm_encoding_mappings::type**

4.2.1 Function Documentation

4.2.1.1 int mm_content_attachparam (struct mm_content * *ct*, struct mm_param * *param*)

Attaches a parameter to a Content-Type object

Parameters:

ct The target Content-Type object

param The Content-Type parameter which to attach

Returns:

0 on success and -1 on failure

4.2.1.2 void mm_content_free (struct mm_content * *ct*)

Releases all memory associated with an Content-Type object

Parameters:

ct A Content-Type object

Returns:

Nothing

4.2.1.3 int mm_content_getencoding (struct mm_content * *ct*, const char * *encoding*)

Gets the numerical ID of a content encoding identifier

Parameters:

ct A valid Content Type object

encoding A string representing the content encoding identifier

Returns:

The numerical ID of the content encoding

4.2.1.4 char* mm_content_getmaintype (struct mm_content * *ct*)

Retrieves the main MIME type stored in a Content-Type object

Parameters:

ct A valid Content-Type object

Returns:

A pointer to the string representing the main type

4.2.1.5 char* mm_content_getparambyname (struct mm_content * *ct*, const char * *name*)

Gets a parameter value from a Content-Type object.

Parameters:

ct the Content-Type object

name the name of the parameter to retrieve

Returns:

The value of the parameter on success or a NULL pointer on failure

4.2.1.6 char* mm_content_getsubtype (struct mm_content * *ct*)

Retrieves the sub MIME type stored in a Content-Type object

Parameters:

ct A valid Content-Type object

Returns:

A pointer to the string holding the current sub MIME type

4.2.1.7 int mm_content_iscomposite (struct mm_content * *ct*)

Checks whether the Content-Type represents a composite message or not

Parameters:

ct A valid Content-Type object

Returns:

1 if the Content-Type object represents a composite message or 0 if not.

4.2.1.8 int mm_content_isvalidencoding (const char * *encoding*)

Verifies whether a string represents a valid encoding or not.

Parameters:

encoding The string to verify

Returns:

1 if the encoding string is valid or 0 if not

4.2.1.9 struct mm_content* mm_content_new (void)

Creates a new object to hold a Content-Type representation. The allocated memory must later be freed using **mm_content_free()** (p. 10)

Returns:

An object representing a MIME Content-Type

See also:

mm_content_free (p. 10)

4.2.1.10 char* mm_content_paramstostring (struct mm_content * *ct*)

Constructs a MIME conform string of Content-Type parameters.

Parameters:

ct A valid Content Type object

Returns:

A pointer to a string representing the Content-Type parameters in MIME terminology, or NULL if either the Content-Type object is invalid, has no parameters or no memory could be allocated.

This function constructs a MIME conform string including all the parameters associated with the given Content-Type object. It should NOT be used if you need an opaque copy of the current MIME part (e.g. for PGP purposes).

4.2.1.11 int mm_content_setencoding (struct mm_content * *ct*, const char * *encoding*)

Set the encoding of a MIME entitity according to a mapping table

Parameters:

ct A valid content type object

encoding A string representing the content encoding

Returns:

0 if successfull or -1 if not (i.e. unknown content encoding)

4.2.1.12 int mm_content_setmaintype (struct mm_content * *ct*, char * *value*, int *copy*)

Sets the MIME main type for a MIME Content-Type object

Parameters:

ct The MIME Content-Type object

value The value which to set the main type to

copy Whether to make a copy of the value (original value must be freed afterwards to prevent memory leaks).

Bug

The xfree() call could lead to undesirable results. Do we really need it?

4.2.1.13 int mm_content_setsubtype (struct mm_content * *ct*, char * *value*, int *copy*)

Sets the MIME sub type for a MIME Content-Type object

Parameters:

ct The MIME Content-Type object

value The value which to set the sub type to

copy Whether to make a copy of the value (original value must be freed afterwards to prevent memory leaks).

Bug

The xfree() call could lead to undesirable results. Do we really need it?

4.2.1.14 char* mm_content_tostring (struct mm_content * *ct*)

Creates a Content-Type header according to the object given

Parameters:

ct A valid Content-Type object

4.3 Accessing and manipulating MIME contexts

Manipulating MiniMIME contexts

- MM_CTX * **mm_context_new** (void)
- void **mm_context_free** (MM_CTX *ctx)
- int **mm_context_attachpart** (MM_CTX *ctx, struct mm_mimepart *part)
- int **mm_context_attachpart_after** (MM_CTX *ctx, struct mm_mimepart *part, int pos)
- int **mm_context_deletepart** (MM_CTX *ctx, int which, int freemem)
- int **mm_context_countparts** (MM_CTX *ctx)
- mm_mimepart * **mm_context_getpart** (MM_CTX *ctx, int which)
- int **mm_context_iscomposite** (MM_CTX *ctx)
- int **mm_context_haswarnings** (MM_CTX *ctx)
- int **mm_context_generateboundary** (MM_CTX *ctx)
- int **mm_context_setpreamble** (MM_CTX *ctx, char *preamble)
- char * **mm_context_getpreamble** (MM_CTX *ctx)
- int **mm_context_flatten** (MM_CTX *ctx, char **flat, size_t *length, int flags)

4.3.1 Detailed Description

Each message in MiniMIME is represented by a so called "context". A context holds all necessary information given about a MIME message, such as the envelope, all MIME parts etc.

4.3.2 Function Documentation

4.3.2.1 int mm_context_attachpart (MM_CTX * *ctx*, struct mm_mimepart * *part*)

Attaches a MIME part object to a MiniMIME context.

Parameters:

ctx the MiniMIME context

part the MIME part object to attach

Returns:

0 on success or -1 on failure. Sets mm_errno on failure.

This function attaches a MIME part to a context, appending it to the end of the message.

The MIME part should be initialized before attaching it using **mm_mimepart_new()** (p. 27).

4.3.2.2 int mm_context_attachpart_after (MM_CTX * *ctx*, struct mm_mimepart * *part*, int *pos*)

Attaches a MIME part object to a MiniMIME context at a given position

Parameters:

ctx A valid MiniMIME context

part The MIME part object to attach

pos After which part to attach the object

Returns:

0 on success or -1 if the given position is invalid

See also:

mm_context_attachpart (p. 14)

This function attaches a MIME part object after a given position in the specified context. If the position is invalid (out of range), the part will not get attached to the message and the function returns -1. If the index was in range, the MIME part will get attached after the MIME part at the given position, moving any possible following MIME parts one down the hierarchy.

4.3.2.3 int mm_context_countparts (MM_CTX * *ctx*)

Counts the number of attached MIME part objects in a given MiniMIME context

Parameters:

ctx The MiniMIME context

Returns:

The number of attached MIME part objects

4.3.2.4 int mm_context_deletepart (MM_CTX * *ctx*, int *which*, int *freemem*)

Deletes a MIME part object from a MiniMIME context

Parameters:

ctx A valid MiniMIME context object

which The number of the MIME part object to delete

freemem Whether to free the memory associated with the MIME part object

Returns:

0 on success or -1 on failure. Sets mm_errno on failure.

This function deletes a MIME part from a given context. The MIME part to delete is specified as numerical index by the parameter "which". If the parameter "freemem" is set to anything greater than 0, the memory that is associated will be free'd by using **mm_mimepart_free()** (p. 25), otherwise the memory is left untouched (if you still have a pointer to the MIME part around).

4.3.2.5 int mm_context_flatten (MM_CTX * *ctx*, char ** *flat*, size_t * *length*, int *flags*)

Creates an ASCII message of the specified context

Parameters:

ctx A valid MiniMIME context object

flat Where to store the message

flags Flags that affect the flattening process

This function "flattens" a MiniMIME context, that is, it creates an ASCII represantation of the message the context contains. The flags can be a bitwise combination of the following constants:

- MM_FLATTEN_OPAQUE : use opaque MIME parts when flattening
- MM_FLATTEN_SKIPENVELOPE : do not flatten the envelope part

Great care is taken to not produce invalid MIME output.

4.3.2.6 void mm_context_free (MM_CTX * *ctx*)

Releases a MiniMIME context object

Parameters:

ctx A valid MiniMIME context

See also:

mm_context_new (p. 17)

This function releases all memory associated with MiniMIME context object that was created using **mm_context_new()** (p. 17). It will also release all memory used for the MIME parts attached, and their specific properties (such as Content-Type information, headers, and the body data).

4.3.2.7 int mm_context_generateboundary (MM_CTX * *ctx*)

Generates a generic boundary string for a given context

Parameters:

ctx A valid MiniMIME context

Returns:

0 on success or -1 on failure

This function generates a default boundary string for the given context. If there is already a boundary for the context, the memory will be free()'d.

4.3.2.8 struct mm_mimepart* mm_context_getpart (MM_CTX * *ctx*, int *which*)

Gets a specified MIME part object from a MimeMIME context

Parameters:

ctx The MiniMIME context

which The number of the MIME part object to retrieve

Returns:

The requested MIME part object on success or a NULL pointer if there is no such part.

4.3.2.9 int mm_context_haswarnings (MM_CTX * *ctx*)

Checks whether there are any warnings associated with a given context

Parameters:

ctx A valid MiniMIME context

Returns:

1 if there are warnings associated with the context, otherwise 0

4.3.2.10 int mm_context_iscomposite (MM_CTX * *ctx*)

Checks whether a given context represents a composite (multipart) message

Parameters:

ctx A valid MiniMIME context object

Returns:

1 if the context is a composite message or 0 if it's flat

4.3.2.11 MM_CTX* mm_context_new (void)

Creates a new MiniMIME context object.

Returns:

a new MiniMIME context object

See also:

mm_context_free (p. 16)

This function creates a new MiniMIME context, which will hold a message. The memory needed is allocated dynamically and should later be free'd using **mm_context_free()** (p. 16).

Before a context can be created, the MiniMIME library needs to be initialized properly using mm_library_init().

4.3.2.12 int mm_context_setpreamble (MM_CTX * *ctx*, char * *preamble*)

Sets a preamble for the given MiniMIME context

Parameters:

ctx A valid MiniMIME context

preamble The preamble to set

Returns:

0 on success or -1 on failure

This function sets the MIME preamble (the text between the end of envelope headers and the beginning of the first MIME part) for a given context object. If preamble is a NULL-pointer then the preamble will be deleted, and the currently associated memory will be free automagically.

4.4 Accessing and manipulating a message's envelope

Accessing and manipulating a message's envelope

- int **mm_envelope_getheaders** (MM_CTX *ctx, char **result, size_t *length)
- int **mm_envelope_setheader** (MM_CTX *ctx, const char *name, const char *fmt,...)
- int **mm_envelope_getrecipients** (MM_CTX *ctx, char **result, size_t *length)

4.4.1 Function Documentation

4.4.1.1 int mm_envelope_getheaders (MM_CTX * *ctx*, char ** *result*, size_t * *length*)

Gets an ASCII representation of all envelope headers

Parameters:

ctx A valid MiniMIME context

result Where to store the resulting ASCII headers

length Where to store the length of the result

Returns:

0 on success or -1 on failure.

Note:

Sets mm_errno on failure

This is mainly a convinience function. It constructs an ASCII representation from all of the message's envelope headers and stores the result in headers. Memory is allocated dynamically, and the total length of the result is stored in length. This function takes care that the output is MIME conform, and folds long lines according to the MIME standard at position 78 of the string. It also nicely formats all MIME related header fields, such as the Content-Type header.

Since the memory needed to store the result is allocated dynamically, one should take care of freeing it again when it's not needed anymore. If an error occurs, *result will be set to NULL, *length will be set to zero and mm_errno will be set to a reasonable value.

4.4.1.2 int mm_envelope_getrecipients (MM_CTX * *ctx*, char ** *result*, size_t * *length*)

Gets the list of recipients for a MIME message

Parameters:

ctx A valid MiniMIME context

result Where to store the result

length Where to store the length of the result

Returns:

0 on success or -1 on error

Note:

Sets mm_errno on error

This functions gets the list of recipients for a given MIME message. It does so by concatenating the "From" and "Cc" header fields, and storing the results in recipients. The memory needed to store the result is allocated dynamically, and the total length of the result is stored in length.

One should take care to free() the result once it's not needed anymore.

4.4.1.3 int mm_envelope_setheader (MM_CTX * *ctx*, const char * *name*, const char * *fmt*, ...)

Sets a header field in the envelope

Parameters:

ctx A valid MiniMIME context

name The name of the header field to set

fmt A format string specifying the value of the header field

Returns:

0 on success or -1 on failure

This function generates a new MIME header and attaches it to the first MIME part (the envelope) found in the given context. If no part is attached already, the function will return an error. The function will store a copy of "name" as the header's name field, and dynamically allocate the memory needed to build the format string.

4.5 MiniMIME error functions

Functions

- void **mm_error_init** (void)
- void **mm_error_setmsg** (const char *fmt,...)
- char * **mm_error_string** (void)

4.5.1 Function Documentation

4.5.1.1 void mm_error_init (void)

Initializes the global error object

This function initializes the global error object mm_error. This must be done when the library is initialized, and is automatically called from mm_init_library().

4.5.1.2 void mm_error_setmsg (const char * *fmt*, ...)

Sets a descriptive error message

Parameters:

fmt The error message as format string

This function is called from the various MiniMIME modules in case an error occured. Should never be called by the user.

4.5.1.3 char* mm_error_string (void)

Retrieves the current error message

Returns:

The currently set error message

This function can be used to retrieve a descriptive error message for the current error, much like strerror() function of libc. When this function is called without an error being set, it returns the string "No error". The string returned does not need to be freed, since it is not dynamically allocated by the library.

4.6 Accessing and manipulating MIME parts

Creating and destroying MIME parts

- mm_mimepart * **mm_mimepart_new** (void)
- mm_mimepart * **mm_mimepart_fromfile** (const char *filename)
- void **mm_mimepart_free** (struct mm_mimepart *part)

Accessing the MIME part's mail header

- int **mm_mimepart_attachheader** (struct mm_mimepart *part, struct mm_mimeheader *header)
- int **mm_mimepart_countheaders** (struct mm_mimepart *part)
- int **mm_mimepart_countheaderbyname** (struct mm_mimepart *part, const char *name)
- mm_mimeheader * **mm_mimepart_getheaderbyname** (struct mm_mimepart *part, const char *name, int idx)
- const char * **mm_mimepart_getheadervalue** (struct mm_mimepart *part, const char *name, int idx)
- int **mm_mimepart_headers_start** (struct mm_mimepart *part, struct mm_mimeheader **id)
- mm_mimeheader * **mm_mimepart_headers_next** (struct mm_mimepart *part, struct mm_mimeheader **id)

Accessing and manipulating the MIME part's body

- char * **mm_mimepart_getbody** (struct mm_mimepart *part, int opaque)
- void **mm_mimepart_setbody** (struct mm_mimepart *part, const char *data, int opaque)
- size_t **mm_mimepart_getlength** (struct mm_mimepart *part)
- char * **mm_mimepart_decode** (struct mm_mimepart *part)
- int **mm_mimepart_flatten** (struct mm_mimepart *part, char **result, size_t *length, int opaque)
- int **mm_mimepart_setdefaultcontenttype** (struct mm_mimepart *part, int composite)

Accessing the MIME part's Content-Type information

- void **mm_mimepart_attachcontenttype** (struct mm_mimepart *part, struct mm_content *ct)
- mm_content * **mm_mimepart_gettype** (struct mm_mimepart *part)

4.6.1 Detailed Description

MIME parts, also called entities, represent the structure of a MIME message. "Normal" internet messages have only a single part, and are called "flat" messages. Multipart messages have more then one part, and each MIME part can have it's own subset of headers.

Provided here are functions to easily access all informations from a MIME part, including their specific headers and bodies.

4.6.2 Function Documentation

4.6.2.1 void mm_mimepart_attachcontenttype (struct mm_mimepart * *part*, struct mm_content * *ct*)

Attaches a context type object to a MIME part

Parameters:

part A valid MIME part object

ct The content type object to attach

Returns:

Nothing

This function attaches a Content-Type object to a MIME part. It does not care whether the Content-Type suites the actual content in the MIME part, so the programmer should take care of that.

4.6.2.2 int mm_mimepart_attachheader (struct mm_mimepart * *part*, struct mm_mimeheader * *header*)

Attaches a mm_mimeheader object to a MIME part

Parameters:

part A valid MIME part object

header A valid MIME header object

Returns:

0 if successfull or -1 if the header could not be attached

4.6.2.3 int mm_mimepart_countheaderbyname (struct mm_mimepart * *part*, const char * *name*)

Retrieves the number of MIME headers with a given name in a MIME part

Parameters:

part A valid MIME part object

name The name of the MIME header which to count for

Returns:

The number of MIME headers within the MIME part

4.6.2.4 int mm_mimepart_countheaders (struct mm_mimepart * *part*)

Retrieves the number of MIME headers available in a MIME part

Parameters:

part A valid MIME part object

Returns:

The number of MIME headers within the MIME part

4.6.2.5 char* mm_mimepart_decode (struct mm_mimepart * *part*)

Decodes a MIME part according to it's encoding using MiniMIME codecs

Parameters:

A valid MIME part object

Returns:

0 if the MIME part could be successfully decoded or -1 if not

Note:

Sets mm_errno on error

This function decodes the body of a MIME part with a registered decoder according to it's Content-Transfer-Encoding header field.

4.6.2.6 int mm_mimepart_flatten (struct mm_mimepart * *part*, char ** *result*, size_t * *length*, int *opaque*)

Creates an ASCII representation of the given MIME part

Parameters:

part A valid MIME part object

result Where to store the result

length Where to store the length of the result

opaque Whether to use the opaque MIME part 0 on success or -1 on error.

See also:

mm_context_flatten (p. 16)

This function creates an ASCII representation of a given MIME part. It will dynamically allocate the memory needed and stores the result in the memory region pointed to by result. The length of the result will be stored in length. If opaque is set to 1, mm_mimepart_flatten will store an opaque version of the MIME part in result, which means no headers will be created or sanitized. This is particulary useful if the part is digitally signed by e.g. PGP, and the signature spans the header fields of the part in question.

4.6.2.7 void mm_mimepart_free (struct mm_mimepart * *part*)

Frees all memory allocated by a mm_mimepart object.

Parameters:

part A pointer to an allocated mm_mimepart object

See also:

mm_mimepart_new (p. 27)

4.6.2.8 struct mm_mimepart* mm_mimepart_fromfile (const char * *filename*)

Creates a MIME part from a file

Parameters:

filename The name of the file to create the MIME part from

Returns:

A pointer to a new MIME part object

This function creates a new MIME part object from a file. The object should be freed using **mm_mimepart_free()** (p. 25) later on. This function does NOT set the Content-Type and neither does any encoding work.

4.6.2.9 char* mm_mimepart_getbody (struct mm_mimepart * *part*, int *opaque*)

Gets the pointer to the MIME part's body data

Parameters:

part A valid MIME part object

opaque Whether to get the opaque part or not

Returns:

A pointer to the MIME part's body

See also:

mm_mimepart_setbody (p. 28)

4.6.2.10 struct mm_mimeheader* mm_mimepart_getheaderbyname (struct mm_mimepart * *part*, const char * *name*, int *idx*)

Get a MIME header object from a MIME part

Parameters:

part A valid MIME part object

name The name of the MIME header which to retrieve

idx Which header field to get (in case of multiple headers of the same name).

Returns:

A pointer to the requested MIME header on success, or NULL if there either isn't a header with the requested name or idx is out of range.

4.6.2.11 const char* mm_mimepart_getheadervalue (struct mm_mimepart * *part*, const char * *name*, int *idx*)

Gets the value of a MIME header object

Parameters:

part A valid MIME part object

name The name of the header field to get the value from

idx The index of the header field to get, in case there are multiple headers with the same name.

Returns:

A pointer to the requested value on success, or NULL if there either isn't a header with the requested name or idx is out of range.

4.6.2.12 size_t mm_mimepart_getlength (struct mm_mimepart * *part*)

Gets the length of a given MIME part object

Parameters:

part A valid MIME part object

Returns:

The size of the part's body in byte.

This function returns the total length of the given MIME part's body. The length does not include the headers of the MIME parts. If the function returns 0, no body part is set currently.

4.6.2.13 struct mm_content* mm_mimepart_gettype (struct mm_mimepart * *part*)

Gets the Content-Type of a given MIME part object

Parameters:

part A valid MIME part object

Returns:

The Content-Type object of the specified MIME part

This function returns a pointer to the Content-Type object of the given MIME part. This pointer might be set to NULL, indicating that there is no Content-Type object for the given MIME part currently.

4.6.2.14 struct mm_mimeheader* mm_mimepart_headers_next (struct mm_mimepart * *part*, struct mm_mimeheader ** *id*)

Returns the next MIME header of a given MIME part object

Parameters:

part A valid MIME part object

id A previously initialized MIME header object

Returns:

A pointer to the MIME header object or NULL if end of headers was reached.

See also:

mm_mimepart_headers_start (p. 27)

4.6.2.15 int mm_mimepart_headers_start (struct mm_mimepart * *part*, struct mm_mimeheader ** *id*)

Initializes a header loop for a given MIME part

Parameters:

part A valid MIME part object

id The address of a MIME header object (to allow reentrance)

Returns:

0 on success or -1 on failure

See also:

mm_mimepart_headers_next (p. 27)

Looping through headers can be done in the following way:

```
struct mm_mimeheader *header, *lheader;

mm_mimepart_headers_start(part, &lheader);

while ((header = mm_mimepart_headers_next(part, &lheader)) != NULL) {
       printf("%s: %s\n", header->name, header->value);
}
```

For convienience, the macro mm_mimepart_headers_foreach() can be used to loop through headers in a one-shot manner.

4.6.2.16 struct mm_mimepart* mm_mimepart_new (void)

Allocates memory for a new mm_mimepart structure and initializes it.

Returns:

A pointer to a struct of type mm_mimeheader or NULL on failure

See also:

mm_mimepart_free (p. 25)

Note:

The memory must be freed by using **mm_mimepart_free()** (p. 25) later on.

4.6.2.17 void mm_mimepart_setbody (struct mm_mimepart * *part*, const char * *data*, int *opaque*)

Sets the MIME part's body data

Parameters:

part A valid MIME part object

data A pointer to the data which to set

See also:

mm_mimepart_getbody (p. 25)

This functions sets the body data for a given MIME part. The string pointed to by data must be NUL-terminated. The data is copied into the MIME part's body, and thus, the memory pointed to by data can be freed after the operation.

4.6.2.18 int mm_mimepart_setdefaultcontenttype (struct mm_mimepart * *part*, int *composite*)

Sets the default Content-Type for a given MIME part

Parameters:

part A valid MIME part object

part Whether the Content-Type should be for composite or not

Returns:

0 on success or -1 on failure

This function sets a default Content-Type according to RFC 2045 with a value of "text/plain; charset="us-ascii"". This function should only be used if the MIME part in question does not have a valid Content-Type specification.

4.7 MIME related utility functions

4.8 Accessing and manipulating MIME parameters

Functions for manipulating MIME parameters

MIME parameters are properties attached to certain MIME headers, such as Content-Type and Content-Disposition. MIME parameters have a textual representations as in *name=value*. They contain important information about the MIME structure of a message, such as the boundary string used, which charset was used to encode the message and so on. This module provides simple to use functions to query or set MIME parameters.

Each MIME header may hold an arbitrary amount of such parameters, which are delimeted by each other with a semicolon.

- mm_param * **mm_param_new** (void)
- void **mm_param_free** (struct mm_param *param)
- mm_param * **mm_param_generate** (const char *name, const char *value)
- char * **mm_param_setname** (struct mm_param *param, const char *name, int copy)
- char * **mm_param_setvalue** (struct mm_param *param, const char *value, int copy)
- const char * **mm_param_getname** (struct mm_param *param)
- const char * **mm_param_getvalue** (struct mm_param *param)

4.8.1 Function Documentation

4.8.1.1 void mm_param_free (struct mm_param * *param*)

Releases all memory associated with a MIME parameter object.

Parameters:

param A valid MIME parameter object to be freed

Returns:

Nothing

See also:

mm_param_new (p. 31)

4.8.1.2 struct mm_param* mm_param_generate (const char * *name*, const char * *value*)

Generates a new Content-Type parameter with the given name and value

Parameters:

name The name of the MIME parameter

value The value of the MIME parameter

Returns:

A new MIME parameter object

See also:

mm_param_free (p. 30)
mm_param_new (p. 31)

This function generates a new MIME parameter, with the name and value given as the arguments. The needed memory for the operation is allocated dynamically. It stores a copy of name and value in the actual object, so the memory holding the arguments can safely be freed after successfull return of this function.

4.8.1.3 const char* mm_param_getname (struct mm_param * *param*)

Gets the name of a MIME parameter object

Parameters:

param A valid MIME parameter object

Returns:

The name of the MIME parameter

4.8.1.4 const char* mm_param_getvalue (struct mm_param * *param*)

Gets the value of a MIME parameter object

Parameters:

param A valid MIME parameter object

Returns:

The value of the MIME parameter

4.8.1.5 struct mm_param* mm_param_new (void)

Creates a new object to hold a MIME parameter.

Returns:

An object representing a MIME parameter

See also:

mm_param_free (p. 30)

Note:

The allocated memory must later be freed using **mm_param_free()** (p. 30)

4.8.1.6 char* mm_param_setname (struct mm_param * *param*, const char * *name*, int *copy*)

Sets the name of the given MIME parameter

Parameters:

param A valid MIME parameter object

name The new name of the parameter

copy If set to > 0, copy the value stored in name

Returns:

The address of the previous name for passing to free()

4.8.1.7 char* mm_param_setvalue (struct mm_param * *param*, const char * *value*, int *copy*)

Sets the value of the given MIME parameter

Parameters:

param A valid MIME parameter object

name The new value for the parameter

copy If set to > 0, copy the value stored in value

Returns:

The address of the previous value for passing to free()

4.9 General purpose utility functions

Utility functions

- void **xfree** (void *)
- char * **xstrdup** (const char *)

Functions

- void * **xmalloc** (size_t size)
- void * **xrealloc** (void *p, size_t size)
- char * **mm_unquote** (const char *string)
- char * **mm_uncomment** (const char *string)
- char * **xstrsep** (char **stringp, const char *delim)
- char * **mm_stripchars** (char *input, char *strip)
- char * **mm_addchars** (char *input, char *add, u_int16_t linelength)

4.9.1 Function Documentation

4.9.1.1 char* mm_addchars (char * *input*, char * *add*, u_int16_t *linelength*)

Adds characters to a string at given positions

Parameters:

input The string to which to add characters

add The character string to add

linelength The position where to add the character

Returns:

A copy of the string with characters added

This function adds the characters add at each linelength positions and returns this new string.

4.9.1.2 char* mm_stripchars (char * *input*, char * *strip*)

Strips a given character set from a string

Parameters:

input The string which to strip

strip The character set to strip off

Returns:

A copy of the original string with all chars stripped

4.9.1.3 char* mm_uncomment (const char * *string*)

Removes MIME comments from a string

Parameters:

string The string to uncomment

Returns:

A pointer to the uncommented string or NULL on error. Sets mm_errno.

This function removes MIME comments from a string (included in parantheses). It returns a pointer to a newly allocated memory region in which the uncommented string is stored. The returned string needs to be freed when it's not used anymore.

4.9.1.4 char* mm_unquote (const char * *string*)

Unquotes a string

Parameters:

string The quoted string to unquote

Returns:

A pointer to the unquoted string

This function unquotes a string. That is, it returns a pointer to a newly allocated memory region in which the unquoted string is stored. Only leading and trailing double-qoutes are removed. The string needs to be freed when it is not needed anymore.

4.9.1.5 void* xmalloc (size_t *size*)

Allocates a block of memory

Parameters:

size The size of the memory region to allocate

Returns:

A pointer to the allocated memory region

xmalloc() (p. 34) calls abort() if either the size argument is negative or the requested memory amount could not be allocated via an assert() call.

4.9.1.6 void* xrealloc (void * *p*, size_t *size*)

realloc() wrapper

Parameters:

p Pointer to a memory region which should be reallocated

size The new size of the memory region

Returns:

A pointer to the reallocated memory region

xrealloc() (p. 34) is a wrapper around realloc() which calls abort() if either the size argument is negative or the requested memory amount could not be allocated.

4.9.1.7 char* xstrsep (char ** *stringp*, const char * *delim*)

separate strings

Parameters:

stringp A pointer to the string being splitted

delim The delimeter string

This function works similar to strsep(), with the difference that delim is treated as a whole.

Chapter 5

MiniMIME File Documentation

5.1 mm_codecs.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <assert.h>
#include "mm_internal.h"
#include "mm_util.h"
```

Functions

Codec manipulation

- int **mm_codec_hasdecoder** (const char *encoding)
- int **mm_codec_hasencoder** (const char *encoding)
- int **mm_codec_isregistered** (const char *encoding)
- int **mm_codec_register** (const char *encoding, char *(*encoder)(char *data, u_int32_t i), char *(*decoder)(char *data))
- int **mm_codec_unregister** (const char *encoding)
- int **mm_codec_unregisterall** (void)
- void **mm_codec_registerdefaultcodecs** (void)

Variables

- mm_codecs **codecs**

5.1.1 Detailed Description

This module contains functions to manipulate MiniMIME codecs

5.2 mm_contenttype.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <ctype.h>
#include <assert.h>
#include "mm_internal.h"
#include "mm_util.h"
```

Data Structures

- struct **mm_encoding_mappings**

Functions

Functions for manipulating Content-Type objects

- mm_content * **mm_content_new** (void)
- void **mm_content_free** (struct mm_content *ct)
- int **mm_content_attachparam** (struct mm_content *ct, struct mm_param *param)
- char * **mm_content_getparambyname** (struct mm_content *ct, const char *name)
- mm_param * **mm_content_getparamobjbyname** (struct mm_content *ct, const char *name)
- int **mm_content_setmaintype** (struct mm_content *ct, char *value, int copy)
- char * **mm_content_getmaintype** (struct mm_content *ct)
- char * **mm_content_getsubtype** (struct mm_content *ct)
- char * **mm_content_gettype** (struct mm_content *ct)
- int **mm_content_setsubtype** (struct mm_content *ct, char *value, int copy)
- int **mm_content_settype** (struct mm_content *ct, const char *fmt,...)
- int **mm_content_iscomposite** (struct mm_content *ct)
- int **mm_content_isvalidencoding** (const char *encoding)
- int **mm_content_setencoding** (struct mm_content *ct, const char *encoding)
- int **mm_content_getencoding** (struct mm_content *ct, const char *encoding)
- char * **mm_content_paramstostring** (struct mm_content *ct)
- char * **mm_content_tostring** (struct mm_content *ct)

5.2.1 Detailed Description

This module contains functions for manipulating Content-Type objects.

5.3 mm_context.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <assert.h>
#include "mm_internal.h"
```

Functions

Manipulating MiniMIME contexts

- MM_CTX * **mm_context_new** (void)
- void **mm_context_free** (MM_CTX *ctx)
- int **mm_context_attachpart** (MM_CTX *ctx, struct mm_mimepart *part)
- int **mm_context_attachpart_after** (MM_CTX *ctx, struct mm_mimepart *part, int pos)
- int **mm_context_deletepart** (MM_CTX *ctx, int which, int freemem)
- int **mm_context_countparts** (MM_CTX *ctx)
- mm_mimepart * **mm_context_getpart** (MM_CTX *ctx, int which)
- int **mm_context_iscomposite** (MM_CTX *ctx)
- int **mm_context_haswarnings** (MM_CTX *ctx)
- int **mm_context_generateboundary** (MM_CTX *ctx)
- int **mm_context_setpreamble** (MM_CTX *ctx, char *preamble)
- char * **mm_context_getpreamble** (MM_CTX *ctx)
- int **mm_context_flatten** (MM_CTX *ctx, char **flat, size_t *length, int flags)

5.3.1 Detailed Description

Modules for manipulating MiniMIME contexts

5.4 mm_envelope.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <ctype.h>
#include <assert.h>
#include "mm_internal.h"
#include "mm_util.h"
```

Functions

Accessing and manipulating a message's envelope

- int **mm_envelope_getheaders** (MM_CTX *ctx, char **result, size_t *length)
- int **mm_envelope_setheader** (MM_CTX *ctx, const char *name, const char *fmt,...)
- int **mm_envelope_getrecipients** (MM_CTX *ctx, char **result, size_t *length)

5.4.1 Detailed Description

This module contains functions for accessing a message's envelope. This are mainly wrapper functions for easy access.

5.5 mm_error.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <assert.h>
#include <errno.h>
#include "mm_internal.h"
#include "mm_util.h"
```

Functions

- void **mm_error_init** (void)
- void **mm_error_setmsg** (const char *fmt,...)
- void **mm_error_setlineno** (int lineno)
- char * **mm_error_string** (void)
- int **mm_error_lineno** (void)

5.5.1 Detailed Description

This module contains functions for MiniMIME error information/manipulation

5.6 mm_header.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <ctype.h>
#include <assert.h>
#include "mm_internal.h"
#include "mm_util.h"
```

Functions

- mm_mimeheader * **mm_mimeheader_new** (void)
- void **mm_mimeheader_free** (struct mm_mimeheader *header)
- mm_mimeheader * **mm_mimeheader_generate** (const char *name, const char *value)
- int **mm_mimeheader_uncomment** (struct mm_mimeheader *header)
- int **mm_mimeheader_uncommentbyname** (struct mm_mimepart *part, const char *name)
- int **mm_mimeheader_uncommentall** (struct mm_mimepart *part)

5.6.1 Detailed Description

This module contains functions for manipulating MIME headers

5.6.2 Function Documentation

5.6.2.1 void mm_mimeheader_free (struct mm_mimeheader * *header*)

Frees a MIME header object

Parameters:

header The MIME header object which to free

5.6.2.2 struct mm_mimeheader* mm_mimeheader_generate (const char * *name*, const char * *value*)

Creates a new MIME header, but does no checks whatsoever (create as-is)

5.6.2.3 struct mm_mimeheader* mm_mimeheader_new (void)

Creates a new MIME header object

Returns:

A new and initialized MIME header object

See also:

mm_mimeheader_free (p. 43)

This function creates and initializes a new MIME header object, which must later be freed using **mm_mimeheader_free()** (p. 43)

5.7 mm_internal.h File Reference

```
#include "mm.h"
```

Defines

- #define **debugp**(m,...)

Functions

Utility functions

- void * **xmalloc** (size_t)
- void * **xrealloc** (void *, size_t)
- void **xfree** (void *)
- char * **xstrdup** (const char *)
- char * **xstrsep** (char **, const char *)

5.7.1 Detailed Description

Data definitions for MiniMIME

5.7.2 Define Documentation

5.7.2.1 #define debugp(m, ...)

Value:

```
do { \
        fprintf(stderr, "%s:%d:: ", __FILE__, __LINE__); \
        fprintf(stderr, m, ## __VA_ARGS__); \
        fprintf(stderr, "\n"); \
        fflush(stderr); \
} while (0);
```

5.8 mm_mimepart.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <ctype.h>
#include <assert.h>
#include "mm_internal.h"
```

Functions

Creating and destroying MIME parts

- mm_mimepart * **mm_mimepart_new** (void)
- mm_mimepart * **mm_mimepart_fromfile** (const char *filename)
- void **mm_mimepart_free** (struct mm_mimepart *part)

Accessing the MIME part's mail header

- int **mm_mimepart_attachheader** (struct mm_mimepart *part, struct mm_mimeheader *header)
- int **mm_mimepart_countheaders** (struct mm_mimepart *part)
- int **mm_mimepart_countheaderbyname** (struct mm_mimepart *part, const char *name)
- mm_mimeheader * **mm_mimepart_getheaderbyname** (struct mm_mimepart *part, const char *name, int idx)
- const char * **mm_mimepart_getheadervalue** (struct mm_mimepart *part, const char *name, int idx)
- int **mm_mimepart_headers_start** (struct mm_mimepart *part, struct mm_mimeheader **id)
- mm_mimeheader * **mm_mimepart_headers_next** (struct mm_mimepart *part, struct mm_mimeheader **id)

Accessing and manipulating the MIME part's body

- char * **mm_mimepart_getbody** (struct mm_mimepart *part, int opaque)
- void **mm_mimepart_setbody** (struct mm_mimepart *part, const char *data, int opaque)
- size_t **mm_mimepart_getlength** (struct mm_mimepart *part)
- char * **mm_mimepart_decode** (struct mm_mimepart *part)
- int **mm_mimepart_flatten** (struct mm_mimepart *part, char **result, size_t *length, int opaque)
- int **mm_mimepart_setdefaultcontenttype** (struct mm_mimepart *part, int composite)

Accessing the MIME part's Content-Type information

- void **mm_mimepart_attachcontenttype** (struct mm_mimepart *part, struct mm_content *ct)
- mm_content * **mm_mimepart_gettype** (struct mm_mimepart *part)

5.8.1 Detailed Description

This module contains functions for manipulating MIME header objects.

5.9 mm_mimeutil.c File Reference

```
#include <sys/time.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <assert.h>
#include "mm_internal.h"
```

Defines

- #define **MM_DATE_LENGTH** 50

Functions

- int **mm_mimeutil_gendate** (char **result)
- int **mm_mimeutil_genboundary** (char *prefix, size_t length, char **result)

5.9.1 Detailed Description

This module contains various MIME related utility functions.

5.9.2 Function Documentation

5.9.2.1 int mm_mimeutil_gendate (char ** *result*)

Generates an RFC 2822 conform date string

Parameters:

timezone Whether to include timezone information

Returns:

A pointer to the actual date string

Note:

The pointer returned must be freed some time

This function generates an RFC 2822 conform date string to use in message headers. It allocates memory to hold the string and returns a pointer to it. The generated date is in the format (example):

Thu, 25 December 2003 16:35:22 +0100 (CET)

This function dynamically allocates memory and returns a pointer to it. This memory should be released with free() once not needed anymore.

5.10 mm_param.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <ctype.h>
#include <assert.h>
#include "mm_internal.h"
#include "mm_util.h"
```

Functions

Functions for manipulating MIME parameters

MIME parameters are properties attached to certain MIME headers, such as Content-Type and Content-Disposition. MIME parameters have a textual representations as in name=value. *They contain important information about the MIME structure of a message, such as the boundary string used, which charset was used to encode the message and so on. This module provides simple to use functions to query or set MIME parameters.*

Each MIME header may hold an arbitrary amount of such parameters, which are delimeted by each other with a semicolon.

- mm_param * **mm_param_new** (void)
- void **mm_param_free** (struct mm_param *param)
- mm_param * **mm_param_generate** (const char *name, const char *value)
- char * **mm_param_setname** (struct mm_param *param, const char *name, int copy)
- char * **mm_param_setvalue** (struct mm_param *param, const char *value, int copy)
- const char * **mm_param_getname** (struct mm_param *param)
- const char * **mm_param_getvalue** (struct mm_param *param)

5.10.1 Detailed Description

Functions to manipulate MIME parameters

5.11 mm_parse.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <ctype.h>
#include <assert.h>
#include "mm_internal.h"
#include "mm_util.h"
#include "mimeparser.h"
#include "mimeparser.tab.h"
```

Functions

- void **PARSER_initialize** (MM_CTX *, int)
- void **PARSER_setbuffer** (const char *)
- void **PARSER_setfp** (FILE *)
- int **mm_parse_mem** (MM_CTX *ctx, const char *text, int parsemode, int flags)
- int **mm_parse_file** (MM_CTX *ctx, const char *filename, int parsemode, int flags)

5.11.1 Detailed Description

Functions to parse MIME messages

5.11.2 Function Documentation

5.11.2.1 int mm_parse_file (MM_CTX * *ctx*, const char * *filename*, int *parsemode*, int *flags*)

Parses a file into a MiniMIME context

Parameters:

ctx A valid MiniMIME context object

filename The name of the file to parse

parsemode The parsemode

flags The flags to pass to the parser

Returns:

0 on success or -1 on failure

Note:

Sets mm_errno if an error occurs

This function parses a MIME message, stored in the filesystem according to the parseflags and stores the results in the MiniMIME context specified by ctx.

The following modes can be used to specify how the message should be parsed:

- MM_PARSE_STRICT: Do not tolerate MIME violations
- MM_PARSE_LOOSE: Tolerate as much MIME violations as possible

The context needs to be initialized before using **mm_context_new()** (p. 17) and may be freed using **mm_context_free()** (p. 16).

5.11.2.2 int mm_parse_mem (MM_CTX * *ctx*, const char * *text*, int *parsemode*, int *flags*)

Parses a NUL-terminated string into a MiniMIME context

Parameters:

ctx A valid MiniMIME context object

text The NUL-terminated string to parse

parsemode The parsemode

flags The flags to pass to the parser

Returns:

0 on success or -1 on failure

Note:

Sets mm_errno if an error occurs

This function parses a MIME message, stored in the memory region pointed to by text (must be NUL-terminated) according to the parseflags and stores the results in the MiniMIME context specified by ctx.

The following modes can be used to specify how the message should be parsed:

- MM_PARSE_STRICT: Do not tolerate MIME violations
- MM_PARSE_LOOSE: Tolerate as much MIME violations as possible

The context needs to be initialized before using **mm_context_new()** (p. 17) and may be freed using **mm_context_free()** (p. 16).

5.11.2.3 void PARSER_initialize (MM_CTX * *newctx*, int *mode*)

Initializes the parser engine.

5.12 mm_util.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <assert.h>
#include "mm_internal.h"
```

Functions

- void * **xmalloc** (size_t size)
- void * **xrealloc** (void *p, size_t size)
- char * **xstrdup** (const char *str)
- void **xfree** (void *p)
- char * **mm_unquote** (const char *string)
- char * **mm_uncomment** (const char *string)
- char * **xstrsep** (char **stringp, const char *delim)
- char * **mm_stripchars** (char *input, char *strip)
- char * **mm_addchars** (char *input, char *add, u_int16_t linelength)
- void **mm_striptrailing** (char **what, const char *charset)

5.12.1 Detailed Description

This module contains utility functions for the MiniMIME library

Chapter 6

MiniMIME Page Documentation

6.1 Bug List

Global mm_content_setmaintype (p. 13) The xfree() call could lead to undesirable results. Do we really need it?

Global mm_content_setsubtype (p. 13) The xfree() call could lead to undesirable results. Do we really need it?

Index