masc_append_dc_key_value

Name

`masc_append_dc_key_value` - Append a key/value pair to a data characteristic.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_append_dc_key_value( struct mas_data_characteristic* dc,
char* key, char* value );
```

Description

`masc_append_dc_key_value` appends *`key`* and *`value`* to *`dc`*. The strings are bytecopied into the dc, and memory is allocated to hold them.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory or there isn’t enough allocated space left in the dc arrays

Examples

See Also

masc_alloc_package

Name

`masc_alloc_package` - Initialize and allocate memory for package contents.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_alloc_package( struct mas_package* package, int size_guess );
```

Description

Allocates *`size_guess`* bytes of memory for the package *`contents`* and zeroes the other *`mas_package`* structure members.

NOTE *Specifying a zero size_guess results in a small, non-zero memory allocation for the package contents.*

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory

Examples

See Also

masc_alloc_package_from_payload

Name

`masc_alloc_package_from_payload` - Re-initialize a package object from a pre-specified payload.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_alloc_package_from_payload( struct mas_package* package,
char* payload );
```

Description

Given a byte sequence *`payload`* and a package *`package`*, this function re-initializes the package, using the payload as its contents. `masc_alloc_package_from_payload` intreprets the payload and initializes the package members, but uses payload directly, without bytecopying it. Do not free *`payload`* until you have finished using *`package`*. On return, *`package`* contains the re-initialized package.

This function asserts that the payload is non-null.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory

Examples

See Also

masc_append_mas_event

Name

`masc_append_mas_event` - Append an event to an event queue.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_append_mas_event(struct mas_event* head,
struct mas_event* event);
```

Description

Append *`event`* to the end of the linked-list of events that begins with *`head`*.

Return value

Returns

- 0 on success
- MERR_NULLPTR if *`event`* or *`head`* is NULL

Examples

See Also

masc_copy_dc

Name

`masc_copy_dc` - Copy a data characteristic.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_copy_dc( struct mas_data_characteristic* dc_dest,
struct mas_data_characteristic* dc_src);
```

Description

Copy a data characteristic from *`dc_src`* to *`dc_dest`*. This allocates memory for the destination key and value strings, but does not allocate *`dc_dest`* itself.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory
- MERR_INVALID if *`dc_dest`* doesn’t have enough free keys
- MERR_NULLPTR if either *`dc_src`* or *`dc_dest`* are NULL

Examples

See Also

masc_cpu_us

Name

masc_cpu_us - Retrieve the current CPU times for the thread in microseconds.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_cpu_us( uint32* user, uint32* sys );
```

Description

Returns the current total CPU usage in CPU microseconds spent in user-space and kernel-space. The caller must have allocated memory to hold *user* and *sys*.

Return value

Returns

- 0 on success
- MERR_INVALID if platform doesn’t support this function

Examples

See Also

masc_debug_package

Name

`masc_debug_package` - dump the contents of a package to the console.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_debug_package( struct mas_package* package, int recdep );
```

Description

Interprets the contents of a *`package`* and calls `masc_log_message` a number of times to display the contents. If *`recdep`* is non-zero, the function will recursively interpret payload members as nested packages. Use *`recdep`* to control the depth that the function will dive into the package: level 1 interprets only the immediate payload members as nested packages, level 2 will interpret payloads of those payloads as nested packages, and so on.

Return value

- MERR_INVALID if an invalid format code was detected

Example Output

```
0: "" (payload: 28 bytes)
4: interpreting payload as package
   0: "name" (string) "net"
   1: "instance" (int32) 16
1: "" (payload: 32 bytes)
4: interpreting payload as package
   0: "name" (string) "anx_oss"
   1: "instance" (int32) 17
```

See Also

masc_destroy_dc

Name

`masc_destroy_dc` - Frees memory used by a data characteristic.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_destroy_dc( struct mas_data_characteristic* dc );
```

Description

Frees all the memory used by *dc*, including the key and value strings.

Return value

Returns 0.

Examples

See Also

masc_destroy_mas_data

Name

`masc_destroy_mas_data` - Frees the memory used by a data object.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_destroy_mas_data( struct mas_data* data );
```

Description

Frees memory used by *`data`*. Also, this frees *`data->segment`*, unless its *`allocated_length`* is zero.

Return value

Returns

- 0 on success
- MERR_MEMORY if *`data`* is NULL

Examples

See Also

masc_destroy_mas_event

Name

`masc_destroy_mas_event` - Frees memory used by an event object.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_destroy_mas_event( struct mas_event* event );
```

Description

Frees the memory used by *`event`*. Also, this frees the *`action_name`*, *`predicate`*, *`type`*, and *`port_dependencies`* if allocated.

Return value

Returns

- 0 on success
- MERR_NULLPTR if *`event`* is NULL

Examples

See Also

masc_destroy_package

Name

`masc_destroy_package` - Frees the memory used by a package object.

Synopsis

```
#include "mas/mas_common.h"

int32 masc_destroy_package( struct mas_package* package );
```

Description

Frees the memory used by *`package`*. Also, frees the memory used by the *`contents`* member.

NOTE *If you want to avoid freeing* *`contents`* *because you're using it somewhere else, simply null it out prior to calling* `masc_destroy_package`.

Return value

Returns 0.

Examples

See Also

masc_finalize_package

Name

`masc_finalize_package` - Prepare a package for transport.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_finalize_package( struct mas_package* package );
```

Description

Sets and checks internal members of the *`package`* object so that its *`contents`* member can be transported as a raw byte sequence. This must be called after all information has been pushed into the package, but before sending its contents.

NOTE THIS COULD BE OBSOLETED. TAKE A LOOK AT THE FUNCTION.

Return value

Returns 0.

Examples

See Also

masc_get_index_of_key

Name

`masc_get_index_of_key` - Find a key in a data characteristic.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_get_index_of_key( struct mas_data_characteristic* dc,
char* key );
```

Description

Scans *dc* key strings for *key* and returns the index of the first match.

Return value

Returns

- the matching index on success (0 is a valid index.)
- MERR_NOTDEF if no match was found

Examples

See Also

masc_get_short_usec_ts

Name

`masc_get_short_usec_ts` - Retrieve a microsecond timestamp.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_get_short_usec_ts(u_int32* usec);
```

Description

Gets the current system time in microseconds and stores it in the 32-bit, wrapping counter *`*usec`*. You cannot use this function to time events longer than the scheduled rollover (every 71.6 minutes). Memory for *`usec`* must be allocated by the caller.

Return value

Returns

- 0 on success
- MERR_IO if there was an error accessing the real-time clock

Examples

See Also

masc_get_string_index

Name

`masc_get_string_index` - Return the index of a string in an array of strings.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_get_string_index( char* sut, char** strings, int num );
```

Description

Given a string-under-test *`sut`* and an array of strings with *`num`* elements, masc_get_string_index returns the index of the string under test in the array.

MAS uses this function to make it easy to use a `switch/case` construct instead of `if (strcmp)...else if (strcmp)..else if (strcmp)...` when choosing one-of-many strings.

Return value

Returns

- index of string in string array on success
- MERR_NOTDEF if the string under test wasn’t found in the array

Examples

See Also

masc_get_systime

Name

`masc_get_systime` - Get the system time in seconds, with a NTP-like format.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_get_systime(struct mas_ntpval* systime);
```

Description

`masc_get_systime` queries the real-time clock for the current system time and converts it to a 64-bit, fixed point representation of the number of seconds elapsed since 0h UTC on January 1, 1900, as specified by the Network Time Protocol. The value is stored in *`*systime`*, which must be allocated by the caller.

Return value

Returns

- 0 on success
- MERR_NULLPTR if *`systime`* is NULL

Examples

See Also

masc_logerror

Name

`masc_logerror` - Log an error message.

Synopsis

```
#include “mas/mas_common.h”

void masc_logerror(const char* errstring, int32 errnum);
```

Description

Logs the error message associated with the error code *`errnum`*, along with an extra string *`errstring`*. The destination of the logging output is determined by the loglevel of *`errnum`*. If the loglevel is below *`MAS_LOGLEVEL`*, then no error is logged. If the loglevel is less than or equal to *`MAS_LOGLEVEL_INFO`*, then the error is logged to the standard output. Otherwise, the error is logged to the standard error output.

The format of the logging output is also determined by the loglevel.

Return value

Returns 0.

Examples

See Also

masc_make_dc

Name

`masc_make_dc` - Create a data characteristic object.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_make_dc( struct mas_data_characteristic** dc, int16 size );
```

Description

Creates a data characteristic object, allocating *`size`* pairs of potential key/value members. On return, *`*dc`* is a pointer to the new object.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory
- MERR_INVALID if *`size`* is out of bounds

Examples

See Also

masc_make_mas_data

Name

`masc_make_mas_data` - Create a new data object.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_make_mas_data( struct mas_data** data_ptr, int size );
```

Description

Allocates space for a mas_data structure. Also allocates *`size`* bytes of memory for the *`segment`* member if *`size`* is nonzero. The data *`segment`* is assumed to be dynamically allocated. If `masc_make_mas_data` isn’t used to allocate memory for the *`segment`* member, then the caller is responsible for allocating this memory and setting the members *`length`* and *`allocated_length`* to the allocated memory size in bytes. On return, *`*data_ptr`* is a pointer to the new object.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory

Examples

See Also

masc_make_mas_event

Name

`masc_make_mas_event` - Create a new event object.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_make_mas_event( struct mas_event** event_ptr );
```

Description

Allocates space for a mas_event structure and initializes it. On return, *`*event_ptr`* is a pointer to the new object.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory

Examples

See Also

masc_make_mas_package

Name

`masc_make_mas_package` - Create a new package object.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_make_package( struct mas_package** package_ptr,
 int size_guess );
```

Description

Allocates memory for a mas_package structure and initializes its members. Also allocates *`size_guess`* bytes of memory for the *`contents`* member. On return, *`*package_ptr`* is a pointer to the new object.

NOTE *Specifying a zero* *`size_guess`* *results in a small, non-zero memory allocation for the package contents.*

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory

Examples

See Also

masc_make_package_from_payload

Name

`masc_make_package_from_payload` - Create a new package object from a pre-specified payload.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_make_package_from_payload( struct mas_package** package_ptr,
char* payload );
```

Description

Given a byte sequence *`payload`*, this function builds a package around it. `masc_make_package_from_payload` allocates memory for the package and initializes its members but uses payload directly, without bytecopying it. Do not free *`payload`* until you have finished using *`*package_ptr`*. On return, *`*package_ptr`* is a pointer to the new object.

This function asserts that the payload is non-null.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory

Examples

See Also

masc_nanosleep

Name

`masc_nanosleep` - Create a new data object.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_nanosleep(u_int32 nsec);
```

Description

Puts the process or thread to sleep for *`nsec`* nanoseconds. This could have rather coarse resolution on some systems, and it is not calibrated like `masc_realsleep`.

Return value

Returns 0.

Examples

See Also

masc_ntp_to_double

Name

`masc_ntp_to_double` - Convert a 64-bit NTP-style timestamp to a double.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_ntp_to_double(struct mas_ntpval* ntptime, double* double-
time);
```

Description

Transforms the 64-bit fixed-point NTP timestamp integer into a floating-point double-precision number and returns the result in the value pointed to by *`doubletime`*. The mas_ntpval structure is used to hold the integer for compatibility with non-64-bit platforms. Caller must have allocated enough memory to hold *`doubletime`*.

Some accuracy may be lost in the translation.

Return value

Returns 0.

Examples

See Also

masc_ntp32_to_double

Name

`masc_ntp32_to_double` - Convert a 32-bit NTP-style timestamp to a double.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_ntp32_to_double(uint32 ntptime, double* doubletime);
```

Description

Transforms the 32-bit fixed-point NTP timestamp integer into a floating-point double-precision number and returns the result in the value pointed to by *`doubletime`*. Caller must have allocated enough memory to hold *`doubletime`*.

Some accuracy may be lost in the transformation.

Return value

Returns 0.

Examples

See Also

masc_timeval_to_double

Name

`masc_timeval_to_double` - Convert a POSIX timeval timestamp to a double.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_timeval_to_double(struct timeval* tv, double* doubletime);
```

Description

Transforms the timeval structure into a floating-point double-precision number of seconds since Midnight, January 1, 1900, and returns the result in the value pointed to by *`doubletime`*. Caller must have allocated enough memory to hold *`doubletime`*.

Some accuracy may be lost in the transformation.

NOTE This function is only defined on POSIX systems.

Return value

Returns 0.

Examples

See Also

masc_package_dc

Name

masc_package_dc - Make a package from a data characteristic.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_package_dc( struct mas_package** package_ptr, struct
mas_data_characteristic* dc  );
```

Description

This is the standard function for packing a data characteristic object into a package. All strings from *dc* are bytecopied. This should be paired with masc_unpack_dc at the receiving end. On return, **package_ptr* is a pointer to the new package.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory

Examples

See Also

masc_unpack_dc

masc_package_mas_event

Name

`masc_package_mas_event` - Make a package from an event.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_package_mas_event( struct mas_package** package_ptr,
struct mas_event* event  );
```

Description

This is the standard function for packing an event object into a package. All members from *`event`* are bytecopied. This should be paired with `masc_unpack_event` at the receiving end. On return, *`*package_ptr`* is a pointer to the new package.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory

Examples

See Also

`masc_unpack_event`

masc_peek_format

Name

`masc_peek_format` - Get the format code of the next package member.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_peek_format( struct mas_package* package, char* retval_fmt );
```

Description

Non-invasively grabs the format code of the next package member and sets *retval_fmt to its value. The caller must allocate memory for `*retval_fmt` to hold the code.

The function asserts that the package is not null.

Return value

Returns

- 0 on success
- MERR_INVALID if the package is null

Examples

See Also

masc_peek_key

Name

`masc_peek_key` - Get the key string of the next package member.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_peek_key( struct mas_package* package, char** key_ptr );
```

Description

Non-invasively grabs the key string of the next package member and sets *key_ptr to point to its start. It doesn’t bytecopy the string, rather, `*key_fmt` is set to the start of the key string in the package contents. The caller must NOT free `*key_ptr` when it is done with it.

The function asserts that the package is not null.

Return value

Returns

- 0 on success
- MERR_INVALID if the package is null

Examples

See Also

masc_print_dc

Name

`masc_print_dc` - Print out the members of a data characteristic for debugging.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_print_dc( struct mas_data_characteristic* dc );
```

Description

Prints out all key/value pairs in *dc*.

Return value

Returns 0.

Examples

See Also

masc_populate_dc

Name

`masc_populate_dc` - Put a variable number of key/value pairs into a dc.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_populate_dc( struct mas_data_characteristic** dc, int npairs,
... );
```

Description

`masc_populate_dc` calls `masc_append_dc_key_value` *`npairs`* times to add keys and values to dc. The strings are bytecopied into the dc, and memory is allocated to hold them. Keys and values are specified in pairs following the npairs argument `(key1, value1, key2, value2, ..., keyN, valueN)`

NOTE Expect weirdness if there are fewer than (`npairs * 2`) arguments following the `npairs` argument.

Return value

- 0 on success
- MERR_MEMORY if there isn’t enough memory or there isn’t enough allocated space left in the dc arrays

Examples

```
masc_populate_dc( &dc, 2, key1, value1, key2, value2 );
```

See Also

masc_pull_double

Name

`masc_pull_double` - Pull a double out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_double( struct mas_package* package, double* retval );
```

Description

Pulls a double out of package *`package`*. One *pulls* members out of a package in the same order as one *pushed* them in. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pull` functions call `mas_assert` to insure the proper type of the next member in the package. Key strings in the package are ignored. When using these functions in conjunction with any of the `masc_pullk` functions, the member to be pulled immediately follows the last member pulled.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double

Examples

See Also

masc_pull_float

Name

`masc_pull_float` - Pull a float out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_float( struct mas_package* package, float* retval );
```

Description

Pulls a float out of package *`package`*. One *pulls* members out of a package in the same order as one *pushed* them in. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pull` functions call `mas_assert` to insure the proper type of the next member in the package. Key strings in the package are ignored. When using these functions in conjunction with any of the `masc_pullk` functions, the member to be pulled immediately follows the last member pulled.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t float

Examples

See Also

masc_pull_int16

Name

`masc_pull_int16` - Pull an int16 out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_int16( struct mas_package* package, int16* retval );
```

Description

Pulls an int16 out of package *`package`*. One *pulls* members out of a package in the same order as one *pushed* them in. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pull` functions call `mas_assert` to insure the proper type of the next member in the package. Key strings in the package are ignored. When using these functions in conjunction with any of the `masc_pullk` functions, the member to be pulled immediately follows the last member pulled.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t int16

Examples

See Also

masc_pull_int32

Name

`masc_pull_int32` - Pull an int32 out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_int32( struct mas_package* package, int32* retval );
```

Description

Pulls an int32 out of package *`package`*. One *pulls* members out of a package in the same order as one *pushed* them in. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pull` functions call `mas_assert` to insure the proper type of the next member in the package. Key strings in the package are ignored. When using these functions in conjunction with any of the `masc_pullk` functions, the member to be pulled immediately follows the last member pulled.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t int32

Examples

See Also

masc_pull_int8

Name

`masc_pull_int8` - Pull an int8 out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_int8( struct mas_package* package, int8* retval );
```

Description

Pulls an int8 out of package *`package`*. One *pulls* members out of a package in the same order as one *pushed* them in. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pull` functions call `mas_assert` to insure the proper type of the next member in the package. Key strings in the package are ignored. When using these functions in conjunction with any of the `mas_pullk` functions, the member to be pulled immediately follows the last member pulled.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t int8

Examples

See Also

masc_pull_payload

Name

`masc_pull_payload` - Pull a payload out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_payload( struct mas_package* package, void**
payload_retval, uint32* len_retval );
```

Description

Pulls a payload out of package *`package`*. One *pulls* members out of a package in the same order as one *pushed* them in. This function allocates memory to hold the payload. The payload is returned in *`*payload_retval`* and its length in bytes is returned in *`*len_retval`*.

The `masc_pull` functions call `mas_assert` to insure the proper type of the next member in the package. Key strings in the package are ignored. When using these functions in conjunction with any of the `masc_pullk` functions, the member to be pulled immediately follows the last member pulled.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory for the payload
- MERR_INVALID if the next member in the package isn’t payload

Examples

See Also

masc_pull_string

Name

`masc_pull_string` - Pull a string out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_string( struct mas_package* package, char**
string_retval);
```

Description

Pulls a string out of package *`package`*. One *pulls* members out of a package in the same order as one *pushed* them in. This function allocates memory to hold the string, returning it in *`*string_retval`*.

The `masc_pull` functions call `mas_assert` to insure the proper type of the next member in the package. Key strings in the package are ignored. When using these functions in conjunction with any of the `masc_pullk` functions, the member to be pulled immediately follows the last member pulled.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory to hold the string
- MERR_INVALID if the next member in the package isn’t string

Examples

See Also

masc_pull_uint16

Name

`masc_pull_uint16` - Pull a uint16 out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_uint16( struct mas_package* package, uint16* retval );
```

Description

Pulls a uint16 out of package *`package`*. One *pulls* members out of a package in the same order as one *pushed* them in. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pull` functions call `mas_assert` to insure the proper type of the next member in the package. Key strings in the package are ignored. When using these functions in conjunction with any of the `masc_pullk` functions, the member to be pulled immediately follows the last member pulled.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t uint16

Examples

See Also

masc_pull_uint32

Name

`masc_pull_uint32` - Pull a uint32 out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_uint32( struct mas_package* package, uint32* retval );
```

Description

Pulls a uint32 out of package *`package`*. One *pulls* members out of a package in the same order as one *pushed* them in. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pull` functions call `mas_assert` to insure the proper type of the next member in the package. Key strings in the package are ignored. When using these functions in conjunction with any of the `masc_pullk` functions, the member to be pulled immediately follows the last member pulled.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t uint32

Examples

See Also

masc_pull_uint8

Name

`masc_pull_uint8` - Pull a uint8 out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_uint8( struct mas_package* package, uint8* retval );
```

Description

Pulls a uint8 out of package *`package`*. One *pulls* members out of a package in the same order as one *pushed* them in. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pull` functions call `mas_assert` to insure the proper type of the next member in the package. Key strings in the package are ignored. When using these functions in conjunction with any of the `masc_pullk` functions, the member to be pulled immediately follows the last member pulled.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t uint8

Examples

See Also

masc_pullk_double

Name

`masc_pullk_double` - Find a key in a package and pull a double from that location.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pullk_double( struct mas_package* package, char* key, double*
retval );
```

Description

Pulls a double with matching key string *`key`* out of package *`package`*. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pullk` functions call `mas_assert` to insure that the specified key is in the package, then call `mas_assert` again to insure the proper type of the requested member in the package. Use `masc_test_key` prior to this call to safely verify the presence of a key.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double
- MERR_NOTDEF if the key isn’t defined in the package

Examples

See Also

masc_pullk_float

Name

`masc_pullk_float` - Find a key in a package and pull a float from that location.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pullk_float( struct mas_package* package, char* key, float*
retval );
```

Description

Pulls a float with matching key string *`key`* out of package *`package`*. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pullk` functions call `mas_assert` to insure that the specified key is in the package, then call `mas_assert` again to insure the proper type of the requested member in the package. Use `masc_test_key` prior to this call to safely verify the presence of a key.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double
- MERR_NOTDEF if the key isn’t defined in the package

Examples

See Also

masc_pullk_int16

Name

`masc_pullk_int16` - Find a key in a package and pull an int16 from that location.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pullk_int16( struct mas_package* package, char* key, int16*
retval );
```

Description

Pulls a int16 with matching key string *`key`* out of package *`package`*. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pullk` functions call `mas_assert` to insure that the specified key is in the package, then call `mas_assert` again to insure the proper type of the requested member in the package. Use `masc_test_key` prior to this call to safely verify the presence of a key.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double
- MERR_NOTDEF if the key isn’t defined in the package

Examples

See Also

masc_pullk_int32

Name

`masc_pullk_int32` - Find a key in a package and pull an int32 from that location.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pullk_int32( struct mas_package* package, char* key, int32*
retval );
```

Description

Pulls an int32 with matching key string *`key`* out of package *`package`*. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pullk` functions call `mas_assert` to insure that the specified key is in the package, then call `mas_assert` again to insure the proper type of the requested member in the package. Use `masc_test_key` prior to this call to safely verify the presence of a key.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double
- MERR_NOTDEF if the key isn’t defined in the package

Examples

See Also

masc_pullk_int8

Name

`masc_pullk_int8` - Find a key in a package and pull an int8 from that location.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pullk_int8( struct mas_package* package, char* key, int8*
retval );
```

Description

Pulls an int8 with matching key string *`key`* out of package *`package`*. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pullk` functions call `mas_assert` to insure that the specified key is in the package, then call `mas_assert` again to insure the proper type of the requested member in the package. Use `masc_test_key` prior to this call to safely verify the presence of a key.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double
- MERR_NOTDEF if the key isn’t defined in the package

Examples

See Also

masc_pullk_payload

Name

masc_pullk_payload - Find a key in a package and pull a payload from that location.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pullk_payload( struct mas_package* package, char* key, void**
payload_retval, uint32* len_retval );
```

Description

Pulls a payload with matching key string *key* out of package *package*. The payload is returned in **payload_retval* and its length in bytes is returned in **len_retval*. This function allocates memory to hold the payload, but the caller must allocate memory for **len_retval*.

The masc_pullk functions call mas_assert to insure that the specified key is in the package, then call mas_assert again to insure the proper type of the requested member in the package. Use masc_test_key prior to this call to safely verify the presence of a key.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double
- MERR_NOTDEF if the key isn’t defined in the package

Examples

See Also

masc_pullk_string

Name

`masc_pullk_string` - Find a key in a package and pull a string from that location.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pullk_string( struct mas_package* package, char* key, char**
retval );
```

Description

Pulls a string with matching key string *`key`* out of package *`package`*. This function allocates memory to hold the string, returning it in *`*string_retval`*.

The `masc_pullk` functions call `mas_assert` to insure that the specified key is in the package, then call `mas_assert` again to insure the proper type of the requested member in the package. Use `masc_test_key` prior to this call to safely verify the presence of a key.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double
- MERR_NOTDEF if the key isn’t defined in the package

Examples

See Also

masc_pullk_uint16

Name

`masc_pullk_uint16` - Find a key in a package and pull a uint32 from that location.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pullk_uint16( struct mas_package* package, char* key, uint16*
retval );
```

Description

Pulls a uint16 with matching key string *`key`* out of package *`package`*. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pullk` functions call `mas_assert` to insure that the specified key is in the package, then call `mas_assert` again to insure the proper type of the requested member in the package. Use `masc_test_key` prior to this call to safely verify the presence of a key.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double
- MERR_NOTDEF if the key isn’t defined in the package

Examples

See Also

masc_pullk_uint32

Name

`masc_pullk_uint32` - Find a key in a package and pull a uint32 from that location.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pullk_uint32( struct mas_package* package, char* key, uint32*
retval );
```

Description

Pulls a uint32 with matching key string *`key`* out of package *`package`*. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pullk` functions call `mas_assert` to insure that the specified key is in the package, then call `mas_assert` again to insure the proper type of the requested member in the package. Use `masc_test_key` prior to this call to safely verify the presence of a key.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double
- MERR_NOTDEF if the key isn’t defined in the package

Examples

See Also

masc_pullk_uint8

Name

`masc_pullk_uint8` - Find a key in a package and pull a uint8 from that location.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pullk_uint8( struct mas_package* package, char* key, uint8*
retval );
```

Description

Pulls a uint8 with matching key string *`key`* out of package *`package`*. The caller must allocate memory for *`*retval`* to hold the data.

The `masc_pullk` functions call `mas_assert` to insure that the specified key is in the package, then call `mas_assert` again to insure the proper type of the requested member in the package. Use `masc_test_key` prior to this call to safely verify the presence of a key.

Return value

Returns

- 0 on success
- MERR_INVALID if the type in the package isn’t double
- MERR_NOTDEF if the key isn’t defined in the package

Examples

See Also

masc_push_double

Name

`masc_push_double` - Push a double into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_double( struct mas_package* package, double val );
```

Description

Pushes a double into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The pushed member’s key string is empty. When using these functions in conjunction with any of the `masc_pushk` functions, the new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_push_float

Name

`masc_push_float` - Push a float into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_float( struct mas_package* package, float val );
```

Description

Pushes a float into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The pushed member’s key string is empty. When using these functions in conjunction with any of the `masc_pushk` functions, the new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_push_int16

Name

`masc_push_int16` - Push an int16 into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_int16( struct mas_package* package, int16 val );
```

Description

Pushes an int16 into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The pushed member’s key string is empty. When using these functions in conjunction with any of the `masc_pushk` functions, the new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_push_int32

Name

`masc_push_int32` - Push an int32 into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_int32( struct mas_package* package, int32 val );
```

Description

Pushes an int32 into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The pushed member’s key string is empty. When using these functions in conjunction with any of the `masc_pushk` functions, the new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_push_int8

Name

`masc_push_int8` - Push an int8 into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_int8( struct mas_package* package, double val );
```

Description

Pushes an int8 into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The pushed member’s key string is empty. When using these functions in conjunction with any of the `masc_pushk` functions, the new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pull_members

Name

`masc_pull_members` - Pull a number of members out of a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pull_members( struct mas_package* package, char* format,
... );
```

Description

`masc_pull_members` allows one to pull any number of members out of a package using a single (although hellish) function call. It uses the *`format`* string to identify the types and number of the package members. The members are returned in the values pointed to by the arguments immediately following *`format`*. The caller must allocate enough memory for each value, except for strings and payload arguments (although memory must be allocated for the payload length argument.)

The characters in the format string identify the types of the package members in the order they were passed. The first character defines the type of the first member, the second character defines the type of the second member, and so on. The following table specifies the mapping of character code to member type:

character code	member type	argument
c	int8	int8
C	uint8	uint8
s	int16	int16
S	uint16	uint16
l	int32	int32
L	uint32	uint32
p	payload	char* payload, uint32 length
a	ASCII string	char*
f	float	float
d	double	double

The pointers for the package members' values are taken one at a time from the stack, immediately following the format argument. Each value requires one argument, with one exception: the payload type requires the payload double-char-pointer followed by a pointer to a 32-bit unsigned integer length.

The masc_pull_* functions are used to push the members into the package. Check their documentation to avoid problems.

The function asserts that there are no invalid format codes.

Expect weirdness if there are fewer than `strlen(`*`format`*`)` arguments following the *`format`* argument, give or take a few payload arguments.

Return value

- 0 on success

Examples

Pull an unsigned, 8-bit integer from a package.

```
masc_pull_members( package, "C", &num  );
```

Pull a double, a payload, and a string from a package:

```
masc_push_members( package, "dpa", &mean, &payload, &payload_len,
&string );
```

See Also

"masc_push_members" on page 60

masc_push_members

Name

`masc_push_members` - Push a variable number of members into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_members( struct mas_package* package, char* format,
... );
```

Description

`masc_push_members` allows one to push any number of members into a package using a single (although hellish) function call. It uses the *`format`* string to identify the types and number of the package members. The members are then specified as arguments immediately following *`format`*.

The characters in the format string identify the types of the package members in the order they were passed. The first character defines the type of the first member, the second character defines the type of the second member, and so on. The following table specifies the mapping of character code to member type:

character code	member type	argument
c	int8	int8
C	uint8	uint8
s	int16	int16
S	uint16	uint16
l	int32	int32
L	uint32	uint32
p	payload	char* payload, uint32 length
a	ASCII string	char*
f	float	float
d	double	double

The values of the package members are taken one at a time from the stack, immediately following the format argument. Each value requires one argument, with one exception: the payload type requires the payload pointer followed by a 32-bit unsigned integer length.

The function asserts that there are fewer than 64 members to be pushed.

The masc_push_* functions are used by this function to push the members into the package. Check their documentation to avoid problems.

The function asserts that there are no invalid format codes.

Expect weirdness if there are fewer than `strlen(`*`format`*`)` arguments following the *`format`* argument, give or take a few payload arguments.

Return value

- 0 on success
- MERR_MEMORY if memory can't be allocated

Examples

Push an unsigned, 8-bit integer into a package.

```
masc_push_members( package, "C", "a number", num  );
```

Push a double, a payload, and a string into a package:

```
masc_push_members( package, "dpa", "mean", mean, "my stuff", payload,
payload_len, "happy string", string );
```

See Also

"masc_pushk_members" on page 62

masc_pushk_members

Name

`masc_pushk_members` - Push a number of members and keys into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_members( struct mas_package* package, char* format,
... );
```

Description

`masc_pushk_members` allows one to push any number of members with key strings into a package using a single (although hellish) function call. It uses the *`format`* string to identify the types and number of the package members. The key strings and members are then specified as arguments immediately following *`format`*.

The format string functions identically to the format string of “masc_push_members” on page 60. Refer there for a description.

Following the format string, each member is specified with a pair of arguments: `(char* key_string, [member_type] value)`. The one exception is the payload type that requires three arguments: `(key_string, char* payload, uint32 payload_length)`.

The function asserts that there are fewer than 64 members to be pushed.

The masc_pushk_* functions are used by this function to push the members into the package. Check their documentation to avoid problems.

The function asserts that there are no invalid format codes.

Expect weirdness if there are fewer than `strlen(`*`format`*`)*2` arguments following the *`format`* argument, give or take a few payload arguments.

Return value

- 0 on success
- MERR_MEMORY if memory can’t be allocated

Examples

Push an unsigned, 8-bit integer into a package.

```
masc_push_members( package, “C”, num  );
```

Push a string and a 32-bit integer into a package:

```
masc_push_members( package, “al”, string, num  );
```

Push a double, a payload, and a string into a package:

```
masc_push_members( package, “dpa”, mean, payload, payload_len, string );
```

See Also

“masc_push_members” on page 60

masc_push_payload

Name

masc_push_payload - Push a payload byte sequence into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_payload( struct mas_package* package, void* payload,
uint32 len );
```

Description

Pushes a byte sequence *payload* of length *len* into package *package*. If there isn’t enough memory in the package to hold the payload, more memory is allocated to the package using an exponential growth strategy.

The pushed member’s key string is empty. When using these functions in conjunction with any of the masc_pushk functions, the new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_push_string

Name

`masc_push_string` - Push a string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_string( struct mas_package* package, char* string );
```

Description

Pushes a character string *`string`* into package *`package`*. If there isn’t enough memory in the package to hold the string, more memory is allocated to the package using an exponential growth strategy.

The pushed member’s key string is empty. When using these functions in conjunction with any of the `masc_pushk` functions, the new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_push_strings

Name

`masc_push_strings` - Push many strings into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_strings( struct mas_package* package, char** strings,
int num );
```

Description

Calls `masc_push_string` *`num`* times to push strings from the string array *`strings`* into the specified package. This function is especially useful to push lists into pacakges.

Consult the masc_push_string documentation for additional information.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

“masc_push_string” on page 65

masc_push_uint16

Name

`masc_push_uint16` - Push a uint16 into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_uint16( struct mas_package* package, uint16 val );
```

Description

Pushes a uint16 into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The pushed member’s key string is empty. When using these functions in conjunction with any of the `masc_pushk` functions, the new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_push_uint32

Name

`masc_push_uint32` - Push a uint32 into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_uint32( struct mas_package* package, uint32 val );
```

Description

Pushes a uint32 into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The pushed member’s key string is empty. When using these functions in conjunction with any of the `masc_pushk` functions, the new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_push_uint8

Name

`masc_push_uint8` - Push a uint8 into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_push_uint8( struct mas_package* package, uint8 val );
```

Description

Pushes a uint8 into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The pushed member’s key string is empty. When using these functions in conjunction with any of the `masc_pushk` functions, the new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pushk_double

Name

`masc_pushk_double` - Push a double with a key string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_double( struct mas_package* package, char* key, double
val );
```

Description

Pushes a double and its identifying string *`key`* into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pushk_float

Name

`masc_pushk_float` - Push a float with a key string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_float( struct mas_package* package, char* key, float
val );
```

Description

Pushes a float and its identifying string *`key`* into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pushk_int16

Name

`masc_pushk_int16` - Push an int16 with a key string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_int16( struct mas_package* package, char* key, int16
val );
```

Description

Pushes an int16 and its identifying string *`key`* into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pushk_int32

Name

`masc_pushk_int32` - Push an int32 with a key string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_int32( struct mas_package* package, char* key, int32
val );
```

Description

Pushes an int32 and its identifying string *`key`* into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pushk_int8

Name

`masc_pushk_int8` - Push an int8 with a key string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_int8( struct mas_package* package, char* key, double
val );
```

Description

Pushes an int8 and its identifying string *`key`* into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pushk_payload

Name

`masc_pushk_payload` - Push a payload byte sequence with a key string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_payload( struct mas_package* package, char* key, void*
payload, uint32 len );
```

Description

Pushes a byte sequence *`payload`* of length *`len`* and its identifying string *`key`* into package *`package`*. If there isn’t enough memory in the package to hold the payload, more memory is allocated to the package using an exponential growth strategy.

The new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pushk_string

Name

`masc_pushk_string` - Push a string with a key string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_string( struct mas_package* package, char* key, char*
string );
```

Description

Pushes a character string *`string`* and its identifying string *`key`* into package *`package`*. If there isn’t enough memory in the package to hold the string, more memory is allocated to the package using an exponential growth strategy.

The new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pushk_uint16

Name

`masc_pushk_uint16` - Push a uint16 with a key string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_uint16( struct mas_package* package, char* key, uint16
val );
```

Description

Pushes a uint16 and its identifying string *`key`* into package *`package`*. If there isn't enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pushk_uint32

Name

`masc_pushk_uint32` - Push a uint32 with a key string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_uint32( struct mas_package* package, char* key, uint32
val );
```

Description

Pushes a uint32 and its identifying string *`key`* into package *`package`*. If there isn’t enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_pushk_uint8

Name

`masc_pushk_uint8` - Push a uint8 with a key string into a package.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_pushk_uint8( struct mas_package* package, char* key, uint8
val );
```

Description

Pushes a uint8 and its identifying string *`key`* into package *`package`*. If there isn't enough memory in the package to hold *`val`*, more memory is allocated to the package using an exponential growth strategy.

The new member immediately follows the last member pushed.

Return value

Returns

- 0 on success
- MERR_MEMORY if memory cannot be allocated

Examples

See Also

masc_realsleep

Name

`masc_realsleep` - System-calibrated nanosleep replacement.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_realsleep(u_int32 nsec);
```

Description

This is a front end for `mas_nanosleep` that takes advantage of compiled in tables for nanosleep values and actual average time slept. This function tries to sleep for time less than or equal to the specified time in nanoseconds, *`nsec`*. If you ask it to sleep for 27 milleseconds (ms), and it knows how to sleep for either 20ms or 30ms, it will sleep for 20ms.

Some Linux kernels always sleep one kernel scheduler tick longer than requested, typically about 10ms. This function compensates for that behavior.

Return value

Returns 0.

Examples

See Also

masc_reset_package

Name

`masc_reset_package` - Clears the members of a package object.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_reset_package( struct mas_package* package );
```

Description

This zeroes all of the members of *`package`* so that it can be reused. The *`contents`* member will be lost; if its allocated memory hasn’t been freed, it will cause a memory leak.

Return value

Returns 0.

Examples

See Also

masc_rtalloc

Name
`masc_rtalloc` - Real time memory allocation.

Synopsis
```
#include “mas/mas_common.h”

void* masc_rtalloc(size_t size);
```

Description
This is currently a wrapper around `malloc` on all platforms. Eventually, one prays for real-time sensitive heap management. For now, `malloc` is good enough.

`masc_rtalloc` asserts that memory can be allocated.

Return value
Returns what `malloc` returns.

Examples

See Also

masc_rtfree

Name

`masc_rtfree` - Real-time memory de-allocation.

Synopsis

```
#include “mas/mas_common.h”

void masc_rtfree(void* ptr);
```

Description

This is currently a wrapper around `free` on all platforms. Eventually, one prays for real-time sensitive heap management. For now, `free` is good enough.

`masc_rtfree` asserts that *`ptr`* is not null.

Return value

Returns what `free` returns.

Examples

See Also

masc_rtrealloc

Name

`masc_rtrealloc` - Real-time memory re-allocation.

Synopsis

```
#include “mas/mas_common.h”

void masc_rtrealloc(void* ptr, size_t size);
```

Description

This is currently a wrapper around `realloc` on all platforms. Eventually, one prays for real-time sensitive heap management. For now, `realloc` is good enough.

`masc_rtrealloc` asserts that memory was allocated by `realloc`.

Return value

Returns what `realloc` returns.

Examples

See Also

masc_strerrorlayer

Name

`masc_strerrorlayer` - Maps an error code's error layer to a string.

Synopsis

```
#include "mas/mas_common.h"

const char* masc_strerrorlayer(int32 error);
```

Description

This returns a string that describes the error layer of *`error`*.

Return value

Returns the error layer string.

Examples

See Also

masc_strmerror

Name

`masc_strmerror` - Map the MAS error code to a descriptive string.

Synopsis

```
#include “mas/mas_common.h”

const char* masc_strmerror(int32 error);
```

Description

Gets a string description of the MAS errorcode in *`error`*.

Return value

Returns the error string.

Examples

See Also

masc_strserror

Name

`masc_strserror` - Map the system error code to a descriptive string.

Synopsis

```
#include “mas/mas_common.h”

char* masc_strserror(int32 error);
```

Description

Part of the error code can map to a system-specific code. On POSIX systems, this code (the serror) is used to hold the value of *`errno`* at the time the error occurred; for these systems, `masc_strserror` is simply a wrapper around `strerror`.

Return value

Returns the error string.

Examples

See Also

masc_test_key

Name

`masc_test_key` - Is this key in this package?

Synopsis

```
#include “mas/mas_common.h”

int32 masc_test_key( struct mas_package* package, char* key );
```

Description

Non-invasively tests the package for the existence of the specified key string *`key`*. If the key string is not found, an error is returned. If the key string is found, zero (0) is returned.

Return value

Returns

- 0 on success
- MERR_NOTDEF if the key isn’t found

Examples

See Also

masc_trim_string

Name

`masc_trim_string` - Trims leading and trailing whitespace from a string.

Synopsis

```
#include “mas/mas_common.h”

void masc_trim_string(char* string);
```

Description

Trim the whitespace from the head and tail of *`string`*. This does in-place operation with character shifting; it can’t be used on const strings.

Return value

none

Examples

See Also

masc_unpack_dc

Name

masc_unpack_dc - Unpackage a data characteristic from a payload.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_unpack_dc(char* payload, struct mas_data_characteristic* dc);
```

Description

This is the standard function for un-packing a data characteristic object from a payload byte sequence. This should be paired with masc_package_dc at the sending end. The caller must have allocated *dc*.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory

Examples

See Also

masc_unpack_mas_event

Name

`masc_unpack_mas_event` - Unpackage a mas_event from a payload.

Synopsis

```
#include “mas/mas_common.h”

int32 masc_unpack_mas_event( char* payload, struct mas_event* event );
```

Description

This is the standard function for unpacking an event object from a payload byte sequence. This should be paired with `masc_package_mas_event` at the sending end. The caller must have allocated *`event`*.

Return value

Returns

- 0 on success
- MERR_MEMORY if there isn’t enough memory

Examples

See Also

`masc_package_mas_event`