NAME

mbuf_tags - a framework for generic packet attributes

SYNOPSIS

#include <sys/mbuf.h>

*struct m_tag **
m_tag_alloc(*uint32_t cookie, uint16_t type, int len, int wait*);

*struct m_tag **
m_tag_copy(*struct m_tag *t, int how*);

int
m_tag_copy_chain(*struct mbuf *to, const struct mbuf *from, int how*);

void
m_tag_delete(*struct mbuf *m, struct m_tag *t*);

void
m_tag_delete_chain(*struct mbuf *m, struct m_tag *t*);

void
m_tag_delete_nonpersistent(*struct mbuf *m*);

*struct m_tag **
m_tag_find(*struct mbuf *m, uint16_t type, struct m_tag *start*);

*struct m_tag **
m_tag_first(*struct mbuf *m*);

void
m_tag_free(*struct m_tag *t*);

*struct m_tag **
m_tag_get(*uint16_t type, int len, int wait*);

void
m_tag_init(*struct mbuf *m*);

*struct m_tag **

m_tag_locate(*struct mbuf *m, uint32_t cookie, uint16_t type, struct m_tag *t*);

*struct m_tag **
m_tag_next(*struct mbuf *m, struct m_tag *t*);

void
m_tag_prepend(*struct mbuf *m, struct m_tag *t*);

void
m_tag_unlink(*struct mbuf *m, struct m_tag *t*);

DESCRIPTION

Mbuf tags allow additional meta-data to be associated with in-flight packets by providing a mechanism for the tagging of additional kernel memory onto packet header mbufs. Tags are maintained in chains off of the mbuf(9) header, and maintained using a series of API calls to allocate, search, and delete tags. Tags are identified using an ID and cookie that uniquely identify a class of data tagged onto the packet, and may contain an arbitrary amount of additional storage. Typical uses of mbuf tags include Mandatory Access Control (MAC) labels as described in mac(9), IPsec policy information as described in ipsec(4), and packet filter tags used by pf(4).

Tags will be maintained across a variety of operations, including the copying of packet headers using facilities such as **M_COPY_PKTHDR**() and **M_MOVE_PKTHDR**(). Any tags associated with an mbuf header will be automatically freed when the mbuf is freed, although some subsystems will wish to delete the tags prior to that time.

Packet tags are used by different kernel APIs to keep track of operations done or scheduled to happen to packets. Each packet tag can be distinguished by its type and cookie. The cookie is used to identify a specific module or API. The packet tags are attached to mbuf packet headers.

The first **sizeof**(*struct m_tag*) bytes of a tag contain a *struct m_tag*:

```
struct m_tag {
        SLIST_ENTRY(m_tag)      m_tag_link;         /* List of packet tags */
        uint16_t                m_tag_id;           /* Tag ID */
        uint16_t                m_tag_len;          /* Length of data */
        uint32_t                m_tag_cookie;       /* ABI/Module ID */
        void                            (*m_tag_free)(struct m_tag *);
};
```

The *m_tag_link* field is used to link tags together (see queue(3) for more details). The *m_tag_id*,

m_tag_len and *m_tag_cookie* fields are set to type, length, and cookie, respectively. *m_tag_free* points to **m_tag_free_default**(). Following this structure are *m_tag_len* bytes of space that can be used to store tag-specific information. Addressing this data region may be tricky. A safe way is embedding *struct m_tag* into a private data structure, as follows:

```
struct foo {
        struct m_tag        tag;
        ...
};
struct foo *p = (struct foo *)m_tag_alloc(...);
struct m_tag *mtag = &p->tag;
```

Note that OpenBSD does not support cookies, it needs *m_tag_id* to be globally unique. To keep compatibility with OpenBSD, a cookie MTAG_ABI_COMPAT is provided along with some compatibility functions. When writing an OpenBSD compatible code, one should be careful not to take already used tag type. Tag types are defined in <*sys/mbuf.h*>.

Packet Tag Manipulation Functions

m_tag_alloc(*cookie*, *type*, *len*, *wait*)
Allocate a new tag of type *type* and cookie *cookie* with *len* bytes of space following the tag header itself. The *wait* argument is passed directly to malloc(9). If successful, **m_tag_alloc**() returns a memory buffer of (*len* + **sizeof**(*struct m_tag*)) bytes. Otherwise, NULL is returned. A compatibility function **m_tag_get**() is also provided.

m_tag_copy(*tag*, *how*)
Allocate a copy of *tag*. The *how* argument is passed directly to **m_tag_alloc**(). The return values are the same as in **m_tag_alloc**().

m_tag_copy_chain(*tombuf*, *frommbuf*, *how*)
Allocate and copy all tags from mbuf *frommbuf* to mbuf *tombuf*. Returns 1 on success, and 0 on failure. In the latter case, mbuf *tombuf* loses all its tags, even previously present.

m_tag_delete(*mbuf*, *tag*)
Remove *tag* from *mbuf*'s list and free it.

m_tag_delete_chain(*mbuf*, *tag*)
Remove and free a packet tag chain, starting from *tag*. If *tag* is NULL, all tags are deleted.

m_tag_delete_nonpersistent(*mbuf*)
Traverse *mbuf*'s tags and delete those which do not have the MTAG_PERSISTENT flag set.

m_tag_first(*mbuf*)
Return the first tag associated with *mbuf*.

m_tag_free(*tag*)
Free *tag* using its *m_tag_free* method. The **m_tag_free_default**() function is used by default.

m_tag_init(*mbuf*)
Initialize the tag storage for packet *mbuf*.

m_tag_locate(*mbuf, cookie, type, tag*)
Search for a tag defined by *type* and *cookie* in *mbuf*, starting from position specified by *tag*. If the latter is NULL, then search through the whole list. Upon success, a pointer to the first found tag is returned. In either case, NULL is returned. A compatibility function **m_tag_find**() is also provided.

m_tag_next(*mbuf, tag*)
Return tag next to *tag* in *mbuf*. If absent, NULL is returned.

m_tag_prepend(*mbuf, tag*)
Add the new tag *tag* at the head of the tag list for packet *mbuf*.

m_tag_unlink(*mbuf, tag*)
Remove tag *tag* from the list of tags of packet *mbuf*.

CODE REFERENCES

The tag-manipulating code is contained in the file *sys/kern/uipc_mbuf2.c*. Inlined functions are defined in <*sys/mbuf.h*>.

SEE ALSO

queue(3), mbuf(9)

HISTORY

The packet tags first appeared in OpenBSD 2.9 and were written by Angelos D. Keromytis <*angelos@openbsd.org*>.