

NAME

au_to_arg32, **au_to_arg64**, **au_to_arg**, **au_to_attr64**, **au_to_data**, **au_to_exit**, **au_to_groups**, **au_to_newgroups**, **au_to_in_addr**, **au_to_in_addr_ex**, **au_to_ip**, **au_to_ipc**, **au_to_ipc_perm**, **au_to_iport**, **au_to_opaque**, **au_to_file**, **au_to_text**, **au_to_path**, **au_to_process32**, **au_to_process64**, **au_to_process**, **au_to_process32_ex**, **au_to_process64_ex**, **au_to_process_ex**, **au_to_return32**, **au_to_return64**, **au_to_return**, **au_to_seq**, **au_to_sock_inet32**, **au_to_sock_inet128**, **au_to_sock_inet**, **au_to_socket_ex**, **au_to_subject32**, **au_to_subject64**, **au_to_subject**, **au_to_subject32_ex**, **au_to_subject64_ex**, **au_to_subject_ex**, **au_to_me**, **au_to_exec_args**, **au_to_exec_env**, **au_to_header**, **au_to_header32**, **au_to_header64**, **au_to_header_ex**, **au_to_header32_ex**, **au_to_trailer**, **au_to_zonename** - routines for generating BSM audit tokens

LIBRARY

Basic Security Module Library (libbsm, -lbsm)

SYNOPSIS

```
#include <bsm/libbsm.h>
```

token_t *

```
au_to_arg32(char n, const char *text, u_int32_t v);
```

token_t *

```
au_to_arg64(char n, const char *text, u_int64_t v);
```

token_t *

```
au_to_arg(char n, const char *text, u_int32_t v);
```

token_t *

```
au_to_attr32(struct vattr *attr);
```

token_t *

```
au_to_attr64(struct vattr *attr);
```

token_t *

```
au_to_attr(struct vattr *attr);
```

token_t *

```
au_to_data(char unit_print, char unit_type, char unit_count, const char *p);
```

token_t *

```
au_to_exit(int retval, int err);
```

token_t *

au_to_groups(*int* *groups);

token_t *

au_to_newgroups(*u_int16_t* n, *gid_t* *groups);

token_t *

au_to_in_addr(*struct in_addr* *internet_addr);

token_t *

au_to_in_addr_ex(*struct in6_addr* *internet_addr);

token_t *

au_to_ip(*struct ip* *ip);

token_t *

au_to_ipc(*char* type, *int* id);

token_t *

au_to_ipc_perm(*struct ipc_perm* *perm);

token_t *

au_to_iport(*u_int16_t* iport);

token_t *

au_to_opaque(*const char* *data, *u_int16_t* bytes);

token_t *

au_to_file(*const char* *file, *struct timeval* tm);

token_t *

au_to_text(*const char* *text);

token_t *

au_to_path(*const char* *text);

token_t *

au_to_process32(*au_id_t* auid, *uid_t* euid, *gid_t* egid, *uid_t* ruid, *gid_t* rgid, *pid_t* pid, *au_asid_t* sid, *au_tid_t* *tid);

token_t *

au_to_process64(*au_id_t* *auid*, *uid_t* *euid*, *gid_t* *egid*, *uid_t* *ruid*, *gid_t* *rgid*, *pid_t* *pid*, *au_asid_t* *sid*,
au_tid_t **tid*);

token_t *

au_to_process32_ex(*au_id_t* *auid*, *uid_t* *euid*, *gid_t* *egid*, *uid_t* *ruid*, *gid_t* *rgid*, *pid_t* *pid*, *au_asid_t* *sid*,
au_tid_addr_t **tid*);

token_t *

au_to_process64_ex(*au_id_t* *auid*, *uid_t* *euid*, *gid_t* *egid*, *uid_t* *ruid*, *gid_t* *rgid*, *pid_t* *pid*, *au_asid_t* *sid*,
au_tid_addr_t **tid*);

token_t *

au_to_return32(*char* *status*, *u_int32_t* *ret*);

token_t *

au_to_return64(*char* *status*, *u_int64_t* *ret*);

token_t *

au_to_return(*char* *status*, *u_int32_t* *ret*);

token_t *

au_to_seq(*long* *audit_count*);

token_t *

au_to_sock_inet32(*struct sockaddr_in* **so*);

token_t *

au_to_sock_inet128(*struct sockaddr_in6* **so*);

token_t *

au_to_sock_int(*struct sockaddr_in* **so*);

token_t *

au_to_socket_ex(*u_short* *so_domain*, *u_short* *so_type*, *struct sockaddr* **sa_local*,
struct sockaddr **sa_remote*);

token_t *

au_to_subject32(*au_id_t* *auid*, *uid_t* *euid*, *gid_t* *egid*, *uid_t* *ruid*, *gid_t* *rgid*, *pid_t* *pid*, *au_asid_t* *sid*,
au_tid_t **tid*);

token_t *

au_to_subject64(*au_id_t* *auid*, *uid_t* *euid*, *gid_t* *egid*, *uid_t* *ruid*, *gid_t* *rgid*, *pid_t* *pid*, *au_asid_t* *sid*,
au_tid_t **tid*);

token_t *

au_to_subject(*au_id_t* *auid*, *uid_t* *euid*, *gid_t* *egid*, *uid_t* *ruid*, *gid_t* *rgid*, *pid_t* *pid*, *au_asid_t* *sid*,
au_tid_t **tid*);

token_t *

au_to_subject32_ex(*au_id_t* *auid*, *uid_t* *euid*, *gid_t* *egid*, *uid_t* *ruid*, *gid_t* *rgid*, *pid_t* *pid*, *au_asid_t* *sid*,
au_tid_addr_t **tid*);

token_t *

au_to_subject64_ex(*au_id_t* *auid*, *uid_t* *euid*, *gid_t* *egid*, *uid_t* *ruid*, *gid_t* *rgid*, *pid_t* *pid*, *au_asid_t* *sid*,
au_tid_addr_t **tid*);

token_t *

au_to_subject_ex(*au_id_t* *auid*, *uid_t* *euid*, *gid_t* *egid*, *uid_t* *ruid*, *gid_t* *rgid*, *pid_t* *pid*, *au_asid_t* *sid*,
au_tid_addr_t **tid*);

token_t *

au_to_me(*void*);

token_t *

au_to_exec_args(*char* ***argv*);

token_t *

au_to_exec_env(*char* ***envp*);

token_t *

au_to_header(*int* *rec_size*, *au_event_t* *e_type*, *au_emod_t* *emod*);

token_t *

au_to_header32(*int* *rec_size*, *au_event_t* *e_type*, *au_emod_t* *emod*);

token_t *

au_to_header64(*int* *rec_size*, *au_event_t* *e_type*, *au_emod_t* *e_mod*);

token_t *

au_to_header_ex(*int* *rec_size*, *au_event_t* *e_type*, *au_emod_t* *e_mod*);

```
token_t *  
au_to_header32_ex(int rec_size, au_event_t e_type, au_emod_t e_mod);
```

```
token_t *  
au_to_trailer(int rec_size);
```

```
token_t *  
au_to_zonename(const char *zonename);
```

DESCRIPTION

These interfaces support the allocation of BSM audit tokens, represented by *token_t*, for various data types.

`au_errno_to_bsm(3)` must be used to convert local `errno(2)` errors to BSM error numbers before they are passed to `au_to_return()`, `au_to_return32()`, and `au_to_return64()`.

RETURN VALUES

On success, a pointer to a *token_t* will be returned; the allocated *token_t* can be freed via a call to `au_free_token(3)`. On failure, NULL will be returned, and an error condition returned via *errno*.

SEE ALSO

`au_errno_to_bsm(3)`, `libbsm(3)`

HISTORY

The OpenBSM implementation was created by McAfee Research, the security division of McAfee Inc., under contract to Apple Computer, Inc., in 2004. It was subsequently adopted by the TrustedBSD Project as the foundation for the OpenBSM distribution.

AUTHORS

This software was created by Robert Watson, Wayne Salamon, and Suresh Krishnaswamy for McAfee Research, the security research division of McAfee, Inc., under contract to Apple Computer, Inc.

The Basic Security Module (BSM) interface to audit records and audit event stream format were defined by Sun Microsystems.