NAME

au_to, au_to_arg, au_to_attr, au_to_data, au_to_groups, au_to_in_addr, au_to_ipc, au_to_ipc_perm, au_to_iport, au_to_me, au_to_opaque, au_to_path, au_to_process, au_to_return, au_to_socket, au_to_text – create audit record tokens

SYNOPSIS

cc [*flag* ...] *file* ... **–lbsm –lsocket –lnsl –lintl** [*library* ...]

#include <sys/types.h>
#include <sys/vnode.h>
#include <netinet/in.h>
#include <bsm/libbsm.h>

token_t *au_to_arg(char *n,* **char** **text,* **u_long** *v***);**

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

token_t *au_to_cmd(u_long *argc,* **char** ***argv,* **char** ***envp***);**

token_t *au_to_data(char *unit_print,* **char** *unit_type,* **char** *unit_count,* **char** **p***);**

token_t *au_to_groups(int **groups***);**

token_t *au_to_in_addr(struct inaddr **internet_addr***);**

token_t *au_to_iport(u_short *iport***);**

token_t *au_to_ipc(int *id***);**

token_t *au_to_ipc_perm(struct ipc_perm **perm***);**

token_t *au_to_iport(u_short *iport***);**

token_t *au_to_me(void);

token_t *au_to_newgroups(int n, int *groups);

token_t *au_to_opaque(char **data,* **short** *bytes***);**

token_t *au_to_path(char **path***);**

token_t *au_to_process (au_id_t *auid,* **uid_t** *euid,* **gid_t** *egid,* **uid_t** *ruid,* **gid_t** *rgid,* **au_asid_t** *sid,* **au_tid_t** **tid***);**

token_t *au_to_return(char *number,* **u_int** *value***);**

token_t *au_to_socket(struct socket **so***);**

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_text(char **text***);**

MT_LEVEL

MT-Safe.

AVAILABILITY

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

DESCRIPTION

au_to_arg( ) formats the data in *v* into an ''argument token.'' The *n* argument indicates the argument number. The *text* argument is a null terminated string describing the argument.

au_to_attr( ) formats the data pointed to by *attr* into a ''vnode attribute token.''

au_to_data( ) formats the data pointed to by *p* into an ''arbitrary data token.'' The *unit_print* parameter determines the preferred display base of the data and is one of **AUP_BINARY**, **AUP_OCTAL**, **AUP_DECIMAL**, **AUP_HEX**, or **AUP_STRING**. The *unit_type* parameter defines the basic unit of data and is one of **AUR_BYTE**, **AUR_CHAR**, **AUR_SHORT**, **AUR_INT**, or **AUR_LONG**. The *unit_count* parameter specifies the number of basic data units to be used and must be positive.

au_to_groups( ) formats the array of 16 integers pointed to by *groups* into a ''groups token.''

au_to_in_addr( ) formats the data pointed to by *internet_addr* into an ''internet address token.''

au_to_ipc( ) formats the data in the *id* parameter into an ''interprocess communications id token.''

au_to_ipc_perm( ) formats the data pointed to by *perm* into an ''interprocess communications permission token.''

au_to_iport( ) formats the data pointed to by *iport* into an ''ip port address token.''

au_to_me( ) collects audit information from the current process and creates a ''subject token'' by calling **au_to_subject( )**.

au_to_newgroups( ) formats the array of *n* integers pointed to by *groups* into a ''newgroups token.''

au_to_subject( ) formats an *auid* (audit user ID), an *euid* (effective user ID), an *egid* (effective group ID), a *ruid* (real user ID), an *rgid* (real group ID), a *pid* (process ID), an *sid* (audit session ID), an *tid* (audit terminal ID), into a ''subject token.''

au_to_opaque( ) formats the *bytes* bytes pointed to by *data* into an ''opaque token.'' The value of *size* must be positive.

au_to_path( ) formats the path name pointed to by *path* into a ''path token.''

au_to_process( ) formats an *auid* (audit user ID), an *euid* (effective user ID), an *egid* (effective group ID), a *ruid* (real user ID), a *rgid* (real group ID), a *pid* (process ID), an *sid* (audit session ID), and a *tid* (audit terminal ID), into a ''process token.'' A process token should be used when the process is the object of an action (ie. when the process is the receiver of a signal).

au_to_return( ) formats an error number *number* and a return value *value* into a ''return value token.''

au_to_socket( ) format the data pointed to by *so* into a ''socket token.''

au_to_text( ) formats the **NULL** terminated string pointed to by *text* into a ''text token.''

RETURN VALUES

These functions return **NULL** if memory cannot be allocated to put the resultant token into, or if an error in the input is detected.

SEE ALSO **bsmconv**(1M), **au_open**(3)