

**NAME**

**jail\_getid, jail\_getname, jail\_setv, jail\_getv, jailparam\_all, jailparam\_init, jailparam\_import, jailparam\_import\_raw, jailparam\_set, jailparam\_get, jailparam\_export, jailparam\_free** - create and manage system jails

**LIBRARY**

Jail Library (libjail, -ljail)

**SYNOPSIS**

```
#include <sys/param.h>
```

```
#include <sys/jail.h>
```

```
#include <jail.h>
```

```
extern char jail_errmsg[];
```

```
int
```

```
jail_getid(const char *name);
```

```
char *
```

```
jail_getname(int jid);
```

```
int
```

```
jail_setv(int flags, ...);
```

```
int
```

```
jail_getv(int flags, ...);
```

```
int
```

```
jailparam_all(struct jailparam **jpp);
```

```
int
```

```
jailparam_init(struct jailparam *jp, const char *name);
```

```
int
```

```
jailparam_import(struct jailparam *jp, const char *value);
```

```
int
```

```
jailparam_import_raw(struct jailparam *jp, void *value, size_t valuelen);
```

```
int
```

```
jailparam_set(struct jailparam *jp, unsigned njp, int flags);
```

*int*

```
jailparam_get(struct jailparam *jp, unsigned njp, int flags);
```

*char \**

```
jailparam_export(struct jailparam *jp);
```

*void*

```
jailparam_free(struct jailparam *jp, unsigned njp);
```

## DESCRIPTION

The **jail** library is an interface to the `jail_set(2)` and `jail_get(2)` system calls, and the `security.jail.param` MIB entries. It simplifies the conversion of prison parameters between internal and string formats, allowing the setting and querying of prisons without knowing the parameter formats.

The **jail\_getid()** function returns the JID of the jail identified by *name*, or -1 if the jail does not exist.

The **jail\_getname()** function returns the name of the jail identified by *jid*, or NULL if the jail does not exist.

The **jail\_setv()** function takes a null-terminated list of name and value strings, and passes it to `jail_set(2)`.

The **jail\_getv()** function takes a null-terminated list of name and value strings, and passes it to `jail_get(2)`. It is the caller's responsibility to ensure that the value strings point to buffers large enough to hold the string representation of the returned parameters.

The **jailparam\_all()** function sets *jpp* to a list of all known jail parameters, and returns the number of parameters. The list should later be freed with **jailparam\_free()** and `free(3)`.

The **jailparam\_init()** function clears a parameter record and copies the *name* to it. After use, it should be freed with **jailparam\_free()**.

The **jailparam\_import()** function adds a *value* to a parameter record, converting it from a string to its native form. The **jailparam\_import\_raw()** function adds a value without performing any conversion.

The **jailparam\_set()** function passes a list of parameters to `jail_set(2)`. The parameters are assumed to have been created with **jailparam\_init()** and **jailparam\_import()**.

The **jailparam\_get()** function passes a list of parameters to `jail_get(2)`. The parameters are assumed to

have been created with **jailparam\_init()** or **jailparam\_list()**, with one parameter (the key) having been given a value with **jailparam\_import()**.

The **jailparam\_export()** function returns the string equivalent of a parameter value. The returned string should be freed after use.

The **jailparam\_free()** function frees the stored names and values in a parameter list. If the list itself came from **jailparam\_all()**, it should be freed as well.

## RETURN VALUES

The **jail\_getid()**, **jail\_setv()**, **jail\_getv()**, **jailparam\_set()** and **jailparam\_get()** functions return a JID on success, or -1 on error.

The **jail\_getname()** and **jailparam\_export()** functions return a dynamically allocated string on success, or NULL on error.

The **jailparam\_all()** function returns the number of parameters on success, or -1 on error.

The **jailparam\_init()**, **jailparam\_import()** and **jailparam\_import\_raw()** functions return 0 on success, or -1 on error.

Whenever an error is returned, *errno* is set, and the global string *jail\_errmsg* contains a description of the error, possibly from **jail\_set(2)** or **jail\_get(2)**.

## EXAMPLES

Set the hostname of jail "foo" to "foo.bar":

```
jail_setv(JAIL_UPDATE, "name", "foo", "host.hostname", "foo.bar",
        NULL);
```

OR:

```
struct jailparam params[2];
jailparam_init(&params[0], "name");
jailparam_import(&params[0], "foo");
jailparam_init(&params[1], "host.hostname");
jailparam_import(&params[1], "foo.bar");
jailparam_set(params, 2, JAIL_UPDATE);
jailparam_free(params, 2);
```

Retrieve the hostname of jail "foo":

```
char hostname[MAXHOSTNAMELEN];
jail_getv(0, "name", "foo", "host.hostname", hostname, NULL);
```

OR:

```
struct jailparam params[2];
jailparam_init(&params[0], "name");
jailparam_import(&params[0], "foo");
jailparam_init(&params[1], "host.hostname");
jailparam_get(params, 2, 0);
hostname = jailparam_export(&params[1]);
jailparam_free(params, 2);
...
free(hostname);
```

## ERRORS

The **jail** functions may return errors from `jail_set(2)`, `jail_get(2)`, `malloc(3)` or `sysctl(3)`. In addition, the following errors are possible:

[EINVAL]            A parameter value cannot be converted from the passed string to its internal form.

[ENOENT]            The named parameter does not exist.

[ENOENT]            A parameter is of an unknown type.

## SEE ALSO

`jail(2)`, `jail(3lua)`, `sysctl(3)`, `jail(8)`

## HISTORY

The **jail** library first appeared in FreeBSD 8.0.

## AUTHORS

James Gritton