NAME

CONF_modules_load_file, CONF_modules_load - OpenSSL configuration functions

SYNOPSIS

```
#include <openssl/conf.h>

int CONF_modules_load_file(const char *filename, const char *appname,
unsigned long flags);
int CONF_modules_load(const CONF *cnf, const char *appname,
unsigned long flags);
```

DESCRIPTION

The function *CONF_modules_load_file()* configures OpenSSL using file **filename** and application name **appname**. If **filename** is NULL the standard OpenSSL configuration file is used. If **appname** is NULL the standard OpenSSL application name **openssl_conf** is used. The behaviour can be cutomized using **flags**.

CONF_modules_load() is idential to *CONF_modules_load_file()* except it reads configuration information from **cnf**.

NOTES

The following **flags** are currently recognized:

CONF_MFLAGS_IGNORE_ERRORS if set errors returned by individual configuration modules are ignored. If not set the first module error is considered fatal and no further modules are loaded.

Normally any modules errors will add error information to the error queue. If **CONF_MFLAGS_SILENT** is set no error information is added.

If **CONF_MFLAGS_NO_DSO** is set configuration module loading from DSOs is disabled.

CONF_MFLAGS_IGNORE_MISSING_FILE if set will make *CONF_load_modules_file()* ignore missing configuration files. Normally a missing configuration file return an error.

CONF_MFLAGS_DEFAULT_SECTION if set and **appname** is not NULL will use the default section pointed to by **openssl_conf** if **appname** does not exist.

Applications should call these functions after loading builtin modules using *OPENSSL_load_builtin_modules()*, any ENGINEs for example using *ENGINE_load_builtin_engines()*, any algorithms for example *OPENSSL_add_all_algorithms()* and (if the application uses libssl) *SSL_library_init()*.

By using *CONF_modules_load_file()* with appropriate flags an application can customise application configuration to best suit its needs. In some cases the use of a configuration file is optional and its absence is not an error: in this case **CONF_MFLAGS_IGNORE_MISSING_FILE** would be set.

Errors during configuration may also be handled differently by different applications. For example in some cases an error may simply print out a warning message and the application continue. In other cases an application might consider a configuration file error as fatal and exit immediately.

Applications can use the *CONF_modules_load()* function if they wish to load a configuration file themselves and have finer control over how errors are treated.

EXAMPLES

Load a configuration file and print out any errors and exit (missing file considered fatal):

```
if (CONF_modules_load_file(NULL, NULL, 0) <= 0) {
fprintf(stderr, "FATAL: error loading configuration file\n");
ERR_print_errors_fp(stderr);
exit(1)
}
```

Load default configuration file using the section indicated by ‘‘myapp’’, tolerate missing files, but

exit on other errors:

```
if (CONF_modules_load_file(NULL, "myapp",
CONF_MFLAGS_IGNORE_MISSING_FILE) <= 0) {
fprintf(stderr, "FATAL: error loading configuration file\n");
ERR_print_errors_fp(stderr);
exit(1)
}
```

Load custom configuration file and section, only print warnings on error, missing configuration file ignored:

```
if (CONF_modules_load_file("/something/app.cnf", "myapp",
CONF_MFLAGS_IGNORE_MISSING_FILE) <= 0) {
fprintf(stderr, "WARNING: error loading configuration file\n");
ERR_print_errors_fp(stderr);
}
```

Load and parse configuration file manually, custom error handling:

```
FILE *fp;
CONF *cnf = NULL;
long eline;
fp = fopen("/somepath/app.cnf", "r");
if (fp == NULL) {
fprintf(stderr, "Error opening configuration file\n");
/* Other missing configuration file behaviour */
} else {
cnf = NCONF_new(NULL);
if (NCONF_load_fp(cnf, fp, &eline) == 0) {
fprintf(stderr, "Error on line %ld of configuration file\n", eline);
ERR_print_errors_fp(stderr);
/* Other malformed configuration file behaviour */
} else if (CONF_modules_load(cnf, "appname", 0) <= 0) {
fprintf(stderr, "Error configuring application\n");
ERR_print_errors_fp(stderr);
/* Other configuration error behaviour */
}
fclose(fp);
NCONF_free(cnf);
}
```

RETURN VALUES

These functions return 1 for success and a zero or negative value for failure. If module errors are not ignored the return code will reflect the return value of the failing module (this will always be zero or negative).

SEE ALSO

conf(5), *OPENSSL_config(3)*, *CONF_free(3)*, *err(3)*

HISTORY

CONF_modules_load_file and CONF_modules_load first appeared in OpenSSL 0.9.7.