OpenSSL - config

Fichers de configuration de la librairie CONF

La librairie CONF openssl peut être utilisée pour lire les fichiers de configuration. Elle est utilisée pour le fichier de configuration maître openssl.cnf et dans d'autres endroits comme dans les fichiers SPKAC et les fichiers d'extensions de certificat pour l'utilitaire x509. Les application openssl peuvent également utiliser la librairie CONF pour leur propre utilisation.

Un fichier de configuration est divisé en nombre de sections. Chaque section commence avec une ligne [section_name] et se termine quand une nouvelle section commence.

La première section d'un fichier de configuration est spécial et est référré à la section par défaut qui est généralement non nommée. Quand un nom est recherché il est d'abord recherché dans une section nommée, puis dans la section par défaut.

L'environnement est mappé en une section appelée ENV. Chaque section consiste d'un nombre de paire nom=valeur.

La valeur peut être étendue en utilisant la forme $var ou ${var}, qui substitue la valeur de la variable dans la section courante. Il est également possible de substituer une valeur d'une autre section en utilisant la syntaxe $section : :name ou ${section : :name}. En utilisant la forme $ENV : :name les variables d'environnement peuvent être substituées. Il est également possible d'assigner des valeurs aux variables d'environnement en utilisant le nom ENV : :name, cela fonctionne si le programme recherche des variables d'environnement en utilisant la librairie CONF au lieu d'appeler getenv() directement.

Configuration de la librairie openssl

Les application peuvent configurer automatiquement certains aspects d'openssl en utilisant le fichier de configuration maître, ou optionnellement un fichier de configuration alternatif. L'utilitaire openssl inclus cette fonctionnalité ; toute sous-commande utilise le fichier de configuration maître sauf si une option est utilisée dans la sous commande pour utiliser un fichier de configuration alternatif.

Pour activer la configuration de librairie la section par défaut doit contenir la ligne appropriée qui pointe vers la section de configuration principale. Le nom par défaut est openssl_conf qui est utilisé par l'utilitaire openssl. D'autres applications peuvent utiliser un nom alternatif

La section de configuration devrait consister en un jeu de nom=valeur qui contient des informations spécifique au module. Le nom représente le nom du module de configuration et la valeur est spécifique au module. Par exemple :

```
openssl_conf = openssl_init
[openssl_init]
oid_section = new_oids
engines = engine_section

[new_oids]
... new oids here ...

[engine_section]
... engine stuff here ...
```

Module de configuration d'objet ASN1

Ce module a le nom oid_section. La valeur est le nom de la section contenant les paires de valeur d'OID. Bien que certains utilitaires openssl ont leur propre section d'objet ASN1, toutes ne l'ont pas. En utilisant le module de configuration d'objet ASN1 tout l'utilitaire openssl peut voir les nouveaux objets. Par exemple :

```
[new_oids]
some_new_oid = 1.2.3.4
some_other_oid = 1.2.3.5
```

Il est également possible de définir la valeur avec un nom long, suivi par une virgule et l'OID :

```
shortName = some object long name, 1.2.3.4
```

Module de configuration engine

Le module de configuration ENGINE a le nom des engines. La valeur pointe vers une section contenant les informations de configuration du moteur.

Chaque section spécifique est utilisée pour définir les algorithmes par défaut, le chargement dynamique, d'initialisation et les contrôles envoyés. par exemple :

```
[engine_section]
# Configure ENGINE named "foo"
foo = foo_section
# Configure ENGINE named "bar"
bar = bar_section

[foo_section]
... foo ENGINE specific commands ...
[bar_section]
... "bar" ENGINE specific commands ...
```

La commande engine_id est utilisée pour donner le nom du moteur. par exemple :

```
[engine_section]
# This would normally handle an ENGINE named "foo"
foo = foo_section

[foo_section]
# Override default name and use "myfoo" instead.
engine_id = myfoo
```

La commande dynamic_path charge et ajoute un engine. C'est équivalent au contrôle SO_PATH avec la partie argument suivie par LIST_ADD avec la valeur 2 et LOAD à l'engine dynamic.

La commande init détermine l'initialisation de l'engine. À 0 l'engine n'est pas initialisé. La commande default_algorithms définis les algorithmes par défaut qu'un engin fournis en utilisant les fonctions ENGINE_set_default_string().

Si le nom match aucune des commande ci-dessus, il sont considérés comme des commandes ctrl qui sont envoyés à l'engine. La valeur de la commande est l'argument de la commande ctrl. Si la v+aleur est une chaîne vide aucune valeur n'est envoyée à la commande. par exemple :

```
[engine_section]
# Configure ENGINE named "foo"
foo = foo_section

[foo_section]
```

```
# Load engine from DSO
dynamic_path = /some/path/fooengine.so
# A foo specific ctrl.
some_ctrl = some_value
# Another ctrl that doesn't take a value.
other_ctrl = EMPTY
# Supply all default algorithms
default_algorithms = ALL
```

Module de configuration EVP

Ce module a le nom alg_section qui pointe vers une section contenant les commandes algorithme. Actuellement la seule commande algorithme supportée est fips_mode dont la valeur devrait être un booléen. par exemple :

```
alg_section = evp_settings

[evp_settings]
fips_mode = on
```

Module de configuration ssl

Ce module a le nom ssl_conf qui pointe vers une section contenant les configurations ssl. Chaque ligne dans la section contient le nom de la configuration et la section. Chaque section consiste de paires commande-valeur pour SSL_CONF. Chaque paire sera passée à SSL_CTX ou la structure SSL en cas d'appel SSL_CTX_config() ou SSL_config() avec le nom de configuration approprié. Noter que tout caractère avant le point initial dans la section est ignoré donc la même commande peut être utilisée plusieurs fois :

```
ssl_conf = ssl_sect

[ssl_sect]
server = server_section

[server_section]
RSA.Certificate = server-rsa.pem
ECDSA.Certificate = server-ecdsa.pem
Ciphers = ALL:!RC4
```

Notes

Si un fichier de configuration tente d'étendre une variable qui n'existe pas alors une erreur est flaggée et le fichier n'est pas chargé. Cela peut se produire si une variable d'environnement n'existe pas.

Cela peut fonctionner en incluant une section par défaut pour fournir une valeur par défaut : si la recherche d'environnement échoue, la valeur par défaut sera utilisée. Pour que cela fonctionne correctement la valeur par défaut doit être définie très tôt dans le fichier de configuration.

Si la même variable existe dans la même section, seule la dernière est prise en compte. Sous certaines circonstances comme avec les DN le même champ peut être présent plusieurs fois :

```
1.OU = "My first OU"
2.OU = "My Second OU"
```

Exemples

Exemple de fichier de configuratio utilisant des fonctionnalités décrite plus haut :

```
HOME=/temp
RANDFILE= ${ENV::HOME}/.rnd
configdir=$ENV::HOME/config

[ section_one ]
# We are now in section one.
# Quotes permit leading and trailing whitespace
any = " any variable name "

other = A string that can \
cover several lines \
by including \\ characters

message = Hello World\n

[ section_two ]
greeting = $section_one::message
```

L'exemple suivant montre come étendre les variables d'environnement. Supposont que l'on veut une variable tmpfile référrant à un fichier temporaire. Le répertoire où il est placé peut être déterminé par la variable d'environnement TEMP ou TMP, mais elle peut ne pas être définie. En incluant les noms des variables d'environnement sans que celle-ci n'existe créé une erreur en chargeant le fichier de configuration. En utilisant la section par défaut pour les 2 valeurs :

```
# si TMP n'existe pas
TMP=/tmp
# Si TMP existe
TEMP=$ENV::TMP
tmpfile=${ENV::TEMP}/tmp.filename
```

Exemple pour entrer en mode fips :

```
openssl_conf = openssl_conf_section

[openssl_conf_section]
alg_section = evp_sect

[evp_sect]
fips_mode = yes
```

Configuration plus complexe :

```
openssl_conf = openssl_conf_section

[openssl_conf_section]
alg_section = evp_sect
oid_section = new_oids

[evp_sect]
fips_mode = no

[new_oids]
# New OID, just short name
newoid1 = 1.2.3.4.1
# New OID shortname and long name
newoid2 = New OID 2 long name, 1.2.3.4.2
```

Les exemples ci-dessus peuvent être utilisées avec toute application utilisant la librairie si openssl_conf est modifié pour correspondre au nom de l'application.

Par exemple, si le second fichie d'exemple ci-dessus est sauvé dans example.cnf, la ligne de commande :
OPENSSL_CONF=example.cnf openssl asn1parse -genstr OID :1.2.3.4.1
va afficher
0 :d=0 hl=2 l= 4 prim : OBJECT :newoid1

BUGS

Il n'y a actuellement aucun moyen d'utiliser des caractères en utilisant la notation octal \nnn. Les chaînes sont toutes terminer par un caractère null, donc les nulls ne peuvent pas faire partie de la valeur. Les fichiers sont chargés en une seule passe. Cela signifie qu'une expansion de variable ne fonctionne que si les variables sont référencées très tôt dans le fichier.