# opensc.conf

Fichier de configuration pour OpenSC

```
app default {
 debug = 0 ; # niveau de debug
 debug_file = "/var/log/opensc-debug.log" ; # Fichier où ecrire les informations de debug (défaut : stdout)
 error_file = "/var/log/opensc-errors.log" ; # Fichier où ecrire les information d'erreur (défaut : stderr)
 profile_dir = PKGDATADIR ; # Répertoire de profil pour pkcs15-init (défaut : /usr/share/opensc)
 reader_drivers = pcsc, openct, ctapi ; # pilotes à charger au démarrage. 'internal' va charger tous les
pilote liés statiquement (défaut)
}

 {
  reader_driver ctapi { # configuration pour le pilote ctapi
   module /usr/local/towitoko/lib/libtowitoko.so {
    ports = 0 ; # port CT-API 0..3 (COM1..4) 4 (printer) 5 (modem) 6..7 (LPT1..2)
   }
  }

  reader_driver pcsc { # configuration pour le pilote pcsc
   max_send_size = 252 ; # Taille max des données envoyée et reçues. Vérifier avec lsusb -vv pour dwMaxIFSD
   max_recv_size = 252 ;
   connect_exclusive = false ; # connecter le lecteur en mode exclusif (défaut : false)
   connect_reset = true ; # réinitialiser la carte après déconnection (défaut : true)
   transaction_reset = false ; # réinitialise la carte après charque transaction (défaut : false)
   enable_pinpad = false ; # active pinpad si détecté (défaut : false)
  }

  reader_driver openct {
   readers = 5 ; Lecteurs virtuels à allouer (défaut : 5)
  }

  card_drivers = customcos, internal ; # pilotes à charger au démarrage. 'internal' charge tous les pilotes
liés dynamiquement (défaut et doit toujours être la dernière entrée)
  card_driver customcos {
   module = /usr/lib/opensc/drivers/card_customcos.so ; # emplacement du pilote
  }

  force_card_driver = autodetect ; # si spécifié, force OpenSC à utiliser ce pilote.(défaut : autodetect)
  # noms des pilotes internes supportés : etoken, flex, cyberflex, gpk, miocos, mcrd, setcos, starcos, tcos,
openpgp, jcop, oberthur, belpic, emv, piv
  card_atr 3b:f0:0d:ca:fe { # nouvelle entrée pour le pilote flex. Format générique : card_atr <hex encoded
ATR (case-sensitive !)>
   atrmask = "ff:ff:ff:ff:ff" ; # masque AND pour l'ATR de la carte pour la l'identification de la référence
ATR.
   driver = "flex" ; # pilote à utiliser
   name = "My CryptoFlex card" ; { nom de la carte
    type = "2002" ; # type de carte (valeur entière)
    flags = "keygen", "rng", "0x80000000" ; # flags au format hexa (ou chaine pour certains : keygen =
capacités de génération de clé embarqué, rng = source de nombre aléatoire embarqué) pour personnaliser les
capacités de la carte.
    pkcs15emu = "custom" ; # Couche d'amulation PKCS #15, force le pilote d'émulation pour les cartes
spécifiques
```

```
    force_protocol = "t0" ; # Force le protocole à utiliser (t0, t1 ou raw)
   }
   card_atr 3B:7D:96:00:00:80:31:80:65:B0:83:11:00:AC:83:00:90:00 { # les cartes PIV on besoin d'un entrée
comme ceci
    name = "PIV-II" ;
    driver = "piv" ;
   }
   card_atr 3b:6e:00:ff:45:73:74:45:49:44:20:76:65:72:20:31:2e:30 { # les cartes Estonian ID et le pilote
Micardo fonctionne ensemble avec T=0. Seul l'ATR 'cold' devrait être spécifié
    force_protocol = t0 ;
   }
   card_atr 3b:fe:94:00:ff:80:b1:fa:45:1f:03:45:73:74:45:49:44:20:76:65:72:20:31:2e:30:43 {
    force_protocol = t0 ;
   }
   card_atr 3b:fe:94:00:ff:80:b1:fa:45:1f:03:45:73:74:45:49:44:20:76:65:72:20:31:2e:30:43 { # Les cartes
D-Trust sont aussi basées sur micardo et T=0
    force_protocol = t0 ;
   }
   card_atr 3b:ff:94:00:ff:80:b1:fe:45:1f:03:00:68:d2:76:00:00:28:ff:05:1e:31:80:00:90:00:23 {
    force_protocol = t0 ;
   }
   card_atr 3b:ff:11:00:ff:80:b1:fe:45:1f:03:00:68:d2:76:00:00:28:ff:05:1e:31:80:00:90:00:a6 {
    force_protocol = t0 ;
   }
   framework pkcs15 { # block spécifique au framework PKCS #15
    use_caching = false ; # spécifie d'utiliser les fichier en cache dans le home de l'utilisateur en
apprenant la carte (pkcs15-tool -L) (ne doit pas être utilisé en root) (défaut : false)
    enable_pkcs15_emulation = yes ; # Active l'émulation pkcs15 (défaut : yes)
    try_emulation_first = yes ; # tente d'abord l'émulation pkcs15 avant le traitement normal pkcs15 (défaut
: no)
    enable_builtin_emulation = yes ; # Active l'émulation intégrée
    builtin_emulators = esteid, openpgp, tcos, starcert, infocamere, postecert, actalis, atrust-acos,
gemsafe, tccardos, PIV-II ; # liste des émulateurs pkcs15 intégrés à tester
    emulate custom { # Paramètre pour un émulateur pkcs15 chargé depuis une librairie externe
     module = /usr/lib/opensc/drivers/p15emu_custom.so ; # emplacement de la librairie
    }
   }
  }

  app opensc-pkcs11 { # paramètres pour le module PKCS11 OpenSC
  pkcs11 {
   num_slots = 4 ; # Nombre de slot maximum par carte à puce
   hide_empty_tokens = yes ; # Cache les slots vides
   lock_login = true ; # opensc tente de locker la carte une fois authentifiée via C_Login. (défaut : false)
   cache_pins = true # normalement le module pkcs11 ne met pas en cache le pin présenté via C_Login, mais
certaines carte ne fonctionne pas correctement avec opensc (défaut : true)
   soft_keygen_allowed = true ; # Force la génération de pair de clé embarqué (défaut : true)
   }
  }
```