Mise en œuvre d'un lecteur NFC ACR122U sous Linux/Ubuntu

tvaira@free.fr © 2014-2019

http://www.acs.com.hk/en/products/3/acr122u-usb-nfc-reader/

NFC (Near Field Communication, communication en champ proche) est une technologie de communication sans-fil à courte portée et haute fréquence, permettant l'échange d'informations entre des périphériques jusqu'à une distance d'environ 10 cm. Cette technologie est une extension de la norme **ISO/CEI 14443** standardisant les cartes de proximité utilisant la radio-identification (**RFID**).

Documents annexes : Annexe 1 (carte Mifare Classic) et Annexe 2 (message ATR)

1. Détection

```
$ dmesg
...
[  728.576207] usb 3-10.4.2: new full-speed USB device number 11 using xhci_hcd
[  728.595917] usb 3-10.4.2: New USB device found, idVendor=072f, idProduct=2200
[  728.595919] usb 3-10.4.2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[  728.595921] usb 3-10.4.2: Product: ACR122U PICC Interface
[  728.595922] usb 3-10.4.2: Manufacturer: ACS
...

$ lsusb
...
Bus 003 Device 011: ID 072f:2200 Advanced Card Systems, Ltd
```

Conclusion : le lecteur USB est bien détecté.

2. Installation logicielle

2.1. Préparation

```
$ sudo apt-get install pcsc-tools pcscd
$ sudo apt-get install libpcsclite-dev
$ sudo apt-get install libpcsclite1
$ sudo apt-get install libusb-dev
```

2.2. Installation de `libnfc`

- **Paquets (Ubuntu)**

```
$ sudo apt-get install libnfc-bin libnfc-dev libnfc5
```

- **ou :**

```
$ wget http://dl.bintray.com/nfc-tools/sources/libnfc-1.7.1.tar.bz2
$ tar xjf libnfc-1.7.1.tar.bz2

$ cd libnfc-1.7.1/
$ ./configure // ou : $ ./configure --with-drivers=acr122_usb
$ make

$ sudo make install
$ sudo ldconfig
```

Remarque : par défaut, l'environnement de développement a été installé dans ***/usr/local****.*

3. Tests

Remarque : si vous avez l'erreur « Unable to claim USB interface (Device or resource busy) », cela est certainement dû à la version du noyau Linux > **3.5** (cf. **`uname -a`**). Dans ce cas, il vous faudra faire :

```
$ sudo vim /etc/modprobe.d/blacklist-libnfc.conf
blacklist nfc
blacklist pn533
blacklist pn533_usb
```

Manuellement :

```
$ sudo modprobe -r pn533_usb
$ sudo modprobe -r pn533
```

Et éventuellement redémarrer **pcscd** :

```
$ sudo service pcscd restart
```

3.1. Test du lecteur

Remarque : pour le sudo cf. ***Droits d'accès*** *(plus loin)*

```
$ sudo nfc-scan-device -v
1 NFC device(s) found:
- ACS / ACR122U PICC Interface:
    acr122_usb:003:011
chip: PN532 v1.6
initator mode modulations: ISO/IEC 14443A (106 kbps), FeliCa (424 kbps, 212 kbps),
ISO/IEC 14443-4B (106 kbps), Innovision Jewel (106 kbps), D.E.P. (424 kbps, 212 kbps,
106 kbps)
target mode modulations: ISO/IEC 14443A (106 kbps), FeliCa (424 kbps, 212 kbps),
D.E.P. (424 kbps, 212 kbps, 106 kbps)
```

Conclusion : il semble que le lecteur soit un ***ACS / ACR122U****.*

3.2. Test d'une carte Mifare

```
$ sudo nfc-list
NFC device: ACS / ACR122U PICC Interface opened
1 ISO14443A passive target(s) found:
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
       UID (NFCID1): 17  f0  79  21
      SAK (SEL_RES): 08
```

Avec le badge de l'école :)

```
$ sudo nfc-list
nfc-list uses libnfc 1.7.1
NFC device: ACS / ACR122U PICC Interface opened
1 ISO14443A passive target(s) found:
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
       UID (NFCID1): 0b  dd  47  cd
      SAK (SEL_RES): 08
```

3.3. Autres tests

Avec les 2 outils installés à partir du paquetage ***`pcsc-tools`*** *:*

```
$ sudo pcsc_scan
PC/SC device scanner
V 1.4.18 (c) 2001-2011, Ludovic Rousseau <ludovic.rousseau@free.fr>
Compiled with PC/SC lite version: 1.7.4
Using reader plug'n play mechanism
Scanning present readers...
0: ACS ACR122U PICC Interface 00 00

Sun Jul 27 09:36:40 2014
Reader 0: ACS ACR122U PICC Interface 00 00
  Card state: Card removed,

Sun Jul 27 09:36:52 2014
Reader 0: ACS ACR122U PICC Interface 00 00
  Card state: Card inserted,
  ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A

ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
+ TS = 3B --> Direct Convention
+ T0 = 8F, Y(1): 1000, K: 15 (historical bytes)
  TD(1) = 80 --> Y(i+1) = 1000, Protocol T = 0
-----
  TD(2) = 01 --> Y(i+1) = 0000, Protocol T = 1
-----
+ Historical bytes: 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00
  Category indicator byte: 80 (compact TLV data object)
    Tag: 4, len: F (initial access data)
      Initial access data: 0C A0 00 00 03 06 03 00 01 00 00 00 00
+ TCK = 6A (correct checksum)

Possibly identified card (using /usr/share/pcsc/smartcard_list.txt):
3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
3B 8F 80 01 80 4F 0C A0 00 00 03 06 .. 00 01 00 00 00 00 ..
    Mifare Standard 1K (as per PCSC std part3)
3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 .. .. 00 00 00 00 ..
    RFID - ISO 14443 Type A Part 3 (as per PCSC std part3)
3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
    Philips MIFARE Standard (1 Kbytes EEPROM)
    http://www.nxp.com/#/pip/pip=[pfp=41863]|pp=[t=pfp,i=41863]
    RFID - ISO 14443 Type A - Transport for London Oyster
    ACOS5/1k Mirfare
    RFID - ISO 14443 Type A - NXP Mifare card with 1k EEPROM
    vivotech ViVOcard Contactless Test Card

$ ATR_analysis 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
+ TS = 3B --> Direct Convention
+ T0 = 8F, Y(1): 1000, K: 15 (historical bytes)
  TD(1) = 80 --> Y(i+1) = 1000, Protocol T = 0
-----
  TD(2) = 01 --> Y(i+1) = 0000, Protocol T = 1
```

```
-----
+ Historical bytes: 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00
  Category indicator byte: 80 (compact TLV data object)
    Tag: 4, len: F (initial access data)
      Initial access data: 0C A0 00 00 03 06 03 00 01 00 00 00 00
+ TCK = 6A (correct checksum)

Possibly identified card (using /usr/share/pcsc/smartcard_list.txt):
3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
3B 8F 80 01 80 4F 0C A0 00 00 03 06 .. 00 01 00 00 00 00 ..
    Mifare Standard 1K (as per PCSC std part3)
3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 .. .. 00 00 00 00 ..
    RFID - ISO 14443 Type A Part 3 (as per PCSC std part3)
3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
    Philips MIFARE Standard (1 Kbytes EEPROM)
    http://www.nxp.com/#/pip/pip=[pfp=41863]|pp=[t=pfp,i=41863]
    RFID - ISO 14443 Type A - Transport for London Oyster
    ACOS5/1k Mirfare
    RFID - ISO 14443 Type A - NXP Mifare card with 1k EEPROM
    vivotech ViVOcard Contactless Test Card
```

*Conclusion : Il semble que la carte lue soit une carte **Philips MIFARE Standard (1 Kbytes EEPROM)**. Lire l'annexe 1 (carte Mifare Classic) et l'annexe 2 (message ATR).*

4. Droits d'accès

Manuellement :

```
$ sudo ./utils/nfc-scan-device -v
1 NFC device(s) found:
- ACS / ACR122U PICC Interface:
    acr122_usb:003:011

$ sudo chmod o+rw /dev/bus/usb/003/011
```

Automatiquement :

```
$ sudo cp contrib/udev/42-pn53x.rules /etc/udev/rules.d/99-pn53x.rules

$ sudo vim /etc/udev/rules.d/99-pn53x.rules
...
ATTRS{idVendor}=="072f", ATTRS{idProduct}=="2200", MODE="0666", SYMLINK+="nfc"
...
```

Remarque : un lien symbolique `/dev/nfc` sera aussi créé automatiquement au branchement du lecteur.

Test :

```
$ nfc-scan-device -v
1 NFC device(s) found:
- ACS / ACR122U PICC Interface:
    acr122_usb:003:011
...
```

5. Programmation

Lien : http://tvaira.free.fr/projets/archives/test-nfc-acr122u.zip

Annexe 1 : la carte MIFARE Classic

MIFARE est une des technologies de carte à puce sans contact les plus répandues dans le monde avec 3,5 milliards de cartes et 40 millions de modules de lecture/encodage. La marque, lancée par **Philips**, est propriété de la société NXP.

MIFARE est fondée (partiellement ou complètement selon les modèles) sur l'un des standards ISO décrivant les cartes à puce sans contact : l'**ISO 14443 de Type A** fonctionnant à **13,56 MHz**. La technologie est intégrée à la fois dans les cartes et dans les lecteurs/encodeurs. Ces lecteurs sont fabriqués par différents fabricants sur la base d'un composant fourni par la société NXP.

En fait, le nom MIFARE englobe plusieurs types de cartes à puce sans contact très différents.

Les cartes **MIFARE dites Classic** (ou **Standard**) sont des cartes à mémoire (logique câblée). Elles ne respectent que partiellement le standard ISO 14443A, puisqu'elles utilisent un jeu de commandes propriétaire à la place du protocole de haut-niveau ISO 14443-4, et ne respectent pas le format de trames ISO 14443-3 dans les communications chiffrées. Elles utilisent un protocole de sécurité propriétaire NXP (CRYPTO1) pour l'authentification et le chiffrement qui a été cassé en 2008.

La carte MIFARE Classic dispose d'un **numéro de série ou CSN de 32 bits pré-encodé** (**UID**) et d'un espace de stockage découpé en segments de données puis en blocs de données avec des mécanismes de sécurité simples.

La carte **MIFARE Classic 1k** offre **768 octets de stockage répartis sur 16 secteurs**. Chaque secteur est composé de **4 blocs de 16 octets** dont **3 blocs (3 x 16 x 16 = 768)** sont accessibles. Un bloc de sécurité supplémentaire vient protéger l'accès au secteur par **deux clefs différentes (nommées A et B)**.

Secteur 0																
Bloc 0	UID				Manufacturer Data											
Bloc 1																
Bloc 2																
Bloc 3	KEY A						ACCESS				KEY B					
Secteur 1																
Bloc 4																
Bloc 5																
Bloc 6																
Bloc 7	KEY A						ACCESS				KEY B					
...																

Applications : Avec leur prix relativement bas, elles sont essentiellement utilisées dans **le transport ou la billetterie**. Dans l'entreprise elles conviennent à de nombreuses applications dont le porte-monnaie électronique, la gestion des accès (physiques et logiques), la gestion horaire et bien d'autres.

Les cartes MIFARE dites Classic sont protégées par une algorithme de chiffrement propriétaire : il s’agît de **CRYPTO1**. La connaissance des clés de chiffrement est nécessaire au décryptage des données présentes sur la carte MIFARE. En d’autres termes, on ne peut lire ou écrire sur la carte MIFARE dite Classic sans connaître les différentes clés de chiffrement.

Remarques : Les informations sensibles, stockées sur la puce MIFARE Classic, sont protégées par un numéro qui agit comme une clé. La sécurité de tout le système est basée sur le fait que cette « clé » ne peut être découverte. En mars, le Professeur Jacobs et son équipe ont découvert qu'il était relativement facile de calculer ces clés à partir de données récupérables, et à partir de là, encore plus facile de créer ou de copier des cartes. (cf. l'utilitaire **`mfoc`** [**https://code.google.com/p/mfoc/**] pour récupérer les clés et l'utilitaire **`nfc-mfclassic`** de `libnfc` pour cloner une carte)

Principe du dialogue entre une carte et un lecteur (ISO 14443-3A) :

- dès que la carte est mise sous tension, elle envoie un message de réponse d'initialisation appelé **ATR** (*Answer To Reset*) et attend une commande du lecteur. Le message ATR indique à l'application cliente les paramètres nécessaires pour établir une communication avec elle.
- Le lecteur envoie un message REQA (*Request Command for Type A*)
- les cartes situées dans la zone d'influence répond ent par **ATQA** (*Answer To Request Code*)
- processus anti-collision : lecture de l'identifian t de la carte (**UID** de la carte lors de sa fabrication) et attend un **SAK** (*Select AcKnowledge*)
- élection d'une carte
- le lecteur informe les autres cartes non sélectionnées qui retournent dans un état semi-passif
- la carte sélectionnée envoie une réponse **ATS** (*Answer To Select*) qui contient les caractéristiques de la carte

Exemple de lecture des 4 premiers blocs d'une carte Mifare Classic 1k :

```
# Message : cmd bloc cle uid avec cmd=0x60 (authentification clé A)
Send : 60 00 ff ff ff ff ff ff 17 f0 79 21

# Message : cmd bloc avec cmd=0x30 (lecture)
Send : 30 00

Receive : 17 f0 79 21 bf 08 04 00 62 63 64 65
66 67 68 69

# Message : cmd bloc avec cmd=0x30 (lecture)
Send : 30 01
Receive : 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00

# Message : cmd bloc avec cmd=0x30 (lecture)
Send : 30 02
Receive : 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00

# Message : cmd bloc avec cmd=0x30 (lecture)
Send : 30 03
Receive : 00 00 00 00 00 00 ff 07 80 69 ff ff
ff ff ff ff
```

Table 9. Command overview

Command	ISO/IEC 14443	Command code (hexadecimal)
Request	REQA	26h (7 bit)
Wake-up	WUPA	52h (7 bit)
Anticollision CL1	Anticollision CL1	93h 20h
Select CL1	Select CL1	93h 70h
Anticollision CL2	Anticollision CL2	95h 20h
Select CL2	Select CL2	95h 70h
Halt	Halt	50h 00h
Authentication with Key A	-	60h
Authentication with Key B	-	61h
Personalize UID Usage	-	40h
MIFARE Read	-	30h
MIFARE Write	-	A0h
MIFARE Decrement	-	C0h
MIFARE Increment	-	C1h
MIFARE Restore	-	C2h
MIFARE Transfer	-	B0h

Remarque : une seule authentification (avec la clé associé à ce secteur) est nécessaire pour lire 4 blocs consécutifs d'un même secteur.

Exemple d'écriture de données dans le bloc n°60 (0x3c) sur une carte Mifare Classic 1k :

```
# Message : cmd bloc cle uid avec cmd=0x60 (authentification clé A)
Send : 60 3c ff ff ff ff ff ff 17 f0 79 21

# Message : cmd bloc data cmd=0xA0 (écriture)
Send : a0 3c 42 6f 6e 6a 6f 75 72 20 6c 65 20 6d 6f 6e 64 65
```

Les droits d'accès :

Les conditions d'accès des 4 blocs d'un secteur sont définies par 3 bits qui sont stockés dans 3+1 octets de la zone ACCESS :

Secteur x																
Bloc n																
Bloc n+1																
Bloc n+2																
Bloc n+3	KEY A						ACCESS				KEY B					
...																

Ces bits contrôlent les droits d'accès (lecture/écriture) de la carte à l'aide des clés secrètes A et B. Les les conditions d'accès peuvent être modifiés, à condition d'avoir les droits pour cette opération.

Pour obtenir les droits à appliquer à un secteur, il faut décoder les 3 octets du champ « Access bits » en utilisant la table 6 de la documentation de la carte Mifare Classic :

Table 6. Access conditions

Access Bits	Valid Commands		Block	Description
$C1_3$, $C2_3$, $C3_3$	read, write	→	3	sector trailer
$C1_2$, $C2_2$, $C3_2$	read, write, increment, decrement, transfer, restore	→	2	data block
$C1_1$, $C2_1$, $C3_1$	read, write, increment, decrement, transfer, restore	→	1	data block
$C1_0$,$C2_0$, $C3_0$	read, write, increment, decrement, transfer, restore	→	0	data block

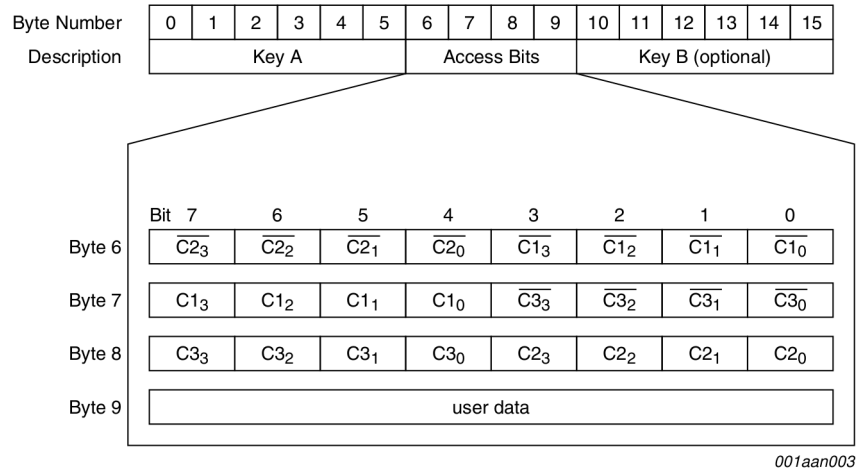

Fig 10. Access conditions

Exemple n°1 : décodage ff 07 80 69

Byte 6	0xFF	1	1	1	1	1	1	1	1
Byte 7	0x07	0	0	0	0	0	1	1	1
Byte 8	0x80	1	0	0	0	0	0	0	0
Byte 9	0x69	*user data*							

On obtient les 3 bits pour chaque bloc :

	C1	C2	C3	
Bloc n	0	0	0	lecture (cléA|B), écriture (cléA|B)
Bloc n+1	0	0	0	lecture (cléA|B), écriture (cléA|B)
Bloc n+2	0	0	0	lecture (cléA|B), écriture (cléA|B)
Bloc n+3	0	0	1	clé A lecture (jamais) et écriture (clé A) bits d'accès lecture (clé A) et écriture (clé A) clé B lecture (clé A) et écriture (clé A)

Pour obtenir les droits à appliquer, on utilise les tables 7 et 8 de la documentation :

Table 7. Access conditions for the sector trailer

Access bits			Access condition for						Remark
			KEYA		Access bits		KEYB		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read[1]
0	1	0	never	never	key A	never	key A	never	Key B may be read[1]
1	0	0	never	key B	key A|B	never	never	key B	
1	1	0	never	never	key A|B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration[1]
0	1	1	never	key B	key A|B	key B	never	key B	
1	0	1	never	never	key A|B	key B	never	never	
1	1	1	never	never	key A|B	never	never	never	

[1] For this access condition key B is readable and may be used for data

Table 8. Access conditions for data blocks

Access bits			Access condition for				Application
C1	C2	C3	read	write	increment	decrement, transfer, restore	
0	0	0	key A|B	key A|B	key A|B	key A|B	transport configuration[1]
0	1	0	key A|B	never	never	never	read/write block[1]
1	0	0	key A|B	key B	never	never	read/write block[1]
1	1	0	key A|B	key B	key B	key A|B	value block[1]
0	0	1	key A|B	never	never	key A|B	value block[1]
0	1	1	key B	key B	never	never	read/write block[1]
1	0	1	key B	never	never	never	read/write block[1]
1	1	1	never	never	never	never	read/write block

Remarque : il est conseillé de toujours laisser la possibilité d'écrire dans la champ « Access bits » (soit 001, 011, ou 101, d'après la table 7)

Exemple n°2 :

On désire appliquer les droits suivants :

- interdire la lecture des clés
- écriture des clés, des bits d'accès des données avec la clé B
- lecture des données et des bits d'accès avec la A ou B
- pas d'incrémentation/décrémentation

On choisit les 3 bits pour chaque bloc :

	C1	C2	C3	
Bloc n	1	0	0	lecture (cléA|B), écriture (B)
Bloc n+1	1	0	0	lecture (cléA|B), écriture (B)
Bloc n+2	1	0	0	lecture (cléA|B), écriture (B)
Bloc n+3	0	1	1	clé A lecture (jamais) et écriture (clé B) bits d'accès lecture (clé A|B) et écriture (clé A|B) clé B lecture (jamais) et écriture (clé B)

Puis les 3 octets du champ « Access bits » :

Byte 6	0x78	0	1	1	1	1	0	0	0
Byte 7	0x77	0	1	1	1	0	1	1	1
Byte 8	0x88	1	0	0	0	1	0	0	0
Byte 9	0x69	*user data*							

Annexe 2 : message ATR (*Answer To Reset*)

ATR format for ISO 14443 Part 3 PICCs

Byte	Value (Hex)	Designation	Description
0	3Bh	Initial Header	-
1	8Nh	T0	Higher nibble 8 means: no TA1, TB1, TC1 only TD1 is following. Lower nibble N is the number of historical bytes (HistByte 0 to HistByte N-1)
2	80h	TD1	Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following. Lower nibble 0 means T = 0
3	01h	TD2	Higher nibble 0 means no TA3, TB3, TC3, TD3 following. Lower nibble 1 means T = 1
4 To 3+N	80h	T1	Category indicator byte, 80 means A status indicator may be present in an optional COMPACT-TLV data object
	4Fh	Tk	Application identifier Presence Indicator
	0Ch		Length
	RID		Registered Application Provider Identifier (RID) # A0 00 00 03 06h
	SS		Byte for standard
	C0 .. C1h		Bytes for card name
	00 00 00 00h	RFU	RFU # 00 00 00 00h
4+N	UUh	TCK	Exclusive-oring of all the bytes T0 to Tk

Table 2: ATR format for ISO 14443 Part 3 PICCs

Example:

ATR for Mifare 1K = {3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6Ah}

ATR											
Initial Header	T0	TD1	TD2	T1	Tk	Length	RID	Standard	Card Name	RFU	TCK
3Bh	8Fh	80h	01h	80h	4Fh	0Ch	A0 00 00 03 06h	03h	00 01h	00 00 00 00h	6Ah

RID = A0 00 00 03 06h (PC/SC Workgroup)

Standard (SS) = 03h (ISO 14443A, Part 3)

Card Name (C0 .. C1) = [00 01h] (Mifare 1K)

Where, Card Name (C0 .. C1)

00 01h: Mifare 1K

00 02h: Mifare 4K

00 03h: Mifare Ultralight

00 26h: Mifare Mini

F0 04h: Topaz and Jewel

F0 11h: FeliCa 212K

F0 12h: Felica 424K

FFh [SAK]: Undefined