Correction
Travaux pratiques
Protocoles ISO 7816-3, ISO 7816-4
Standards

TP n°1 – Interaction avec la carte SIM - Les premières commandes APDUs en utilisant un outil de script PC/SC

TP n°2 – Programmation d'une application Java pour explorer la carte SIM

TP n°3 – Interaction avec la carte EMV

Correction TP n°1 – Interaction avec la carte SIM - Les premières commandes APDUs en utilisant un outil de script PC/SC

Il faut d'abord télécharger TestResMan et la spécification 3GPP TS 11.11.

Les commandes sont saisies en Hexadécimal dans la fenêtre Command. Dans la fenêtre Output sont visualisées les réponses APDU renvoyées par la carte.

Rappel du format des commandes APDU

Commande APDU						
Entête obligatoire				Corps optionnel		
CLA	INS	P1	P2	Lc	Data field	Le

- CLA (1 octet): Classe d'instructions : indique la structure et le format pour une catégorie de commandes et de réponses APDU.
- INS (1 octet): code d'instruction: spécifie l'instruction de la commande.
- P1 (1 octet) et P2 (1 octet): paramètres de l'instruction.
- Lc (1 octet): nombre d'octets présents dans le champ données de la commande.
- Data field (octets dont le nombre est égal à la valeur de Lc): une séquence d'octets dans le champ données de la commande.

Rappel du format des réponses APDU

Réponse APDU		
Corps optionnel	Partie obligatoire	
Data field	SW1	SW2

- Data field (longueur variable): une séquence d'octets reçus dans le champ données de la réponse.
- SW1 (1 octet) et SW2 (1 octet): Status words (Mots d'état) état de traitement par la carte.

Dans toute la suite, le paragraphe qui commence avec le symbole => contient la commande en Héxa saisie dans la fenêtre Command. Dans la sélection du répertoire GSM, la commande est en rouge. Le paragraphe qui commence avec le symbole <= contient la réponse en Héxa retournée par la carte et qui s'affiche dans la fenêtre Output. Dans la sélection du répertoire GSM, la réponse est en bleu.

Sélection d'un répertoire GSM :

=> SELECT 2 octets du répertoire GSM

Sending (commande envoyée vers la carte): **A0 A4 00 00 02 7F 20**

<= Received (données reçues de la carte): **9F 1A**

Lorsque la carte désire envoyer des données, elle envoie SW=9F XX au terminal pour lui dire qu'elle souhaite lui envoyer XX octets. La valeur XX est en Héxadécimal.

=> GET RESPONSE 1A octets

Sending: **A0 C0 00 00 1A**

<= Received: **00 00 04 7A 7F 20 02 00 00 00 00 00 0D 91 00 12 08 00 83 8A 83 8A 00 00 83 83 90 00**
Normal processing.

Analyse de la réponse : 00 00 04 7A 7F 20 02 00 00 00 00 00 0D 19 00 12 08 00 83 8A 83 8A 00 00 83 83 90 00

Suite d'octets	**Signification**
00 00	Sans signification
04 7A	Taille mémoire non utilisée (1146 octets)
7F 20	Le nom du répertoire
02	Type répertoire
00 00 00 00 00	Sans signification
0D	13 octets de données GSM
19	= 00011001 (bit de poids fort à 0 implique la présentation du code PIN)
00	Nombre de sous-répertoires
12	Nombre de fichiers
08 00	Octets non commentés
83	Code CHV1 (PIN utilisateur) initialisé, 3 essais de PIN1 possibles
8A	Code initialisé, 10 essais PUK1 possibles
83	Code CHV2 (PIN admin) initialisé, 3 essais PIN2 possibles
8A	Code initialisé, 10 essais PUK2 possibles
00 00 83 83	Octets non commentés
90 00	Status word (succès de la commande)

Vérification du code PIN :

=> VERIFY code PIN=0000 (transmis en code ASCII, le chiffre 0 est codé à l'aide de la valeur 30. Le code PIN est complété par des FF pour faire 8 chiffres)
A0 20 00 01 08 30 30 30 30 FF FF FF FF

<= Réponse : **90 00** (en cas d'erreur on obtient 98 04 pour authentication failed).

Lecture de l'IMSI:

Le fichier EF-IMSI (6F 07) est de type transparent, localisé dans le répertoire GSM.

sélection du fichier EF-IMSI
=> A0 A4 00 00 02 6F 07 (SELECT)
<= 9F 0F
=> A0 C0 00 00 0F (GET RESPONSE)
<= 00 00 00 09 6F 07 04 00 1B 00 1B 01 02 00 00 90 00 (taille du fichier : 09 octets)
=> A0 B0 00 00 09 (READ BINARY)
<= 08 29 80 10 36 60 04 81 80 90 00 (IMSI)

On peut interpréter le code IMSI comme suit :
- inverser les quartets de chaque octet IMSI, ensuite omettre le quartet le plus à gauche, prendre les 3 quartets les plus à gauche qui donnent le numéro du pays (voir liste des Mobile Country Code sur wikipedia) : http://fr.wikipedia.org/wiki/Mobile_country_code

- saisir le code IMSI ainsi modifié sans le chiffre le plus à gauche pour retrouver les informations codées dans l'IMSI en utilisant ce lien:
https://www.numberingplans.com/?page=analysis&sub=imsinr

Exemple :
IMSI = 29 80 20 57 20 38 65 32
IMSI inversé = 92 08 02 75 02 83 56 23
IMSI à saisir pour analyse = 208027502835623
(208: Mobile Country Code de la France).

Information on IMSI number range 20801XXXXXXXXX

Country or destination	France
Network operator	Orange France
Network name	Orange F
Network status*	active

Lecture de TMSI et LAI:

Le fichier EF-LOCI (6F 7E) contient une liste d'attributs (TMSI sur 4 octets, LAI sur 5 octets, TMSI-TIME sur 1 octet et Location-Update Status sur 1 octet) :
=> A0 A4 00 00 02 6F 7E (SELECT)
<= 9F 0F
=> A0 C0 00 00 0F (GET RESPONSE)
<= 00 00 00 0B 6F 7E 04 00 11 FF 15 01 02 00 00 90 00 (taille du fichier : 0B= 11 octets)
=> A0 B0 00 00 0B (READ BINARY)
<= 78 08 8F 48 02 F8 10 38 00 00 00 90 00 (TMSI sur 4 octets et LAI sur 5 octets)

Exécution de l'algorithme d'authentification du GSM:

La commande RUN-GSM-ALGO (INS=88) exécute la fonction A3/A8 du GSM. Son argument d'entrée est un nombre aléatoire de 16 octets (RAND). Elle retourne une liste de deux valeurs : la signature SRES (4 octets) et la clé Kc (8 octets) :
=> A0 88 00 00 10 01 23 45 67 89 AB CE DF 01 23 45 67 89 0A BC DE (=RAND)
<= 9F 0C
=> A0 C0 00 00 0C (GET RESPONSE)
<= 11 02 60 F1 67 F0 E8 FE F2 BC 60 00 90 00 (SRES sur 4 octets, Kc sur 8 octets)

Mise à jour du fichier EF-Kc:

La carte SIM ne met pas à jour automatiquement le fichier EF-Kc (6F 20) à la fin de l'exécution de la procédure RUN-GSM-ALGO. Cette opération est réalisée par le mobile grâce à la commande UPDATE-BINARY (INS= D6). La valeur stockée dans le fichier EF-Kc est une liste de deux valeurs, la clé Kc et un octet de validation (00 lorsque le contenu est valide et 07 dans le cas contraire).
=> A0 A4 00 00 02 6F 20 (SELECT)
<= 9F 0F
=> A0 C0 00 00 0F (GET RESPONSE)
<= 00 00 00 09 6F 20 04 00 11 00 BB 01 02 00 00 00 90 00 (taille du fichier : 09 octets)
=> A0 B0 00 00 09 (READ BINARY)

<= FF FF FF FF FF FF FF FF 07 80 90 00
=> A0 D6 00 00 09 01 23 45 67 89 AB CD EF 00 (UPDATE BINARY Kc + octet de validation)
<= 90 00
=> A0 B0 00 00 09 (READ BINARY)
<= 01 23 45 67 89 AB CD EF 00 90 00

Lecture de la table de services SIM:

Le fichier EF-SIM-Service-Table (6F 38) est la liste des services offerts par la carte SIM. Chaque service, identifié par son ordre d'apparition (n°1, n°2, etc.) est associé à deux bits : le premier de poids fort renseigne l'existence du service (1 pour présent), le deuxième de poids faible indique son activation (1 pour actif).
=> A0 A4 00 00 02 6F 38 (SELECT)
<= 9F 0F
=> A0 C0 00 00 0F (GET RESPONSE)
<= 00 00 00 0D 6F 38 04 00 1B 00 BB 01 02 00 00 90 00 (taille du fichier 0D)
=> A0 B0 00 00 0D (READ BINARY)
<= FF 3C 3F 03 00 3C 03 00 0C 00 00 00 90 00

FF = 11 .11 .11. 11, les services 1, 2, 3 et 4 sont présents sur la SIM et sont actifs.
3C = 00.11.11.00 ; les services 5 et 8 sont absents et inactifs.

Le service n°1 permet la désactivation du code PIN utilisateur (CHV1).
Le service n°2 notifie la présence d'un annuaire de numéros abrégés (ADN).
Le service n°3 est un annuaire de numéros non abrégés (FDN).
Le service n°4 indique l'existence d'un fichier EF-SMS réalisant le stockage des SMS.

Les fichiers EF-ADN, EF-FDN et EF-SMS occupent un espace mémoire de l'ordre de plusieurs ko. Ils sont localisés dans le répertoire DF-TELECOM (7F 10).

Sélection du répertoire DF-TELECOM

=> A0 A4 00 00 02 7F 10 (SELECT)
<= 9F 1A
=> A0 C0 00 00 1A (GET RESPONSE)
<= 00 00 04 7A 7F 10 02 00 00 00 00 00 0D 11 00 09 08 00 83 8A 83 8A 00 00 83 83 90 00

Lecture et écriture des SMS dans la SIM:

Le fichier EF-SMS (6F 3C) est un fichier cyclique.

=> A0 A4 00 00 02 6F 3C (SELECT)
<= 9F 0F
=> A0 C0 00 00 0F (GET RESPONSE)
<= 00 00 14 A0 6F 3C 04 00 11 00 BB 01 02 01 B0 90 00

Taille du fichier : 14 A0 = 5280 octets, taille de l'enregistrement : B0 = 176 octets. Donc, il y a 30[1] enregistrements (5280/176).
Les conditions d'accès sont définies dans les octets 9, 10 et 11 (11 00 BB) :

[1] En héxa 1E

- les commandes READ, SEEK et UPDATE sont contrôlés par CHV1
- les commandes REHABILITATE et INVALIDATE sont contrôlées par ADM

READ RECORD (INS=B2) permet de lire des enregistrements :
=> A0 B2 01 04 B0 (lit l'enregistrement 1)
<= 01 07 91 33 86 09 40 00 F0 04 05 85 02 09 F4 00 F1 70 90 92 91 63 90 80 96 CD A7 … FF FF FF FF 90 00

Suite d'octets	**Signification**
01	Indication de présence d'un SMS (00 sinon)
07	Longueur de l'attribut SMS information
91 33 86 90 40 00 F0	Numéro du centre de SMS
04	Le premier octet d'un message SMS deliver
05	La longueur du numéro de l'émetteur du SMS
85	Type du numéro de l'émetteur 1000 0101
02 09 F4	Numéro de l'émetteur
00	L'octet TP-PID (Protocol Identifier)
F1	L'octet TP-DCS (Date Coding Scheme)
70 90 92 91 63 90 80	TP-SCTS (time stamp, l'heure d'émission)
96	TP-UDL (User Date Length), longueur du message 150 octets

La taille maximale d'un SMS est de 176 octets.

La commande UPDATE-RECORD (INS=DC) permet d'écrire un SMS dans le fichier EF-SMS :
=> A0 DC 01 04 B0 [176 octets]
<= 90 00

Lecture de l'annuaire des numéros ADN :

Le fichier cyclique EF-ADN (6F 3A) est un annuaire téléphonique.

=> A0 A4 00 00 02 6F 3A (SELECT)
<= 9F 0F
=> A0 C0 00 00 0F (GET RESPONSE)
<= 00 00 1B 58 6F 3A 04 00 11 00 22 01 02 01 1C 90 00

Taille du fichier : 1B 58 = 7000 octets, taille d'un enregistrement : 1C (28 octets)
D'où : 7000/28= 250 enregistrements.

Lecture du 1er enregistrement :
=> A0 B2 01 04 1C
<= 11 53 65 72 76 2E 20 43 6C 69 65 6E 74 FF 03 81 23 53 FF FF FF FF FF FF FF FF FF FF 90 00

Suite d'octets	**Signification**
11 53 65 72 76 2E 20 43 6C 69 65 6E 74 FF	Serv. client (étiquette de 14 octets associée au numéro de téléphone)
03	Longueur en octets des attributs TON/NPI et numéro de téléphone (3=1+2)

81	Type de numéro de téléphone, 81 pour le GSM
25 53	Numéro de téléphone
FF	Capability/configuration identifier, attribut non utilisé
FF	Extension1 record identifier, attribut non utilisé

En général, un numéro de téléphone comporte 10 chiffres (soit 5 octets). Par exemple le numéro 06 22 58 63 78 sera inscrit dans l'annuaire sous la forme : 06 22 58 63 78

Lecture de l'annuaire de service FDN :

Le fichier cyclique EF-FDN (6F 3B) est un annuaire des services.

=> A0 A4 00 00 02 6F 3B (SELECT)
<= 9F 0F
=> A0 C0 00 00 0F (GET RESPONSE)
<= 00 00 03 48 6F 3B 04 00 12 00 BB 01 02 01 1C 90 00

Taille du fichier : 03 48 = 840 octets, taille d'un enregistrement : 1C (28 octets)
D'où : 840/28= 30 enregistrements.

Lecture du 1er enregistrement :
=> A0 B2 01 04 1C
<= 6 D 6F 62 69 63 61 72 74 65 FF FF FF FF FF 02 81 22 FF FF FF FF FF FF FF FF FF FF 90 00

Opérations liées au code PIN :

Un code PIN comporte 8 octets, représentant des chiffres au format ASCII. Les octets insignifiants sont codés par FF.
La commande **`VERIFY CHV`** permet de présenter le code PIN (CHV1) du propriétaire du mobile à la carte SIM. Elle est codée sous la forme : A0 20 00 P2 08 [PIN]. P2 = 01 dans le cas de CHV1 (PIN utilisateur) et égal à 02 si CHV2.

=> A0 20 00 01 08 30 30 30 30 FF FF FF FF
<= 90 00

L'usage du code PIN utilisateur est annulée grâce à la commande **`DISABLE PIN (A0 26 00 01 08 [PIN]).`** La commande **`ENABLE PIN (A0 28 01 08 [PIN])`** réalise l'opération inverse.

Activation du code PIN :
=> A0 28 00 01 08 30 30 30 30 FF FF FF FF
<= 90 00

Annulation de code PIN
=> A0 26 00 01 08 30 30 30 30 FF FF FF FF
<= 90 00

La modification du code est possible grâce à la commande **`CHANGE CHV (A0 24 00 01 10 [Ancien_PIN] [Nouveau_PIN]).`**
=> A0 24 00 01 10 30 30 30 30 FF FF FF FF 31 32 33 34 FF FF FF FF
<= 90 00

Au terme de 3 essais infructueux de présentation du code PIN utilisateur, la carte est bloquée. La commande **UNBLOCK CHV (A0 2C 00 01 10 [PUK] [PIN]** annule cet état et permet au module d'être à nouveau opérationnel. Le code PUK est un code unique de 8 chiffres associé à la SIM.

=> A0 2C 00 01 10 31 32 33 34 35 36 37 38 30 30 30 30 FF FF FF FF
<= 90 00

```
import java.util.List;

import javax.smartcardio.*;

public class ClientPC {

	/**
	 * @param args
	 */
	public static void main(String[] args) {

		// Sélectionner votre lecteur de carte
		TerminalFactory tf = TerminalFactory.getDefault();
		CardTerminals list = tf.terminals();

		List<CardTerminal> terminals = null;
		try {
			terminals = list.list();
		} catch (CardException ce) {
			ce.printStackTrace();
		}
		// Pour avoir la liste explicite
//		System.out.println("Terminals: " + terminals);

		// sélectionne le premier lecteur
		CardTerminal cad = terminals.get(0);


/*
 *	Une autre méthode pour avoir la liste
			Iterator<CardTerminal> iter = list.list().iterator();
			while (iter.hasNext()) {
				System.out.println(iter.next().getName());
			}
		} catch(Exception e){
			e.printStackTrace();
		}
		// sélectionne le terminal souhaité
		CardTerminal cad = list.getTerminal("SCM. SCR35xx USB Smart Card Reader 0");
*/
		if (cad == null)
		{
			System.out.println("Pas de lecteur détecté !");
		}
		else
		{
			try
			{
				if (!cad.isCardPresent())
				{
					System.out.println("Inserer une carte");

					cad.waitForCardPresent(0);
				}

				//  Etablir la connexion avec la carte à puce
				Card c = cad.connect("*");

				//Ouverture d'un canal de communication
				CardChannel canal = c.getBasicChannel();

				CommandAPDU commande = new CommandAPDU(new byte[]
						{
						(byte) 0xA0, (byte) 0xA4, (byte) 0x00,(byte) 0x00, (byte)
0x02,(byte) 0x7F, (byte) 0x20}
				);

	System.out.println("Commande:"+ByteArrayToHexString(commande.getBytes()));

				ResponseAPDU reponse = canal.transmit(commande);

	System.out.println("Reponse:"+ByteArrayToHexString(reponse.getBytes()));

				commande = new CommandAPDU(new byte[]
						{
```

```
                                                (byte) 0xA0, (byte) 0xC0, (byte) 0x00,(byte) 0x00, (byte)
reponse.getSW2()}
                                );

        System.out.println("Commande:"+ByteArrayToHexString(commande.getBytes()));

                                reponse = canal.transmit(commande);

        System.out.println("Reponse:"+ByteArrayToHexString(reponse.getBytes()));

                                commande = new CommandAPDU(new byte[]
                                                {
                                                (byte) 0xA0, (byte) 0x20, (byte) 0x00,(byte) 0x01, (byte)
0x08, (byte) 0x30, (byte) 0x30, (byte) 0x30, (byte) 0x30, (byte) 0xFF, (byte) 0xFF,(byte) 0xFF, (byte)
0xFF }
                                );

        System.out.println("Commande:"+ByteArrayToHexString(commande.getBytes()));

                                reponse = canal.transmit(commande);

        System.out.println("Reponse:"+ByteArrayToHexString(reponse.getBytes()));

                                // Déconnexion
                                c.disconnect(true);

                        } catch(Exception e){
                                e.printStackTrace();
                        }
                }

        }

    /**
     * Utility method to convert a byte array to a hexadecimal string.
     *
     * @param bytes Bytes to convert
     * @return String, containing hexadecimal representation.
     */
    public static String ByteArrayToHexString(byte[] bytes) {
        final char[] hexArray = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
        char[] hexChars = new char[bytes.length * 2]; // Each byte has two hex characters (nibbles)
        int v;
        for (int j = 0; j < bytes.length; j++) {
            v = bytes[j] & 0xFF; // Cast bytes[j] to int, treating as unsigned value
            hexChars[j * 2] = hexArray[v >>> 4]; // Select hex character from upper nibble
            hexChars[j * 2 + 1] = hexArray[v & 0x0F]; // Select hex character from lower nibble
        }
        return new String(hexChars);
    }

    /**
     * Utility method to convert a hexadecimal string to a byte string.
     *
     * <p>Behavior with input strings containing non-hexadecimal characters is undefined.
     *
     * @param s String containing hexadecimal characters to convert
     * @return Byte array generated from input
     * @throws java.lang.IllegalArgumentException if input length is incorrect
     */
    public static byte[] HexStringToByteArray(String s) throws IllegalArgumentException {
        int len = s.length();
        if (len % 2 == 1) {
            throw new IllegalArgumentException("Hex string must have even number of characters");
        }
        byte[] data = new byte[len / 2]; // Allocate 1 byte per 2 hex characters
        for (int i = 0; i < len; i += 2) {
            // Convert each character into a integer (base-16), then bit-shift into place
            data[i / 2] = (byte) ((Character.digit(s.charAt(i), 16) << 4)
                    + Character.digit(s.charAt(i+1), 16));
        }
        return data;
    }

}
```

Correction TP n°3 – Interaction avec la carte EMV - Les premières commandes APDUs en utilisant un outil de script PC/SC

Il faut d'abord télécharger TestResMan et la spécification 3GPP TS 11.11.

Les commandes sont saisies en Hexadécimal dans la fenêtre Command. Dans la fenêtre Output sont visualisées les réponses APDU renvoyées par la carte.

Rappel du format des commandes APDU

Commande APDU						
Entête obligatoire				Corps optionnel		
CLA	INS	P1	P2	Lc	Data field	Le

- CLA (1 octet): Classe d'instructions : indique la structure et le format pour une catégorie de commandes et de réponses APDU.
- INS (1 octet): code d'instruction: spécifie l'instruction de la commande.
- P1 (1 octet) et P2 (1 octet): paramètres de l'instruction.
- Lc (1 octet): nombre d'octets présents dans le champ données de la commande.
- Data field (octets dont le nombre est égal à la valeur de Lc): une séquence d'octets dans le champ données de la commande.

Rappel du format des réponses APDU

Réponse APDU		
Corps optionnel	Partie obligatoire	
Data field	SW1	SW2

- Data field (longueur variable): une séquence d'octets reçus dans le champ données de la réponse.
- SW1 (1 octet) et SW2 (1 octet): Status words (Mots d'état) état de traitement par la carte.

Dans toute la suite, le paragraphe qui commence avec le symbole => contient la commande en Héxa saisie dans la fenêtre Command. Dans la sélection du répertoire GSM, la commande est en rouge. Le paragraphe qui commence avec le symbole <= contient la réponse en Héxa retournée par la carte et qui s'affiche dans la fenêtre Output. Dans la sélection du PSE, la réponse est en bleu.

ATR avec « Card State » :3B 65 00 00 20 63 CB AD 80

Sélection du PSE :

=> SELECT 1PAY.SYS.DDF01 (c'est-à-dire 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 avec une longueur de 0E)

Sending (commande envoyée vers la carte): **00 A4 04 00 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31**

<= Received (données reçues de la carte): **61 20**

Lorsque la carte désire envoyer des données, elle envoie SW=61 XX au terminal pour lui dire qu'elle souhaite lui envoyer XX octets. La valeur XX est en Héxadécimal.

=> GET RESPONSE 20 octets
Sending: **00 C0 00 00 20**

<= Received: **6F 1E 84 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 A5 0C 88 01 01 5F 2D 02 66 72 9F 11 01 01 90 00**
Normal processing.

Analyse de la réponse : 6F 1E 84 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 A5 0C 88 01 01 5F 2D 02 66 72 9F 11 01 01 90 00
(Book 1 section 11.3.4 table 43)

Suite d'octets	**Signification**
6F	Tag d'un FCI Template
1E	Longueur du template
84	Tag d'un DF
0E	Longueur du DF
31 50 41 59 2E 53 59 53 2E 44 44 46 30 31	Nom du DF, ici 1PAY.SYS.DDF01
A5	Tag d'un FCI proprietary Template
0C	Longueur du template
88	Tag pour les SFI des EF du DF courant
01	Longueur des SFI
01	SFI
5F 2D	Tag pour le langage préféré
02	Longueur du code langage
66 72	'fr' en ASCII
9F 11	Tag de l'Issuer Code Table Index
01	Longueur
01	Valeur de l'Issuer Code Table Index
90 00	Status word (succès de la commande)

Il faut décoder le SFI codée sur 5 bits dans la valeur : 0x01=0000 0001b c'est à dire 1.

Lecture du SFI du PSE :

En utilisant la spécification (book 1 section 11.2.2 table 39), on sait que les 3 bits du P2 sont 100b quand P1 est le numéro d'un enregistrement et les 5 bits significatifs sont le SFI concerné, soit 0000 1100b =0x0C

Lecture du 1er enregistrement avec une taille à 00 car on ne sait pas combien d'octets lire :
=> 00 B2 01 0C 00
<= 6C 2B
La carte nous indique que nous aurions du en demander 2B. On recommence donc

=> 00 B2 01 0C 2B
<= 70 29 61 27 4F 07 A0 00 00 00 42 10 10 50 0B 43 42 20 43 4F 4D 50 54 41 4E 54 9F 12 0B 43 42 20 43 4F 4D 50 54 41 4E 54 87 01 01 90 00

On sait d'après les spécifications (book 1 section 12.2.3, tables 46, 47 et 48) comment lire cette réponse.

Suite d'octets	Signification
70	Tag
29	Longueur des données
61	Tag
27	Longueur de l'entrée 1
4F	Tag du nom de l'ADF
07	Longueur du nom de l'ADF
A0 00 00 00 42 10 10	Nom de l'ADF
50	Tag de l'étiquette de l'application
0B	Longueur de l'étiquette de l'application
43 42 20 43 4F 4D 50 54 41 4E 54	Étiquette (en ASCII) : CB COMPTANT
9F12	Tag du nom préféré de l'application
0B	Longueur du nom préféré
43 42 20 43 4F 4D 50 54 41 4E 54	Nom préféré (en ASCII) : CB COMPTANT
87	Tag de la priorité
01	Longueur
01	0000 0001b : Bit de poids fort à 0, elle peut donc être choisi sans confirmation et elle est de priorité 1 (la plus haute possible)
90 00	Status word (succès de la commande)

Lecture du 2eme enregistrement avec une taille à 00 car on ne sait pas combien d'octets lire :
=> 00 B2 02 0C 00
<= 6C 27
La carte nous indique que nous aurions du en demander 27. On recommence donc

=> 00 B2 02 0C 27
<= 70 25 61 23 4F 07 A0 00 00 00 42 20 10 50 09 43 42 20 43 52 45 44 49 54 9F 12 09 43 42 20 43 52 45 44 49 54 87 01 02 90 00

On sait d'après les spécifications (book 1 section 12.2.3, tables 46, 47 et 48) comment lire cette réponse.

Suite d'octets	Signification
70	Tag
25	Longueur des données
61	Tag
23	Longueur de l'entrée 1
4F	Tag du nom de l'ADF
07	Longueur du nom de l'ADF
A0 00 00 00 42 20 10	Nom de l'ADF
50	Tag de l'étiquette de l'application
09	Longueur de l'étiquette de l'application
43 42 20 43 52 45 44 49 54	Étiquette (en ASCII) : CB CREDIT
9F12	Tag du nom préféré de l'application
09	Longueur du nom préféré
43 42 20 43 52 45 44 49 54	Nom préféré (en ASCII) : CB CREDIT
87	Tag de la priorité

01	Longueur
02	0000 0010b : Bit de poids fort à 0, elle peut donc être choisi sans confirmation et elle est de priorité2
90 00	Status word (succès de la commande)

Lecture du 3eme enregistrement avec une taille à 00 car on ne sait pas combien d'octets lire :
=> 00 B2 03 0C 00
<= 6C 1D
La carte nous indique que nous aurions du en demander 1D. On recommence donc

=> 00 B2 03 0C 1D
<= 70 1B 61 19 4F 07 A0 00 00 00 03 10 10 50 04 56 49 53 41 9F 12 04 56 49 53 41 87 01 03 90 00

On sait d'après les spécifications (book 1 section 12.2.3, tables 46, 47 et 48) comment lire cette réponse.

Suite d'octets	**Signification**
70	Tag
1B	Longueur des données
61	Tag
19	Longueur de l'entrée 1
4F	Tag du nom de l'ADF
07	Longueur du nom de l'ADF
A0 00 00 00 03 10 10	Nom de l'ADF
50	Tag de l'étiquette de l'application
04	Longueur de l'étiquette de l'application
56 49 53 41	Étiquette (en ASCII) : VISA
9F12	Tag du nom préféré de l'application
04	Longueur du nom préféré
56 49 53 41	Nom préféré (en ASCII) : VISA
87	Tag de la priorité
01	Longueur
03	0000 0011b : Bit de poids fort à 0, elle peut donc être choisi sans confirmation et elle est de priorité3
90 00	Status word (succès de la commande)

Si on essaye de lire un 4eme enregistrement nous avons une erreur.

=> 00 B2 04 0C 00
<= 6A 83
6A 83 signifie « Record not found ».

Sélection de l'application de priorité 1 (ici CB COMPTANT) :

=> SELECT A0 00 00 00 42 10 10 avec une longueur de 07
Sending (commande envoyée vers la carte): **00 A4 04 00 07 A0 00 00 00 42 10 10**
<= Received (données reçues de la carte): **61 3C**
Lorsque la carte désire envoyer des données, elle envoie SW=61 XX au terminal pour lui dire qu'elle souhaite lui envoyer XX octets. La valeur XX est en Héxadécimal.

=> GET RESPONSE 3C octets
Sending: **00 C0 00 00 3C**

<= Received: **6F 3A 84 07 A0 00 00 00 42 10 10 A5 2F 50 0B 43 42 20 43 4F 4D 50 54 41 4E 54 87 01 01 5F 2D 02 66 72 9F 11 01 01 9F 12 0B 43 42 20 43 4F 4D 50 54 41 4E 54 BF 0C 05 9F 4D 02 0B 32 90 00**
Normal processing.

Analyse de la réponse : 6F 3A 84 07 A0 00 00 00 42 10 10 A5 2F 50 0B 43 42 20 43 4F 4D 50 54 41 4E 54 87 01 01 5F 2D 02 66 72 9F 11 01 01 9F 12 0B 43 42 20 43 4F 4D 50 54 41 4E 54 BF 0C 05 9F 4D 02 0B 32 90 00
(Book 1 section 11.3.4 table 45)

Suite d'octets	**Signification**
6F	Tag d'un FCI Template
3A	Longueur du template
84	Tag d'un DF
07	Longueur du DF
A0 00 00 00 42 10 10	Nom du DF, ici A0 00 00 00 42 10 10
A5	Tag d'un FCI proprietary Template
2F	Longueur du template
50	Tag de l'étiquette de l'application
0B	Longueur de l'étiquette de l'application
43 42 20 43 4F 4D 50 54 41 4E 54	Étiquette (en ASCII) : CB COMPTANT
87	Tag de la priorité
01	Longueur
01	0000 0001b : Bit de poids fort à 0, elle peut donc être choisi sans confirmation et elle est de priorité 1 (la plus haute possible)
88	Tag pour les SFI des EF du DF courant
01	Longueur des SFI
01	SFI
5F 2D	Tag pour le langage préféré
02	Longueur du code langage
66 72	'fr' en ASCII
9F 11	Tag de l'Issuer Code Table Index
01	Longueur
01	Valeur de l'Issuer Code Table Index
9F12	Tag du nom préféré de l'application
0B	Longueur du nom préféré
43 42 20 43 4F 4D 50 54 41 4E 54	Nom préféré (en ASCII) : CB COMPTANT
BF 0C	Tag du FCI Issuer Discretionary Data
05	Longueur des données
9F 4D	Tag d'une entrée de log
02	Longueur de l'entrée
0B 32	Interprétation dans le Book 3, section D4, table 44 0x0B=0000 1011b c'est à dire le SFI=11

	0x32 est le nombre maximum d'enregistrement dans le log de transaction On peut aller les lire avec 00B2015C00 … 00B2325C00 car le SFI devient 0101 1100b=0x5C
90 00	Status word (succès de la commande)

Le FCI ne contient donc pas de PDOL (Tag 9F38) qui est optionnelle. Nous émettrons donc une commande GPO en indiquant l'absence de.

GET PROCESSING OPTION :
=> **80 A8 00 00 02 83 00**
<= Received (données reçues de la carte): **61 10**

Si la PDOL avait été présente nous aurions envoyé : 80 A8 00 00 (longueur PDOL + 02) 83 (longueur PDOL) PDOL

=> GET RESPONSE 0x10 octets
Sending: **00 C0 00 00 10**

<= Received: **80 0E 3D 00 10 01 02 00 18 01 02 01 08 01 02 01 90 00**
Normal processing.

(Book 3, section 6.5.8.4, figure 4)

Suite d'octets	**Signification**
80	Format 1
0E	Longueur
3D 00	AIP (Application Interchange Profile, 2 octets – Book 3, section C1, tables 36 et 37) ici 0x3D00 = 0011 1101 0000 0000b Ne supporte pas la SDA Supporte la DDA Supporte le CHV Faire le Terminal Risk Management Supporte l'authentification de l'émetteur Supporte la CDA
10 01 02 00 18 01 02 01 08 01 02 01	AFL (Application File Locator, groupe de 4 octets, Book 3, section 10.2) 0x10=0001 0000b avec le SFI codé sur les 5 bits les plus significatifs, soit 2 01 : le premier enregistrement à lire 02 :le dernier enregistrement à lire pour ce SFI 00 : nombre d'enregistrements impliqués dans les données d'authentification 0x18=0001 1000b avec le SFI codé sur les 5 bits les plus significatifs, soit 3 01 : le premier enregistrement à lire 02 :le dernier enregistrement à lire pour ce SFI 01 : nombre d'enregistrements impliqués dans les données d'authentification

	0x08=0000 1000b avec le SFI codé sur les 5 bits les plus significatifs, soit 1 01 : le premier enregistrement à lire 02 :le dernier enregistrement à lire pour ce SFI 01 : nombre d'enregistrements impliqués dans les données d'authentification
90 00	Status word (succès de la commande)

SFI=1
=> 00 B2 01 0C 00
<= 6C 17

=>00B2010C17
<= 70 15 57 13 **49 75 89 XX XX XX XX XX** D1 70 32 01 73 80 77 82 40 00 0F 90 00
ATTENTION NUMERO DE CARTE
Book 3, table 34 Tag 57 : Track 2 Equivalent Data
Expiration : 03/17
Service code : 201

=> 00B2020C00
<= 6C 3A

=> 00B2020C3A
<= 70 38 5F 20 1A 53 41 55 56 45 52 4F 4E 2F 44 41 4D 49 45 4E 2E 4D 52 20 20 20 20 20 20 20 20 9F 1F 18 37 33 38 30 37 37 38 32 34 30 30 30 30 30 30 38 32 34 30 30 30 30 30 30 90 00
ATTENTION IL Y A MON NOM

Book 3, table 34 Tag 5F20 : Cardholder Name
Book 3, table 34 Tag 9F1F : Track 1 Discretionary Data

SFI=2
=> 00 B2 01 1C 00
<= 6C 84

=>00B2011C84
<= 70 81 81 5F 25 03 15 03 01 5F 24 03 17 03 31 5A 08 49 75 89 **XX XX XX XX XX** 5F 34 01 00 9F 07 02 FF 00 8E 0C 00 00 00 00 00 00 00 00 02 03 01 03 9F 0D 05 BC 08 F4 A0 00 9F 0E 05 00 50 08 00 00 9F 0F 05 BC 28 F4 F8 00 5F 28 02 02 50 8C 18 9F 02 06 9F 03 06 9F 1A 02 95 05 5F 2A 02 9A 03 9C 01 9F 37 04 9F 34 03 8D 17 8A 02 9F 02 06 9F 03 06 9F 1A 02 95 05 5F 2A 02 9A 03 9C 01 9F 37 04 9F 4A 01 82 90 00

Book 3, table 33 Tag 5F25 : Application Effective Date
Book 3, table 33 Tag 5F24 : Application Expiration Date
Book 3, table 34 Tag 5A : Application Primary Account Number (PAN)
Book 3, table 34 Tag 5F34 : Application Primary Account Number (PAN) Sequence Number
Book 3, table 34 Tag 9F07 : Application Usage Control
Book 3, table 34 Tag 8E : Cardholder Verification Method (CVM) List
Book 3, table 34 Tag 9F0D : Issuer Action Code – Default
Book 3, table 34 Tag 9F0E : Issuer Action Code – Denial
Book 3, table 34 Tag 9F0F : Issuer Action Code – Online
Book 3, table 34 Tag 9F28 : Issuer Country Code

Book 3, table 34 Tag 8C : Card Risk Management Data Object List 1 (CDOL1)
Book 3, table 34 Tag 8D : Card Risk Management Data Object List 2 (CDOL2)
Book 3, table 34 Tag 9F4A : Static Data Authentication Tag List

=> 00 B2 02 1C 00
<= 6C 0C

=>00B2021C0C
<= 70 0a 9f 08 02 00 03 5f 30 02 02 01 90 00
Book 3, table 34 Tag 9F08 : Application Version Number
Book 3, table 34 Tag 5F30 : Service Code

SFI=3
=> 00 B2 01 14 00
<= 6C C1

=>00B20114C1
<= 70 81 BE 9F 46 81 B0 CC 86 FD CC B4 E6 D6 51 79 8E 68 0A 43 19 36 B9 3F CB E9 98 84 61 CE 93 B8 E3 D2 5F 78 F0 FE C7 13 2A 54 FA 0C 7A 30 0A BB E9 E3 9D 32 39 F7 3B 27 69 22 8A FB 92 62 18 4F 0C E2 7E 5D 5F 31 82 FF D0 B5 7A 3F 49 EC 49 B2 B2 5E 74 05 42 24 57 C8 28 79 BC F4 8C 86 8E F5 DB D2 65 09 16 02 E4 90 19 30 77 9A 79 DE E5 BC F7 14 5F E3 DE 82 35 F3 90 E0 07 18 A0 73 AB 58 45 55 4F 49 3E 69 C4 2D 7D A9 99 E8 82 66 DF CE 29 D1 02 78 31 E5 09 47 03 93 08 D2 5E C6 2D 5A 21 BB FB 25 79 07 89 3F BD A1 74 34 99 8C 3E B0 80 A1 A6 0A 96 37 EF 9F 47 01 03 9F 49 03 9F 37 04 90 00

Book 3, table 28 Tag 9F 46 : ICC Public Key Certificate
Book 3, table 28 Tag 9F 47 : ICC Public Key Exponent
Book 3, table 28 Tag 9F 49 : Dynamic Data Authentication Data Object List (DDOL)
Book 3, table 34 Tag 9F 37 : Unpredictable Number

=> 00 B2 02 14 00
<= 6C E3

=>00B20214E3
<= 70 81 E0 90 81 B0 03 04 85 E0 AF 13 C5 69 6F 89 A0 30 AC 2A 07 2B 77 22 BC 45 7F 02 55 91 27 71 85 F8 2D 54 08 46 9F 38 6A 5F 30 F7 48 40 BE EF 39 38 E1 B3 D4 6D 3E A3 38 77 C0 F6 14 92 8C 7B F9 4E 46 95 77 24 2A D4 9B DE 55 F3 04 9F CE 13 E5 9E 6A 9D 79 33 2F 7C 3C FF 88 FE 87 98 2C 5D 73 1F 3D 23 3A F9 78 F5 2F BC C7 84 6C A4 50 99 95 E2 65 39 A6 50 93 BE A5 F3 78 AB CD 06 84 45 09 B4 B4 44 50 85 7E 3F E4 19 65 DB 54 14 99 21 DF 15 BA 63 5A 7B 5A 34 7F 1A 05 E9 44 04 5E 29 66 31 3C 63 7A 88 D8 28 DD 13 11 59 D8 4E 80 02 48 AE 31 D8 37 38 8F 01 07 9F 32 01 03 92 24 48 FB 26 06 E0 19 61 B6 8E 4C 59 FD E9 7E 5A 7B 0D 52 BC 2E 77 A3 4A 3E BA 65 3C B7 BD 53 5A 93 B6 0D 52 65 90 00

Book 3, table 27 Tag 90 : Issuer Public Key Certificate
Book 3, table 28 Tag 8F : Certification Authority Public Key Index
Book 3, table 28 Tag 9F 32 : Issuer Public Key Exponent
Book 3, table 27 Tag 92 :Issuer Public Key Remainder