Les cartes à puces et JAVA
Le package javax.smartcardio

Geoffroy Vibrac

APSIDE GROUPE

THE INTERNATIONAL INSTITUTE OF SUPINFO INFORMATION TECHNOLOGY

Developpez.com

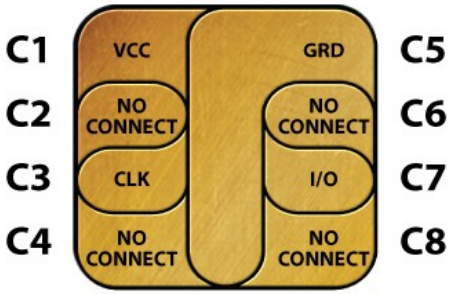
Plan

- Pourquoi parler des cartes à puces
- Que lire ? Cartes à puce, tags ...
- Avec quoi lire ? Les lecteurs
- Comment lire ? Les commandes APDU
- Avant Java 6
- le package smartcardio
- Demos

Pourquoi parler des cartes à puces

- Opportunités de Business !
  - Nombreuses applications autour de la carte à puce
    - Contrôle d'accès
    - Paiement
    - Fidélisation
    - Téléphonie mobile
    - Transport
    - Santé …
  - Croissante forte du RFID et NFC
  - Internet des objets

Que lire ? Carte à puce à contact



- 1ers brevets en 1974, 1ère carte à microprocesseur 1979
- Nombreuses normes. Principale : ISO 7816
- Carte à mémoire / synchrones
  - Mémoire uniquement + I/O
  - Peu sécurisée, plus utilisée
  - 1ères cartes téléphoniques
- Carte à microprocesseur
  - 8 bits / 4 MHz
  - Mémoires ROM, RAM + EEPROM ou Flash
  - Intelligence dans la carte pour plus de sécurité
  - On peut personnaliser les applications (JavaCard)

Que lire ? Carte à puce sans contact

- Fonctionne par radiofréquence (ex 125kHz ou 13,56MHz)
- Nombreuses normes ex : ISO 14443 (13,56Mhz) 1 norme MAIS 2 protocoles 14443/A et 14443/B et 4 parties => nombreuses incompatibilités
- A mémoire ou à microprocesseur.
- Tag RFID/NFC
- Smart object

Avec quoi lire ? Les lecteurs

- Avant : lecteurs non standards
- 1997 : Spécifications PC/SC (Personal computer/Smart Card)
  - Core members : Oracle, Toshiba, Microsoft, Gemalto
  - But : « promouvoir l'interopérabilité des lecteurs de cartes et facilité le développement d'applications »
  - Version 2.01.09 (avril 2010)
  - http://www.pcscworkgroup.com

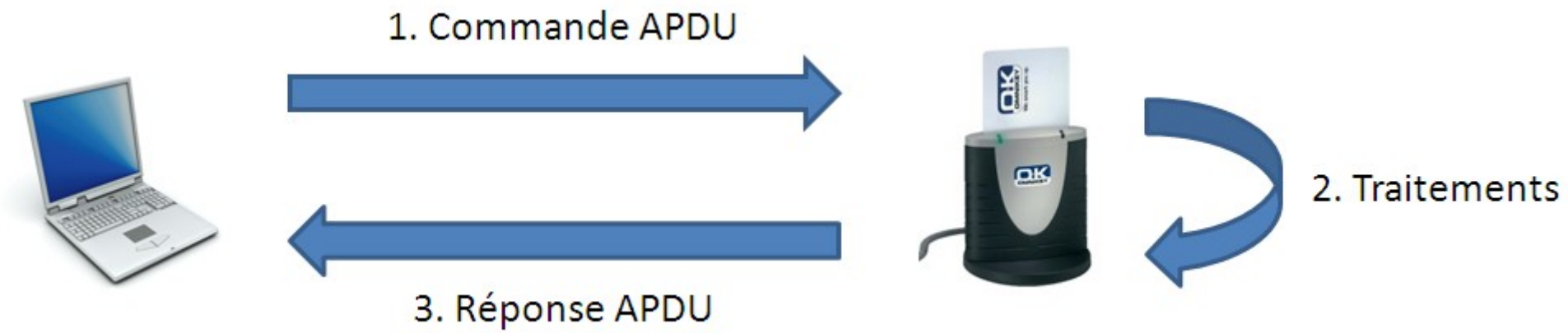
PC/SC Workgroup

Avec quoi lire ? Les lecteurs

- Lecteur PC/SC
  - Connectique USB
  - Pilote générique (CCID ou ICCD) ou constructeur
  - Multiplateforme
    - Windows (api WinSCard)
    - Linux et Mac OS X avec PCSC-Lite & openCCID
  - Peu cher
    - À contact : 10 / 20 Euros
    - Sans contact : 70 / 90 Euros

Comment lire ? Mode de communication

- Communication transactionnelle à 3 étapes :



APDU : Application Protocol Data Unit

Comment lire ? Commandes APDU

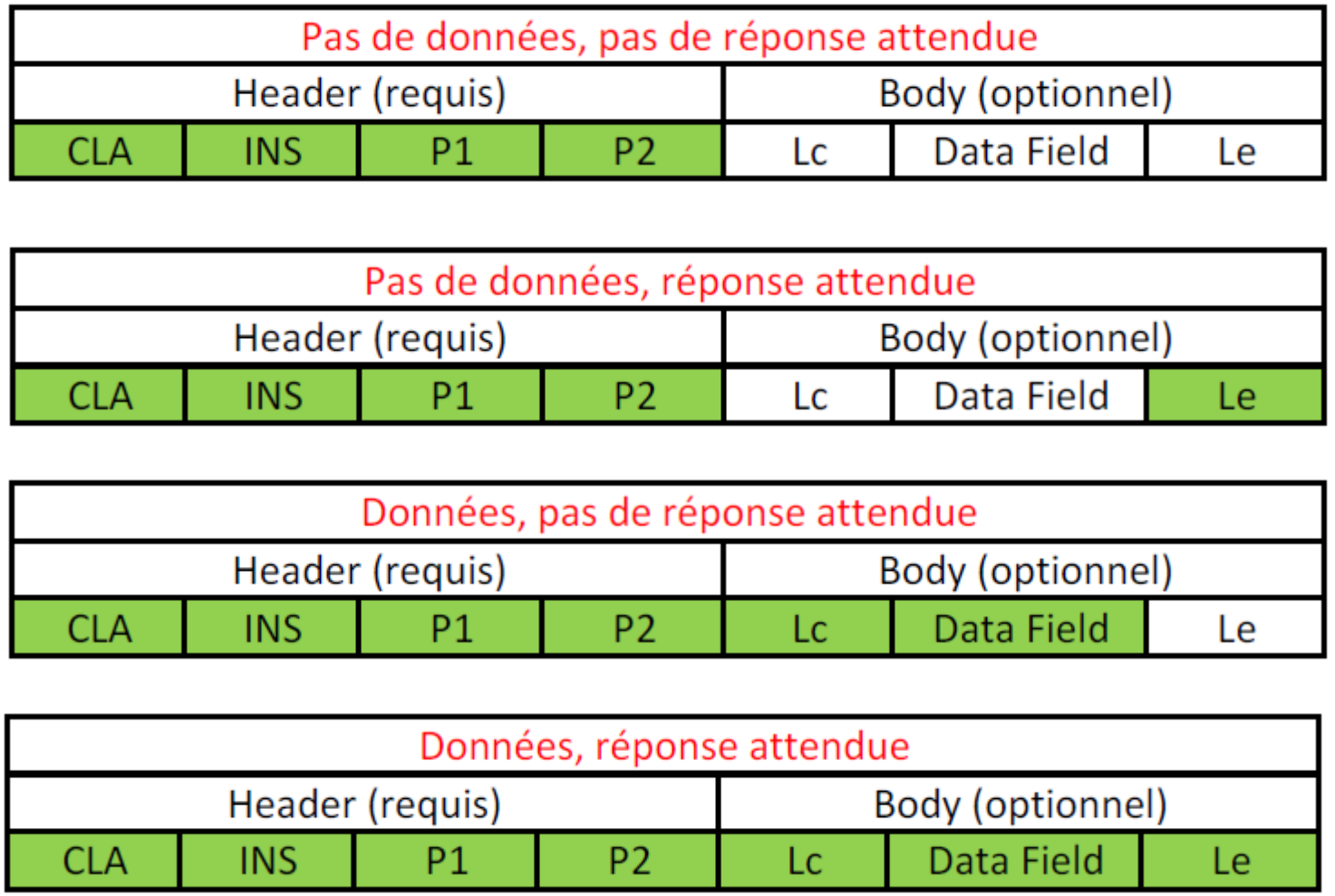
- Format des commandes APDU : on manipule des octets

Commande APDU						
Header (requis)				Body (optionnel)		
CLA	INS	P1	P2	Lc	Data Field	Le

- **CLA (1 octet)** : classe de la commande
  - Exemple : 00 (defaut), A0 (GSM), FF (selection protocol)
- **INS (1 octet)** : code d'instruction = la « procédure » a exécuter
  - Exemple : A4 « select file », CA « get data », B2 « read record »
- **P1 (1 octet), P2 (1 octet)** : paramètres de l'instruction
- **Lc (1 octet)** : Taille du champs Data Field en octet
- **Data Field** : séquence d'octet de longueur Lc
- **Le (1 octet)** : taille maxi de la réponse attendue = LEN

Comment lire ? Commandes APDU

- 4 types de commandes possible :



Comment lire ? Réponses APDU

- Format des réponses

Reponse APDU		
Body (optionnel)	Statut	
Data Field	SW 1	SW 2

- Data Field : séquence d'octet contenant la réponse
- SW 1 (1 octet) et SW 2 (1 octet) : Status Word
  - Exemple de code retour :
    - 90 00 : Succès
    - 6E 00 : CLA error
    - 6D 00 : INS error
    - 37 00 : Len error

Avant Java 6

- JPC/SC Java API du M.U.S.C.L.E

- Nécessite une bibliothèque native
  - .so pour linux
  - .dll pour windows

- Encore utilisé
- Permet d'élargir le nombre de machines compatibles

Le package smartcardio : Vue générale

- Java Smartcard I/O API définit par la JSR 268
- 12 classes
- Utilise la couche PC/SC
- Permet de communiquer par APDU
- Windows / Linux / Mac

Le package smartcardio : Les classes

- Les principales classes
  - Avec quoi lire ? Le lecteur
    - CardTerminals
    - CardTerminal
    - TerminalFactory
  - Que lire ? La carte
    - Card
  - Comment lire ? La communication par APDU
    - CardChannel
    - CommandAPDU
    - ResponseAPDU

Le package smartcardio : Le lecteur

- On a besoin d'un CardTerminal
- Il s'obtient dans la liste des CardTeminals
- Liste obtenu par la fabrique TerminalFactory
  - Permet de se connecter à d'autres types de lecteurs
  - PC/SC par défaut getDefault()

```
// j'instancie une fabrique par défaut = PC/SC
TerminalFactory factory = TerminalFactory.getDefault();
//je créé un objet représentant les lecteurs PC/SC
CardTerminals readers = factory.terminals();
// je peut alors instancier une liste de lecteurs
List<CardTerminal> readersList = readers.list();
// je parcours la liste avec la methode getName() pour récupérer les noms des lecteurs
for (CardTerminal ct : readersList){
    System.out.println(ct.getName());
}
// j'instancie un lecteur depuis son nom
CardTerminal myReader = readers.getTerminal("Gemplus USB Smart Card Reader 0");
```

Le package smartcardio : La carte

- Les méthodes waitForCardPresent() et isCardPresent() de CardTerminal permettent d'attendre et tester l'insertion carte
- Connexion avec connect(String protocol) de CardTerminal (protocol : protocoles de communication T=0, T=1, *)
- Déconnexion avec la méthode disconnect() de Card

```
// je créé un objet carte null
Card card = null;
System.out.println("Attente de la carte");
// attente de la carte pendant 5 secondes
myReader.waitForCardPresent(5000);
// si la carte est présente => on se connecte à la carte
if (myReader.isCardPresent()){
    // connexion avec tout protocol disponible
    card = myReader.connect("*");
    if (card != null){
        // j'affiche le protocole
        System.out.println("Protocol de la carte : " + card.getProtocol());
        //déconnexion de la carte
        card.disconnect(true);
    }
}
```

Le package smartcardio : Communication

- Récupération d'un canal de communication avec getBasicChannel()
- Création d'une commande APDU avec la classe CommandAPDU
  - 9 constructeurs
  - Pas de setters
- Transmission via le canal avec transmit()
- Retourne une ResponseAPDU
  (contenu de la réponse avec getBytes())

Pas de données, pas de réponse attendue						
Header (requis)				Body (optionnel)		
CLA	INS	P1	P2	Lc	Data Field	Le

Pas de données, réponse attendue						
Header (requis)				Body (optionnel)		
CLA	INS	P1	P2	Lc	Data Field	Le

Données, pas de réponse attendue						
Header (requis)				Body (optionnel)		
CLA	INS	P1	P2	Lc	Data Field	Le

Données, réponse attendue						
Header (requis)				Body (optionnel)		
CLA	INS	P1	P2	Lc	Data Field	Le

```
// récupération d'un canal de communication avec la carte
CardChannel channel = myCard.getBasicChannel();
// Création de la commande APDU : demande de l'UID de la carte
// CLA : 0xFF : demande de protocol
// INS : 0xCA : get data
// P1, P2 : 0x00 pas de paramètre
// Le : 256 byte maxi
CommandAPDU commandeAPDU = new CommandAPDU((byte) 0xFF, (byte) 0xCA, (byte) 0x00, (byte) 0x00, 256);
// transmission de la commande par le canal qui retourne la réponse
ResponseAPDU reponseAPDU = channel.transmit(commandeAPDU);
// affichage de la réponse
System.out.println(Util.byteArrayToHexString(reponseAPDU.getBytes()));
```

Demos

1. Lecture d'une carte multiservices :
   - Contrôle d'accès
   - Paiement
   - Accès bibliothèque

2. Lecture d'un tag mifare

Merci de votre attention

geoffroy.vibrac@gmail.com

twitter : @gVibrac