DEMONEY : description et cas d'application au projet GECCOO

Nicolas Stouls - LSR / IMAG

Le 24 mars 2004

Plan

I. Présentation générale de DEMONEY

II. Spécification de DEMONEY

III. Propriétés et objectifs de sécurité

IV. Geccoo

I. Présentation générale de DEMONEY

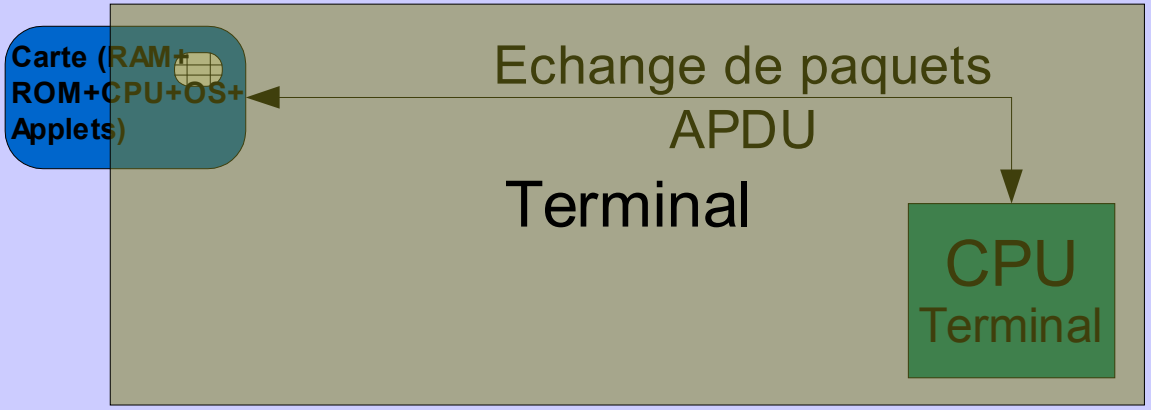
DEMONEY

- Applet Java Card
  - *Développée par :* ***Trusted Logic***
  - *Cadre :* ***projet SecSafe***
  - *Objectif : faire une* ***étude de cas*** *représentative des Applets Java Card*
- Porte monnaie électronique
  - *Plate forme cible :* ***carte à puce***
  - *OS :* ***Java Card 2.1***
- Différentes implantations avec différentes API mais le même comportement

Fonctionnalités

- Une seule monnaie
- Lecture des informations publiques
- Débit dans un magasin
- Crédit sur un terminal de banque en liquide ou par virement bancaire
- Communication avec une autre Applet
  - carte de fidélité
- Administration
  - Déblocage clé, modification paramètres, etc

Concepts de base d'une architecture carte à puce



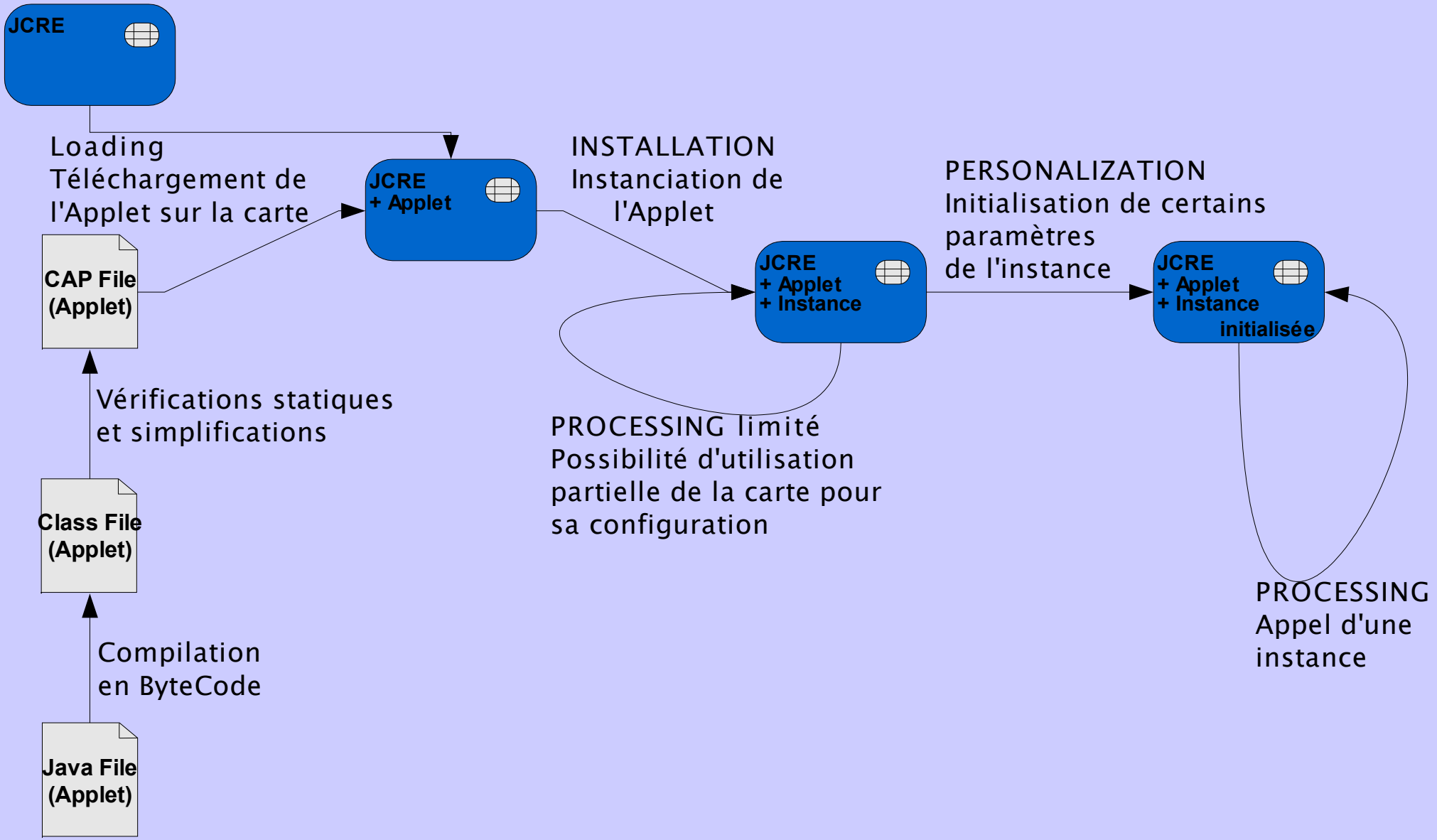
- La carte possède :
  - CPU, RAM, ROM, OS *(Java Card)* et programmes *(Applets)*
- Une communication est menée par le terminal
  - La carte ne fait que répondre
- Communication = échange de paquets APDU
  - APDU = Tableau de bytes contenant une commande et des données

Description d'un APDU

Tableau de byte utilisé pour les transferts entre la carte et le terminal

Nom du champ	Taille	Description
CLA	*1 byte*	*Code de la classe de l'instruction*
INS	*1 byte*	*Code de l'instruction*
P1	*1 byte*	*Code de l'instruction*
P2	*1 byte*	*Code de l'instruction*
LC	*1 byte*	*Taille de la donnée*
DATA	*LC bytes*	*Donnée*
LE	*1 byte*	*Taille de la réponse attendue*

Cycle de vie d'une applet



Organisation des couches logicielles de la carte avec Open Platform

2 interfaces différentes

- Interface avec le système d'exploitation
  - Méthodes appelables par l'OS
  - Méthodes appelables par les autres Applets
- Interface avec le terminal
  - APDU envoyés à l'OS puis passés à l'Applet
  - Commandes reçues par la méthode process

II. Spécification de DEMONEY

Variantes d'implantation

- **Trois implantations du même porte monnaie :**
  - Stand-Alone :
    - *DEMONEY gère toutes les fonctionnalités de sécurité en interne.*
  - Avec API Visa Open platform 2.0 :
    - *DEMONEY ne gère que la signature des réponses.*
  - Avec API Visa Open platform 2.1 :
    - *OP gère : les clefs, le PIN, le chiffrement et la signature*
    - *Messages envoyés à OP qui les déchiffre pour DEMONEY.*
    - *Réponses signées automatiquement.*
- **Note :**
  - *Une valeur par défaut des clefs et du PIN est présente sur la carte.*

Gestion des clefs

- **DEMONEY Standalone :**
  - *La clef $K_{enc/auth}$ sert a :*
    - ***Chiffrer**, **Authentifier** et **Signer** les messages.*
  - *Gestion effectuée par DEMONEY.*
- **DEMONEY OP 2.0 :**
  - *La clef $K_{enc/auth}$ sert a **signer** les messages **émis**.*
    - *Elle est gérée par DEMONEY.*
  - *OP2.0 gère les **autres** clefs et transactions*
- **DEMONEY OP 2.1 :**
  - *OP2.1 gère **toutes** clefs et transactions*

Les clefs utilisées par DEMONEY

- 1 clef = 2-key triple DES sur 16bits
  - Clefs de type symétrique
- Clefs statiques :

Noms possibles	***Rôle***
S-ENC, K_{enc}	Utilisée pour générer $K_{enc/auth}$
MAC-Key, S-MAC, K_{MAC}	Utilisée pour générer C-MAC et R-MAC
DEA, KEK, K_{KEK}	Utilisée pour chiffrer les données sensibles

- Clefs liées à la session :

Noms possibles	***Rôle***
$K_{enc/auth}$	Utilisée pour l'authentification et le chiffrement des messages
C-MAC	Utilisée pour la signature des commandes
R-MAC	Utilisée pour la signature des réponses

Méthodes externes

- **install** :
  - *Méthode statique pour la création d'une instance*
- **processData** :
  - *Utilisé pour la personalisation et avec l'API VOP2.1*
- **select** :
  - *Informe l'Applet qu'elle est sélectionnée*
- **process** :
  - *Transmet une commande APDU*
- **deselect** :
  - *Informe l'Applet qu'elle n'est plus sélectionnée*

Liste des commandes APDU

passées à l'Applet par la commande *process*

Commande	Effet
STORE DATA	*Personalisation des données de la carte*
SELECT	Seul CodeOp imposé par JavaCard
INITIALIZE UPDATE	Non implanté dans la version OP2.1
EXTERNAL AUTHENTIFICATION	
PUT KEY	
PIN CHANGE PIN UNBLOCK	Implanté que dans la version Standalone
PIN VERIFY	*Vérifie que le code PIN entré est correct*
INITIALIZE TRANSACTION	*Initialise un crédit ou un débit*
COMPLETE TRANSACTION	*Finalise un crédit ou un débit*
GET DATA	*Renvoie les informations publiques de l'instance*
PUT DATA	*Mise à jour des différentes données internes de la carte*
READ RECORD	Lecture du log

Différents niveaux de sécurité

- Publique
  - Accès aux données publiques *(**GETDATA**, **READRECORD**)*
  - Début de protocole *(**INITIALISZEUPDATE**, **EXTERNALUPDATE**, **VERIFYPIN**)*
- Débit *(terminal de débit et canal chiffré)*
  - Débit de la carte *(**INITIALIZETRANSACTION**[Debit], **COMPLETE TRANSACTION**)*
- Crédit *(terminal de crédit et canal chiffré)*
  - Crédit de la carte avec des espèces *(**INITIALIZETRANSACTION** [Credit par espèces])*
- Crédit-Identifié *(terminal bancaire de crédit, canal chiffré et code PIN vérifié)*
  - Crédit de la carte par virement *(**INITIALIZETRANSACTION**[Credit par virement bancaire])*
- Administration *(terminal d'administration et canal chiffré)*
  - Gestion de la carte *(**PUTDATA**, **PUTKEY**, **PINCHANGE**, **PINUNBLOCK**, **STOREDATA**)*

Le niveau de sécurité **Publique** *est suffisant dans la version OP2.1*

INSTALLATION

Instanciation de l'Applet

Objectif :

1. *Instancier une Applet*
2. *Enregistrer l'instance créée dans la liste du JCRE des instances présentes sur la carte*
3. *Initialiser les données de sécurisation (nb d'essais PIN, Données de diversification des clefs, taille du fichier de log)*

Cas d'erreur de l'INSTALLATION

- **Cas d'erreur :**
  - Si pas d'appel de Register()
  - Si une exception est levée
    - *Par Register ou Install*
- **Conséquences :**
  - L'Applet n'est pas instanciée.
  - Le JCRE nettoie la mémoire des éventuelles pollutions produites
- **Pas de *RESET* possible car effectué en milieu sécurisé**
  - ex : banque

PERSONALIZATION

Uniquement dans le cas de l'implantation VISA Open Platform

- Objectif : Initialiser au moins les variables suivantes :
  1. *L'identificateur de porte monnaie*
  2. *Le solde de la carte*
  3. *Le nombre de transactions maximum*
  4. *Le solde maximum*
  5. *Le débit maximum par opération*
  6. *[L'identificateur de compte en banque (pour virements)]*
  7. *[AID d'une carte de fidélité]*

- PROCESSING limité :
  - *Seules les opérations nécessaires au chargement des données attendues peuvent survenir.*

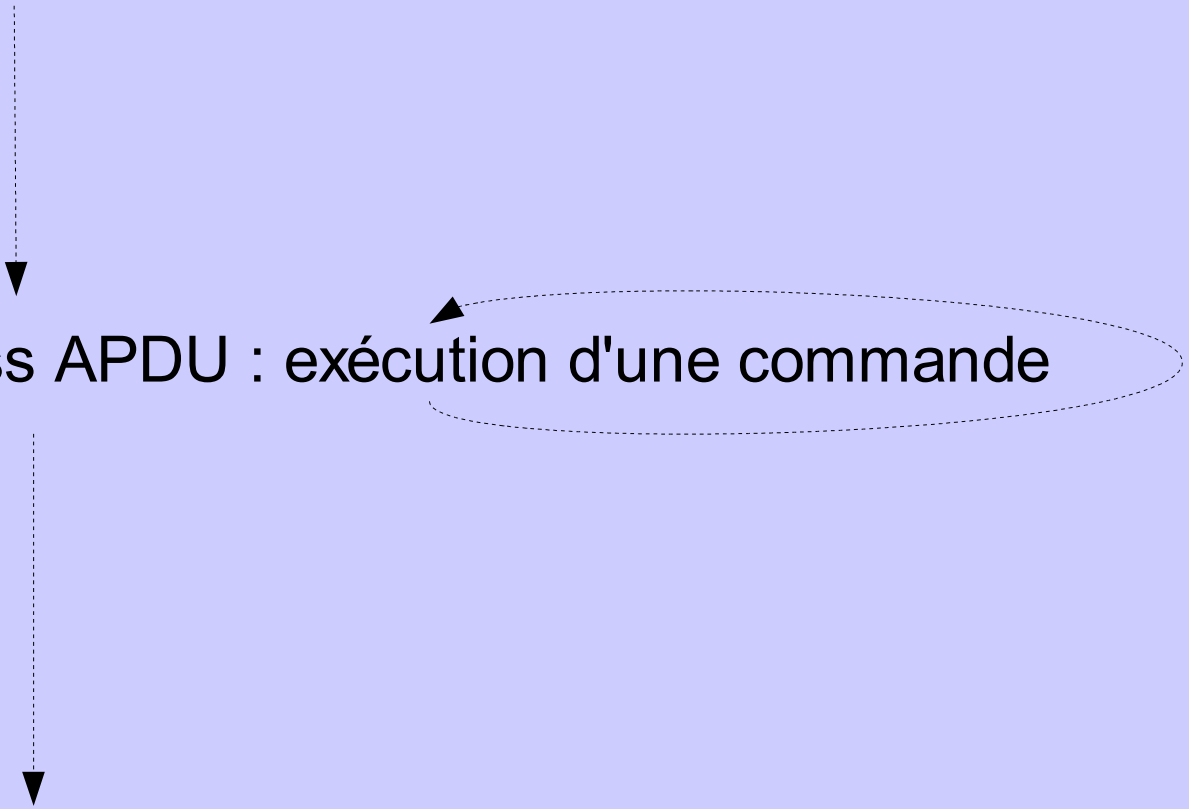
Comportements possible de la PERSONALIZATION

PROCESSING

Utilisation d'une Applet

Select APDU (choix d'un AID – Applet IDentifier)

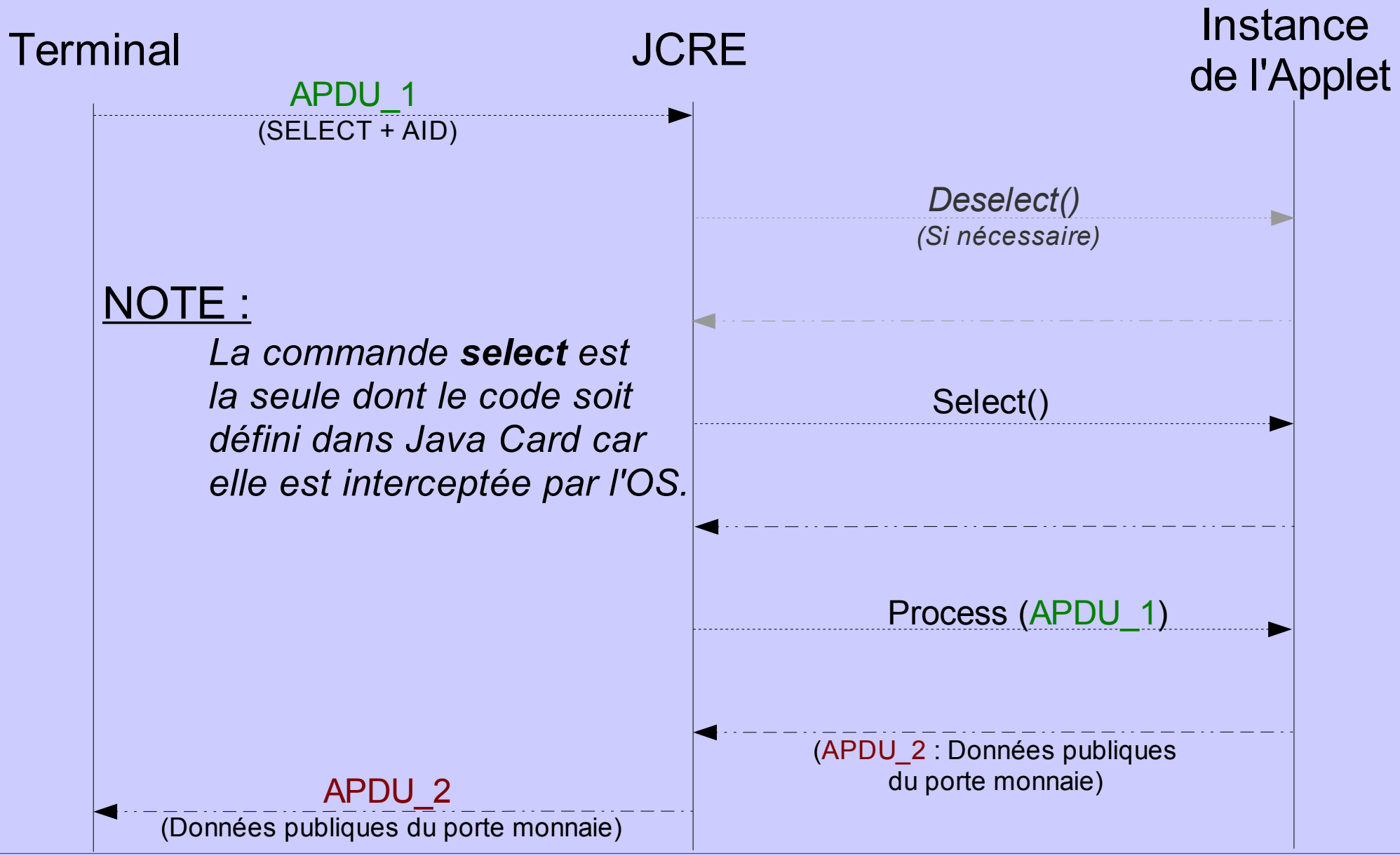
Select : Renvoie les infos publiques de l'instance de l'Applet



Deselect : Par selection d'une applet.

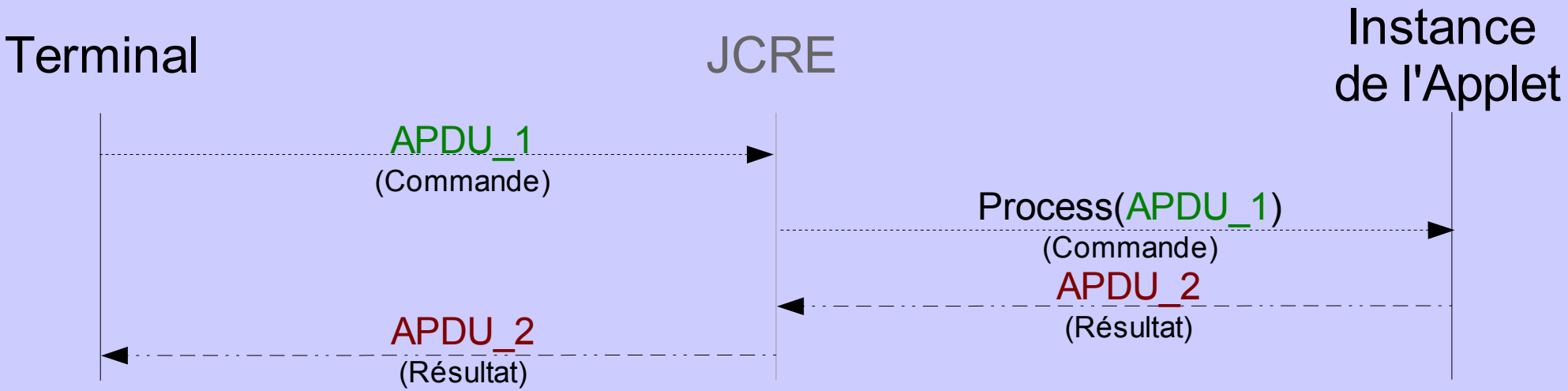
Note : Reset n'appelle pas la méthode Deselect

SELECT APDU


Terminal
JCRE
Instance de l'Applet
APDU_1
(SELECT + AID)
Deselect()
(Si nécessaire)
NOTE :
La commande select est la seule dont le code soit défini dans Java Card car elle est interceptée par l'OS.
Select()
Process (APDU_1)
(APDU_2 : Données publiques du porte monnaie)
APDU_2
(Données publiques du porte monnaie)

PROCESS

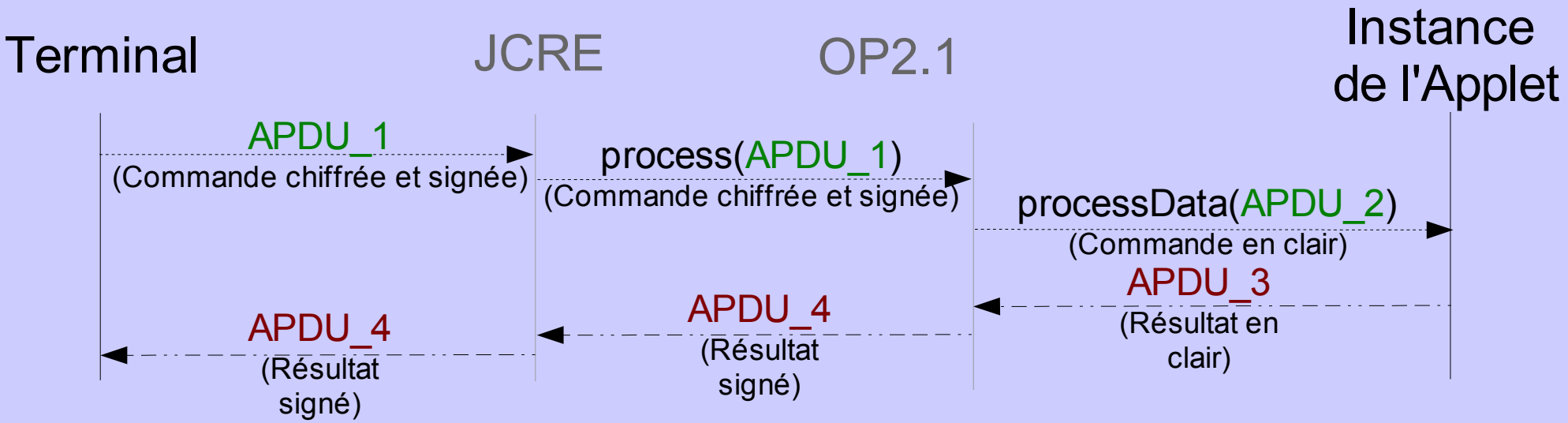
Envoie d'une commande à une instance de l'Applet cas Standalone et Open Platform 2.0



- AID non précisé
  - Message traité par l'Applet sélectionnée
- Information de chiffrement contenue dans CLA

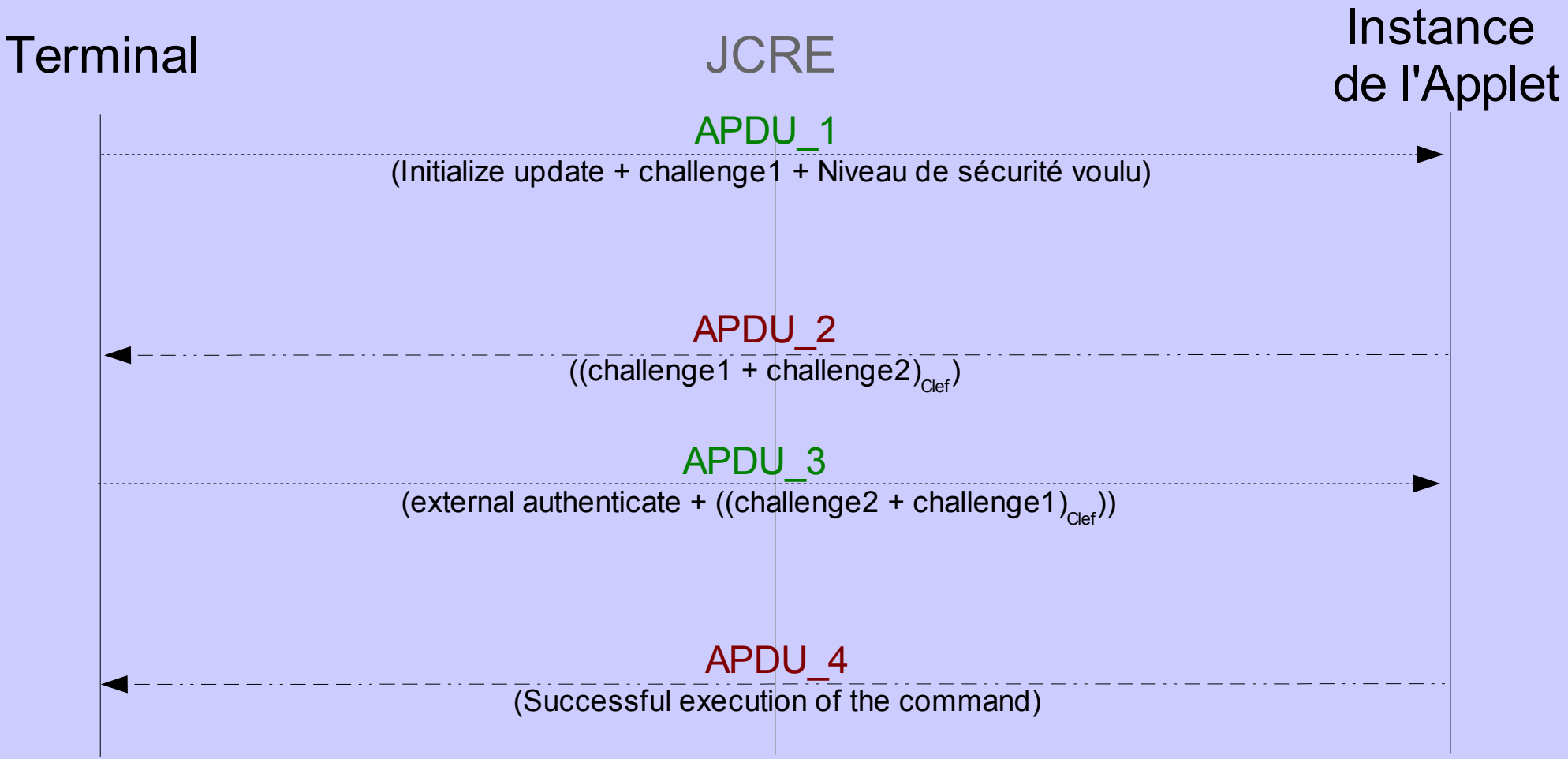
PROCESS

Envoie d'une commande à une instance de l'Applet cas Open Platform 2.1



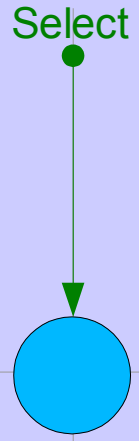
- Open Platform intercepte les messages destinés à DEMONEY et les déchiffrent.
- Les réponses sont signées par OP.

Sécurisation canal



Note : Clef est la clef de session issue de Challenge1 et de la clef privée de l'instance associée au niveau de sécurité voulu.

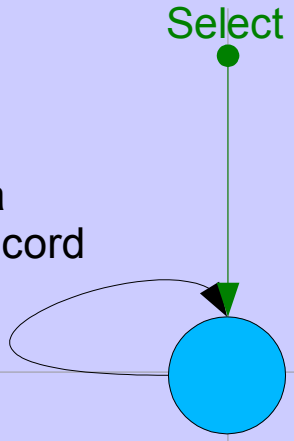
Comportements possibles dans la phase de PROCESSING



Publique
Crédit-Ident

Administration
Débit ou crédit
(suivant le terminal)

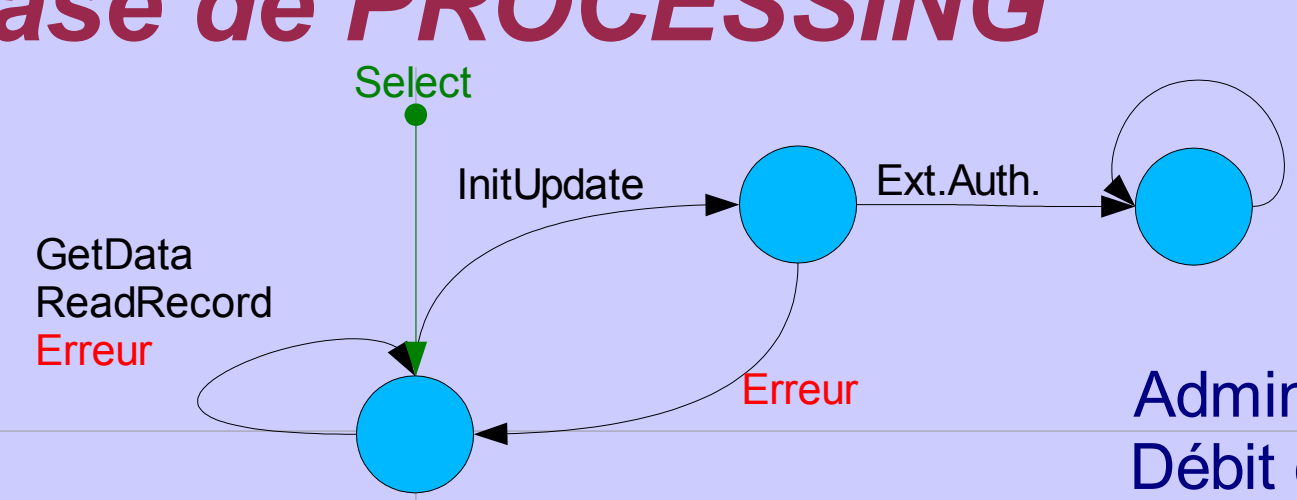
Comportements possibles dans la phase de PROCESSING



Publique
Crédit-Ident

Administration
Débit ou crédit
(suivant le terminal)

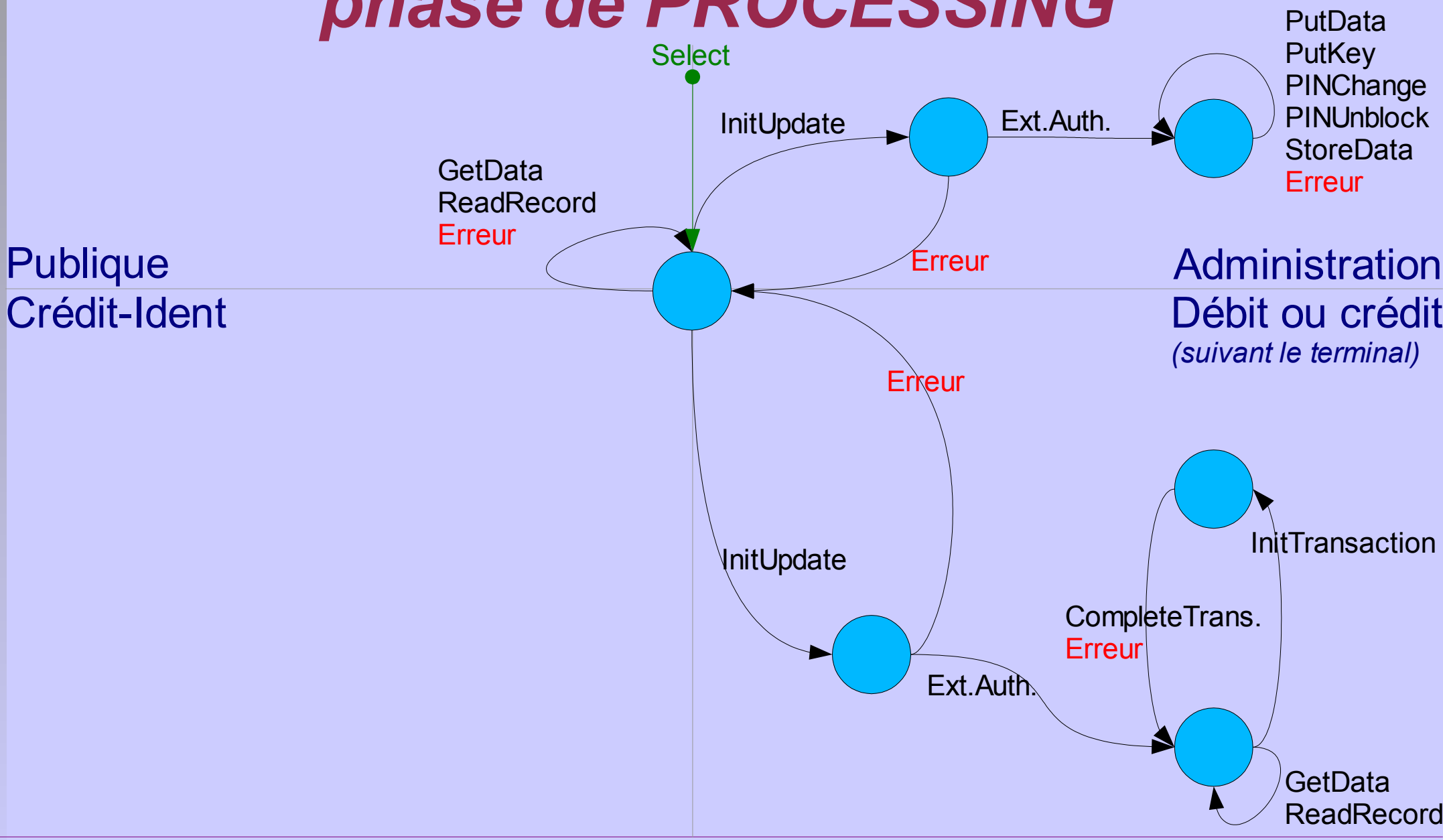
Comportements possibles dans la phase de PROCESSING



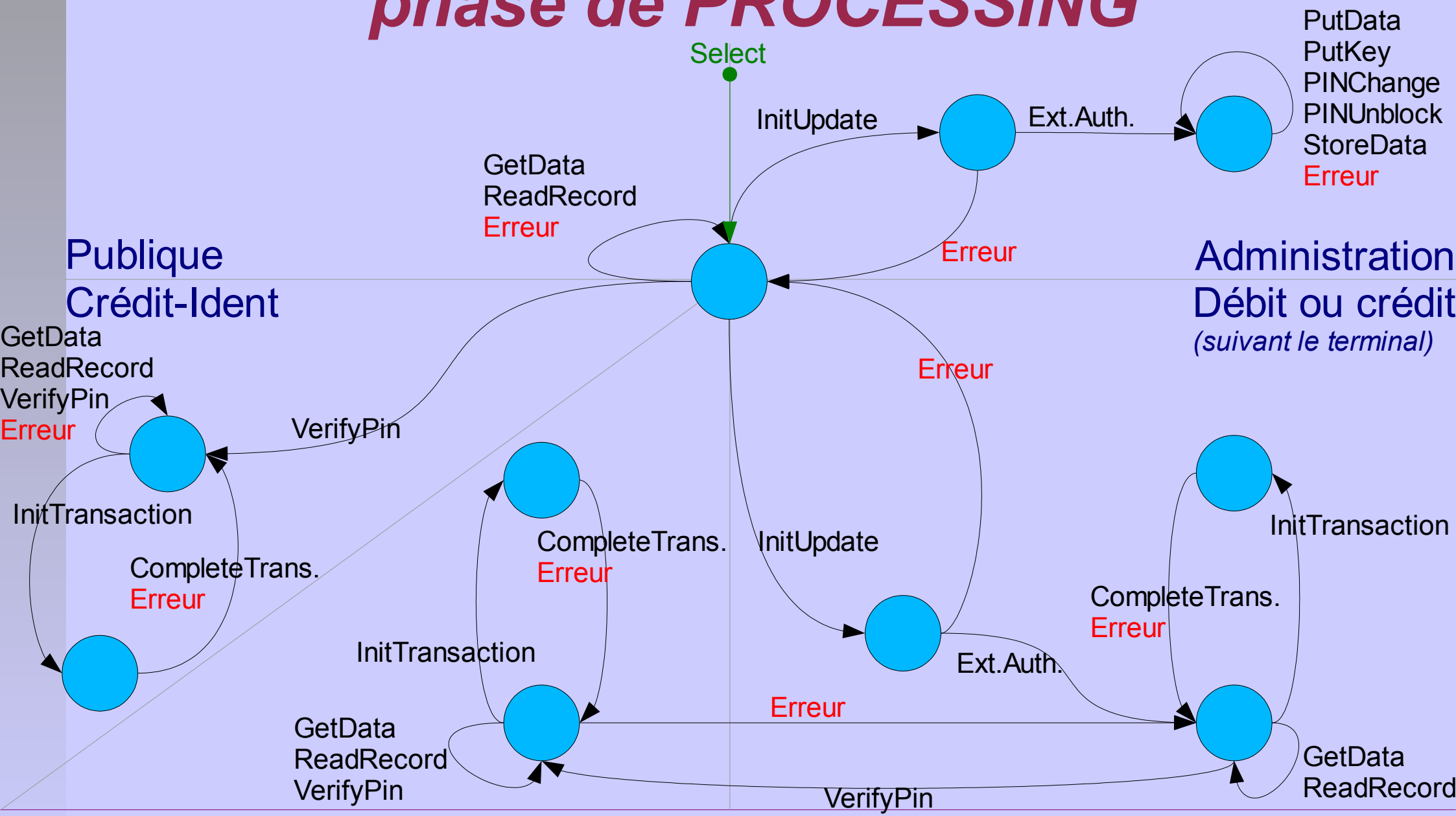
Publique
Crédit-Ident

Administration
Débit ou crédit
(suivant le terminal)

Comportements possibles dans la phase de PROCESSING

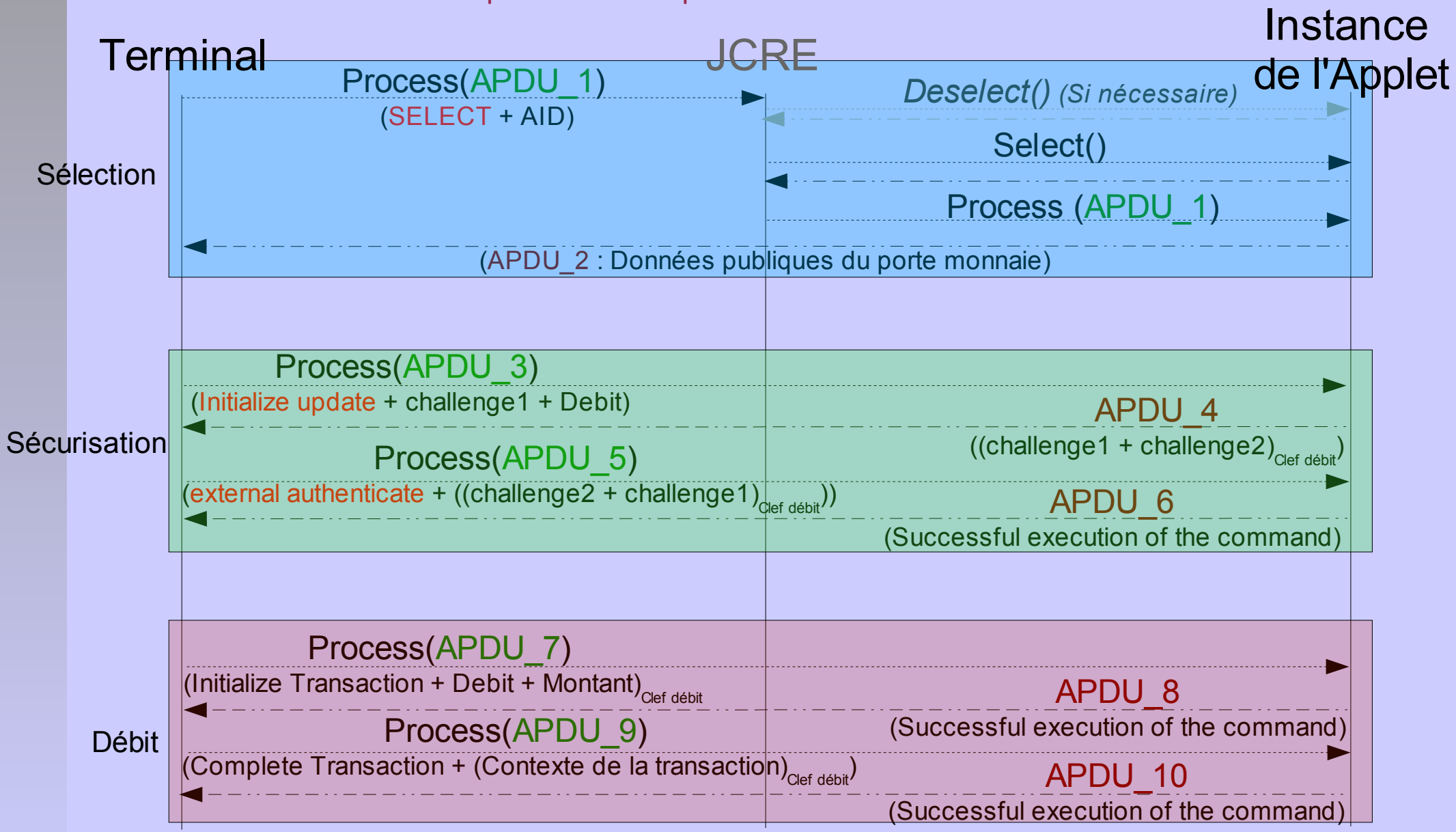


Comportements possibles dans la phase de PROCESSING



Un exemple d'exécution

Cas d'un paiement avec pour la version Standalone ou API OP2.0



III. Propriétés et objectifs de sécurité

Propriétés de DEMONEY

- **ALLOCATION :** *Mémoire permanente allouée qu'à l'initialisation et à la personalisation.*
- **Préservation des secrets :** *Pas de perte directe d'information confidentielle.*
- **Exécution conditionnelle :** *Quelques opérations ne sont possibles que dans certains cas.*
- **Exceptions :** *Seules des erreurs ISOException peuvent être levées.*
  Sauf pour les cas spécifiques à DEMONEY, non prévus dans la norme.
- **Arithmétique :** *Contrôle des dépassements.*
- **Atomicité :** *Utilisation correcte du mécanisme de transaction.*

Objectifs et politiques de sécurité

- **On ne peut pas créer d'argent :**
  - *Authentification mutuelle carte / terminal*
  - *Protection contre les rejeux*
  - *Crédit : récupérer de l'argent avant de créditer*
  - *Le terminal d'achat a un certificat inimitable du débit*
  - *Solde ≥ 0*
- **Il est difficile de perdre de l'argent :**
  - *Solde mis à jour seulement si débit possible*
  - *Crédit : récupérer de l'argent avant de créditer*

Objectifs et politiques de sécurité
(suite)

- **Seul le titulaire du compte peut créditer la carte par virement :**
  - ***Code PIN avec nombre d'essais limité***
  - ***Le terminal a un certificat inimitable qu'un code PIN a été entré.***
- ***Carte à durée de vie limitée :***
  - *Limitation du nombre de transactions faisables*
- ***Intérêt limité pour les pirates :***
  - *Limitation du solde maximum*
  - *Carte a durée de vie limitée*

IV. Geccoo

Génération de code certifié pour les applications orientées objet

Principaux objectifs de Geccoo

- Vérification formelle de programmes à objets.
  - Introduction de niveaux de raffinement
  - Compositionnalité des preuves

  Tâche LSR
- Application au cas particulier des cartes à puce.
  - Programmes simples
  - Propriétés de sécurité explicités
  - Domaine qui intéresse les industriels
- Vérification de propriétés de sécurité

Méthodologies suivies

- **Mise au point d'un langage de spécification**
  - Prise en compte du raffinement
  - Lier les preuves à leur composant et composer les preuves
  - L'un des choix : étendre la méthode B
    - *Etat de l'art : Towards Dynamic Population Management of Abstract Machines in the B Method (B. Aguirre, J. Bicarregui, T. Bimitrakos & T. Maibaum)*
      - *Pas de prise en compte de l'héritage*
      - *Pas de gestion de plusieurs implantations différentes pour une même machine*
  - Autre choix : définition d'un nouveau langage de spécification
  - Utilisation de l'étude de cas DEMONEY
    - *Etudier les problèmes de composition, de gestion dynamique et d'héritage des objets.*
- **Extension et correction du JML**
  - Décoration de DEMONEY
    - *Etudier les problèmes de visibilité des variables dans la composition des invariants.*

Cas de la mise au point d'un langage de spécification

- Simplifications de Java Card:
  - Java Card ne supporte pas de multi-threading
  - Toutes les fonctions des objets sont représentées
    - *Création*
    - *Accès (Méthodes et champs)*
    - *Prise en paramètre*
    - *Retour de fonction*
    - *Cast*
  - Sauf le ramasse miette.
    - *Les objets sont effacés lorsque la RAM n'est plus alimentée.*
  - Objets mis en RAM
    - *sauf lors de l'installation et de la personnalisation*

Intérêt de DEMONEY

- Utilisation de nombreuses fonctionnalités Java Card :
  - *Notamment la communication entre Applets.*
- Spécification détaillée :
  - *Politique de sécurité bien définie ;*
  - *Propriétés explicitées ;*
  - *Comportement entièrement défini.*
- Programme réaliste et raisonnablement simple
- Possibilité de choisir l'API utilisé
  - *Flexibilité de la spécification utilisée.*

Possibilités d'avancées

- **Spécification de DEMONEY :**
  - Exploitation de la spécification donnée dans cet exposé.
  - En B, JML ou autre.
  - Introduction de niveaux de raffinement ?
- **Essais de preuves :**
  - Cohérence de la specification.
  - Correction de la spec par rapport aux propriétés de sécurité.
  - Correction du code par rapport à la spec.
- **Objectif :**
  - Comprendre les besoins de programmes à objets.
    - Trouver comment prouver leur correction
  - Trouver comment factoriser les preuves.
  - Trouver comment recomposer ces preuves avec un faible coût.

Références

- **Présentation de l'application DEMONEY** C. Paulin *(01/2004)*
- **DEMONEY : a démonstrative Electronic purse** R. Marlet et C. Mesnil *(Trusted Logic, 11/2002)*
- **DEMONEY : JavaCard implementation**, R. Marlet *(11/2002 – 3 pages pour préciser l'annexe B)*
- **Security properties and Java Card Specificities to be studied in the SacSafe Project**, R. Marlet, D. Le Metayer *(8/2001 – Spécificités des applets JavaCard. Propriétés à vérifier par analyse statique)*
- **JavaCard 2.1 Platform Specifications**
- **Manual for TL-FIT** Draft, Trusted Logic