Demoney: Java Card Implementation

Date: November 21, 2002 (UTC)
Authors: Renaud Marlet (Trusted Logic)
Classification: Public
Number: SECSAFE-TL-008
Version: 0.8
Status: Draft

Abstract. This document describes specific requirements concerning a Java Card implementation of Demoney, a demonstrative electronic purse for smart cards to be used as a case study to experiment program analyses in the SecSafe project. These requirements are met by the Java Card implementation provided by Trusted Logic.

1 Introduction

Demoney is a demonstrative electronic purse for smart cards. This application is to be used as a case study to experiment program analyses in the SecSafe project.

Document SECSAFE-TL-007 provides a specification of Demoney. The present document list requirements concerning a Java Card implementation of this specification. These requirements are met by the Java Card implementation provided by Trusted Logic.

2 Java Card Implementation Requirements

This section lists the APIs that Java Card implementations of the different Demoney variant must or may use. It also defines how JC implementations of Demoney have to notify a purchase agent.

2.1 Java Card 2.1.1 API

The Java Card 2.1.1 API includes the following features (among others):

- Key management: definition, encryption, decryption,
- Signature management: signature generation and verification,
- Random number generation,
- PIN management: definition, verification,
- Session data: transient arrays,
- Atomicity: starting, committing and aborting transactions.

It is recommended for a Java Card implementation of DEMONEY-STAND-ALONE to use all the above features. An implementation of DEMONEY-OP-2.0 does not need the PIN support as it is handled by OP 2.0. An implementation of DEMONEY-OP-2.1 only needs support for atomicity as all other issues are handled by OP 2.1.
2.2 Open Platform 2.0 API

The Open Platform 2.0 Java Card API provides the following features (among others):

- Full processing of commands INITIALIZE UPDATE and EXTERNAL AUTHENTICATE,
- APDU unwrapping: signature verification and, possibly, message decryption,
- Key decryption and verification,
- Global PIN management.

All these features have to be used to implement DEMONEY-OP-2.0 because this specification variant relies on resources that are managed by OP rather than by the application itself: secure channel based on the personalization key set, global PIN.

2.3 Open Platform 2.1 API

The Open Platform 2.1 Java Card API provides the following features (among others):

- Full processing of commands INITIALIZE UPDATE and EXTERNAL AUTHENTICATE,
- Command message unwrapping: signature verification and, possibly, message decryption,
- Response message wrapping: signature generation,
- Message data decryption and encryption,
- Global PIN management.

All these features have to be used to implement DEMONEY-OP-2.1 because this specification variant relies on resources that are managed by OP rather than by the application itself: keys of all secure channels, global PIN.

2.4 Purchase Agent Notification

Demoney also has a mechanism to notify purchases to another application present on the card, such as a loyalty application or a budget manager. In a Java Card implementation, this mechanism is based on a shareable interface.

When a debit is performed in Demoney, if the AID of a purchase agent is defined, then the following operations are performed:

- A shareable object associated to this purchase agent (via its AID) is requested to the system. The byte parameter to the request method `getAppletShareableInterfaceObject` must be zero.
- This shareable object is cast to the following interface:

```java
package fr.trustedlogic.demo.purchaseAgent;

interface PurchaseAgent {
    public void notifyPurchase(byte[] buffer, short offset);
}
```

- The `notifyPurchase` method is called with a buffer\(^1\) loaded with the information described in Table 1.

---

\(^1\)In fact, because of the Java Card firewall, only the APDU buffer can be read by another applet. The argument buffer of the method `notifyPurchase` can thus only be the APDU buffer.
Because a given card can run Demoney, or the purchase agent, or both, the `PurchaseAgent` interface cannot be in Demoney’s package nor in the actual purchase agent package; it has to be in a separate package. This package is to be loaded before Demoney or the purchase agent is loaded onto the card.

Other payment applications may also communicate to a purchase agent via the same interface.